



**AiCE Undergraduate Research Project  
Final Report**

**Spring 2025 Semester**

***AURA:***  
***AI-Powered Unified Response and Analysis***

**Team Members**

Niracha Janavatara  
Kasidis Manasurangkul  
Pon Yimcharoen  
Supakorn Etitum  
Shine Min Kha

***Advisor***  
***Charnon Pattiyanon***

**05/05/2025**

## Table of Contents

<b>Chapter 1.....</b>	<b>8</b>
<b>Introduction.....</b>	<b>8</b>
1.1 Problem Statement.....	8
1.2 Project Solution Approach.....	8
The proposed solution addresses the inefficiencies in the NHSO call center by automating the data entry process through an AI-powered pipeline. This pipeline leverages Speech-to-Text (STT), summarization, classification, and CRM integration to streamline operations and ensure accurate data capture.....	8
1.2.1 Speech-to-Text (STT):.....	8
1.2.2 Summarization:.....	9
1.2.3 KB Classification/Retrieval:.....	9
1.2.4 Retrieval-Augmented Generation (RAG):.....	9
1.2.5 CRM Integration:.....	9
1.2.6 Data Synthesis:.....	9
1.3 Project Objectives.....	10
1.3.1 Last semester:.....	10
1.3.2 This semester:.....	10
<b>Chapter 2.....</b>	<b>11</b>
2.1 Fundamental Theory and Concepts.....	11
2.1.1 Large Language Models (LLMs).....	11
2.1.2 Transformer Model.....	11
2.1.3 RESTful APIs.....	11
2.1.4 Relational Database.....	11
2.1.5 Knowledge base retrieval and reranking.....	12
2.1.6 Retrieval-Augmented Generation (RAG).....	12
2.1.7 LoRA (Low-Rank Adaptation of Large Language Models).....	13
2.1.8 Supervised Learning.....	13
2.1.9 Data Synthesis.....	13
2.1.10 Multimodal Models.....	14
2.1.11 Vision-Language Models (VLMs).....	14
2.1.12 Speech-to-Text Models.....	14
2.1.13 Speaker Diarization Models.....	14
2.1.14 Agentic LLMs.....	14
2.3.15 Asynchronous Processing.....	15
2.3.16 Bi-Encoder Architecture.....	15
2.1.17 Cross-Encoder Architecture.....	15
2.1.18 Contrastive and Triplet Loss.....	16
2.1.19 Hard Negative Mining.....	16
2.1.20 BM25 Ranking Function.....	16
2.1.21 ColBERT (Contextualized Late Interaction over BERT).....	16
2.2 Evaluation Metrics.....	16
2.2.1 Rouge Score.....	16
2.2.2 Loss Function.....	17
2.2.3 Precision.....	17

2.2.4 Recall.....	17
2.2.5 F1 Score.....	18
2.2.6 Accuracy.....	18
2.2.7 Word Error Rate (WER).....	18
2.2.8 Diarization Error Rate (DER).....	18
2.2.9 Mean Reciprocal Rank (MRR).....	19
2.2.10 Token Intersection-over-Union (Token-IoU).....	19
2.2.11 Latency (P95).....	19
2.3 Technologies.....	19
2.3.1 Whisper.....	19
2.3.2 pyannote-audio.....	20
2.3.3 DeepCut.....	20
2.3.4 JiWER.....	20
2.3.5 BERT.....	20
2.3.6 ThaiBERT.....	20
2.3.7 WangchanBERTa.....	21
2.3.8 Llama 3.....	21
2.3.9 Node JS.....	21
2.3.10 Express js.....	21
2.3.11 MySQL.....	21
2.3.12 Accelerate.....	22
2.3.13 Nvidia Triton Inference Server.....	22
2.3.14 PyTorch.....	22
2.3.15 Hugging Face Transformers.....	23
2.3.16 Kubernetes.....	23
2.3.17 BGE-M3.....	23
2.3.18 FAISS.....	24
2.3.19 Qwen2.5.....	24
2.3.20 VLLM.....	25
2.3.21 Docker.....	25
2.3.22 Docker Compose.....	25
2.3.23 FastAPI.....	25
2.3.24 PostgreSQL.....	26
2.3.25 Elasticsearch.....	26
2.3.26 LLaMA Factory.....	26
2.3.27 Gemini 2.0.....	26
API platform.....	26
2.3.28 Gemma 3.....	26
2.3.29 FlagEmbeddin.....	27
2.3.30 OpenAI API Compatibility.....	27
2.3.31 HNSW Index.....	27
2.3.32 SentencePiece.....	27
2.3.33 ICU Thai Analyzer.....	27
2.3.34 ColBERT Multi-Vector Retrieval.....	28

2.3.35 Triton gRPC Inference (Cross-Encoder Serving).....	28
<b>Chapter 3.....</b>	<b>29</b>
3.1 User Research.....	29
3.2 System Design.....	29
3.2.1 System Artifacts and Components.....	29
3.2.2 Pipeline Process Explanation.....	31
3.3 Model Pipeline Development.....	50
3.3.1 Speech To Text model.....	50
3.3.1.1 Research about STT model.....	50
3.3.1.2 Evaluation Criteria.....	50
3.3.1.3 Model Information.....	51
3.3.1.4 VLLM with STT models.....	52
3.3.1.5 Fine Tuning Speech to text model.....	53
3.3.1.6 Speaker Diarization.....	54
3.3.1.7 Fine Tuning Speaker Diarization.....	54
3.3.1.8 Connecting Speaker Diarization and Speech To Text Model.....	55
3.3.2. Topic Classification.....	55
3.3.2.1 Objective and success criteria.....	55
Transcript Chunking Strategy.....	57
Data Preparation.....	66
Feature Engineering.....	66
Model Development.....	66
Evaluation Methodology.....	66
Results Analysis.....	67
4.2.2.1 Index search comparison.....	67
3.3.3 Summarization model.....	68
3.3.3.1 Generate Label from larger LLM.....	68
3.3.3.2 Re-train the model.....	68
<b>Chapter 4.....</b>	<b>77</b>
4.1 Data Synthesis.....	77
4.2 Model Performance.....	78
4.2.1 Speech To Text.....	78
4.2.1.1 monsoon whisper fine-tuned.....	78
4.2.2 Topic clustering.....	83
4.2.2.1.....	83
4.2.2.3 Interpretation of Results.....	83
4.2.2.4 Example Output.....	83
4.2.3 Summarization.....	83
<b>Chapter 5.....</b>	<b>91</b>
5.1 Summary of Accomplishments.....	91
5.1.1 Speech To Text.....	91
5.1.2 Topic Classification.....	91
5.1.3 Summarization.....	92
5.2 Issues and Obstacles.....	92

5.2.1 Speech To Text.....	92
5.2.1.1 Difficulty in Transcribing Thai Audio.....	92
5.2.1.2 Challenges with Overlapping Speech.....	92
5.2.1.3 Lack of Access to Relevant Datasets.....	92
5.2.2. Data Synthesis.....	92
5.2.3 TopicClassification.....	93
5.2.2.1 Data Limitations.....	93
5.2.2.2 Model Challenges.....	93
5.2.2.3 Computational Constraints:.....	93
5.2.2.4 Evaluation Gaps:.....	93
5.2.4 Summarization.....	93
5.3 Future Directions.....	94
5.3.1 Speech To Text.....	94
5.3.2 Topic Classification.....	95
5.3.2.1 Knowledge Base Classification.....	95
5.3.3 Summarization.....	96
5.3.4 Deployment.....	96
5.4 Lessons Learned.....	96
5.4.1 Speech-to-Text.....	96
5.4.1.1 The Importance of Thorough Testing.....	96
5.4.1.2 GPU Utilization.....	96
5.4.3 Topic Clustering/Classification.....	97
5.4.2.1 Model Selection and Fine-Tuning.....	97
5.4.2.2 Evaluation and Metrics.....	97
5.4.2.3 Working with Missing Data.....	97
5.4.2.4 Proof-of-Concept Development.....	97
5.4.4 Summarization.....	98
5.4.5 Process and Team Dynamics.....	98
5.4.4.1 Time Management.....	98
5.4.4.2 Team Communication.....	98
5.4.4.3 Task Specialization.....	98
5.4.4.4 Adapting to Challenges.....	98
Overall Lessons.....	99
<b>References.....</b>	<b>99</b>

## Chapter 1 Introduction

### 1.1 Problem Statement

In Thailand, access to healthcare services is a fundamental right for every Thai citizen. To achieve the goal of universal health coverage (UHC), the Thai government established the National Health Security Office (NHSO), a public organization that oversees initiatives such as the 30-baht healthcare scheme. This program ensures that citizens have access to affordable medical care. One of the key services provided by the NHSO is the 1330 Contact Center, which handles a high volume of calls daily from individuals seeking medical assistance, information about diseases, or guidance on navigating the healthcare system.

Currently, agents at the 1330 Contact Center manually input data into the Customer Relationship Management (CRM) system during or after each call. This process involves summarizing conversations, identifying the disease or issue mentioned by the caller, selecting the relevant knowledge base (KB) entry used to address the caller's concerns, and completing additional fields in the CRM system.

However, this manual approach introduces several inefficiencies and challenges:

- *Increased agent workload:*

Agents must simultaneously interact with callers and record data in the CRM system, leading to longer call handling times and increased cognitive load.

- *Potential for errors:*

Capturing medical details accurately, such as disease classification and KB mapping, is complex and prone to mistakes, resulting in incomplete or inaccurate CRM records.

- *Delayed response times:*

Time spent on post-call data entry reduces agents' availability for new calls, lowering the overall efficiency of the contact center.

- *Inconsistent data quality:*

Manual processes create variability in the level of detail and accuracy of recorded data, complicating its use for decision-making, reporting, or training purposes.

These inefficiencies underscore the need for an automated solution to streamline CRM data entry, reduce errors, and allow agents to dedicate more time to delivering quality service **to callers**.

### 1.2 Project Solution Approach

The proposed solution addresses the inefficiencies in the NHSO call center by automating the data entry process through an AI-powered pipeline. This pipeline leverages Speech-to-Text (STT), summarization, classification, and CRM integration to streamline operations and ensure accurate data capture.

#### 1.2.1 Speech-to-Text (STT):

- After each call, the recording is processed by an STT model (e.g., Monsoon Whisper) to generate a transcript.
- The STT model is fine-tuned for Thai language data, ensuring accurate transcription of conversations with medical terminology.

#### *1.2.2 Summarization:*

- The generated transcript is passed to a summarization model (e.g., Llama) to create a concise and structured call summary.
- This summary captures the key points discussed during the call, such as the caller's concerns and the agent's responses.

#### *1.2.3 KB Classification/Retrieval:*

- The transcript and/or summary are analyzed using classification models to identify key information, such as:
  - The disease or condition mentioned by the caller.
  - The relevant KB articles used to address the caller's queries.
  - Values for other dropdown fields in the CRM form (e.g., call categories, urgency levels).
- These classification models are trained to handle the nuances of Thai medical conversations.

#### *1.2.4 Retrieval-Augmented Generation (RAG):*

- A retrieval layer is integrated for KB-related tasks to dynamically fetch the most up-to-date entries from the NHSO's knowledge base. This ensures that agents and callers always receive accurate and current information.

#### *1.2.5 CRM Integration:*

- The outputs from the summarization and classification models are automatically populated into the CRM system.
- Fields that previously required manual input are now pre-filled, significantly reducing agent effort and post-call processing time.

#### *1.2.6 Data Synthesis:*

- We synthesized the transcripts from the existing knowledge base to address the issue of not having call recordings from NHSO.

Fields that previously required manual input are now pre-filled, significantly reducing agent effort and post-call processing time.

### **How This Automation Solves the Problem:**

- *Reduced Agent Workload:*

By automating transcription, summarization, and classification, agents are freed from the repetitive task of manual data entry.

- *Enhanced Accuracy:*

AI models ensure consistent and precise data capture, reducing errors in disease classification and KB mapping.

- *Improved Efficiency:*

Automating post-call processes decreases call handling time, allowing agents to attend to more calls.

- *Data Standardization:*

Automated processes result in uniform, high-quality CRM records, making the data more reliable for reporting and decision-making.

This system not only enhances call center efficiency but also improves the quality of service provided to callers, aligning with NHSO's mission of delivering reliable healthcare support to the public.

### 1.3 Project Objectives

#### *1.3.1 Last semester:*

- Develop and evaluate an automated transcription system capable of accurately transcribing Thai call data, focusing on handling multi-accent, noisy environments with multiple speakers typical of NHSO call scenarios.
- Create a proof-of-concept summarization and classification model pipeline that demonstrates the feasibility of automating CRM data entry. This pipeline will simulate key functionalities, including summarizing caller concerns and classifying topics, in preparation for training with real NHSO data.

#### *1.3.2 This semester:*

- Deliver a working prototype that demonstrates end-to-end functionality, including automated transcription, summarization, classification, and CRM integration.
- Develop and test an API that integrates seamlessly with NHSO's CRM system, ensuring real-time data entry, consistency, and reliability.

These objectives aim to reduce agent workload, improve data accuracy, and establish a scalable solution for automating CRM data entry at the NHSO. The proof-of-concept study will provide critical insights into the performance and adaptability of the proposed pipeline, laying the foundation for the prototype and API development.

## Chapter 2 Background

### 2.1 Fundamental Theory and Concepts

#### 2.1.1 Large Language Models (LLMs)

A large language model (LLM) is a type of deep learning model designed to process and generate human-like text by leveraging vast amounts of data and computational power. Built on architectures like Transformers, LLMs are trained on diverse text datasets to learn patterns, semantics, and contextual relationships in language. These models, such as GPT (Generative Pre-trained Transformer), BERT (Bidirectional Encoder Representations from Transformers), and LLaMA (Large Language Model Meta AI), contain billions of parameters, enabling them to understand and generate text across a wide range of tasks, from summarization and translation to question answering and creative writing. By fine-tuning on specific tasks, LLMs can achieve state-of-the-art performance while maintaining adaptability to various domains, making them a cornerstone of modern natural language processing. [1][2]

#### 2.1.2 Transformer Model

The Transformer model, introduced in "Attention Is All You Need" by Vaswani et al. (2017), revolutionized NLP with its self-attention mechanism, enabling it to capture long-range dependencies without relying on recurrence or convolution. Input embeddings combined with positional encodings provide sequence order, while multi-head attention allows the model to focus on multiple relationships simultaneously. The architecture consists of an encoder, which generates representations of the input, and a decoder, which uses these representations to make predictions. Feedforward layers, residual connections, and layer normalization enhance training stability and efficiency. The Transformer's scalability and versatility have made it the foundation for state-of-the-art models like BERT, GPT, and T5. [1]

#### 2.1.3 RESTful APIs

RESTful APIs (Representational State Transfer Application Programming Interface) are interfaces that enable various applications to communicate and exchange data over the internet. [rest api] They use HTTP protocols to perform operations like creating, reading, updating, and deleting data. This approach allows for scalable and efficient interaction between different software components, making RESTful APIs a popular choice for web and mobile application development. Their use of standard methods and stateless communication ensures that applications remain responsive and robust across diverse network conditions.[3]

#### 2.1.4 Relational Database

Relational database is a type of database that stores data that are related to one another.[relational database] it organizes data into tables consisting of rows and columns. Each row, known as a record, has a unique identifier called a key. The columns represent the attributes of the data, establishing a clear structure to store and manage information. Related records in different tables are joined together by foreign keys, which are columns that have the same value in multiple tables to identify the relationship. [4]

# Relational Database

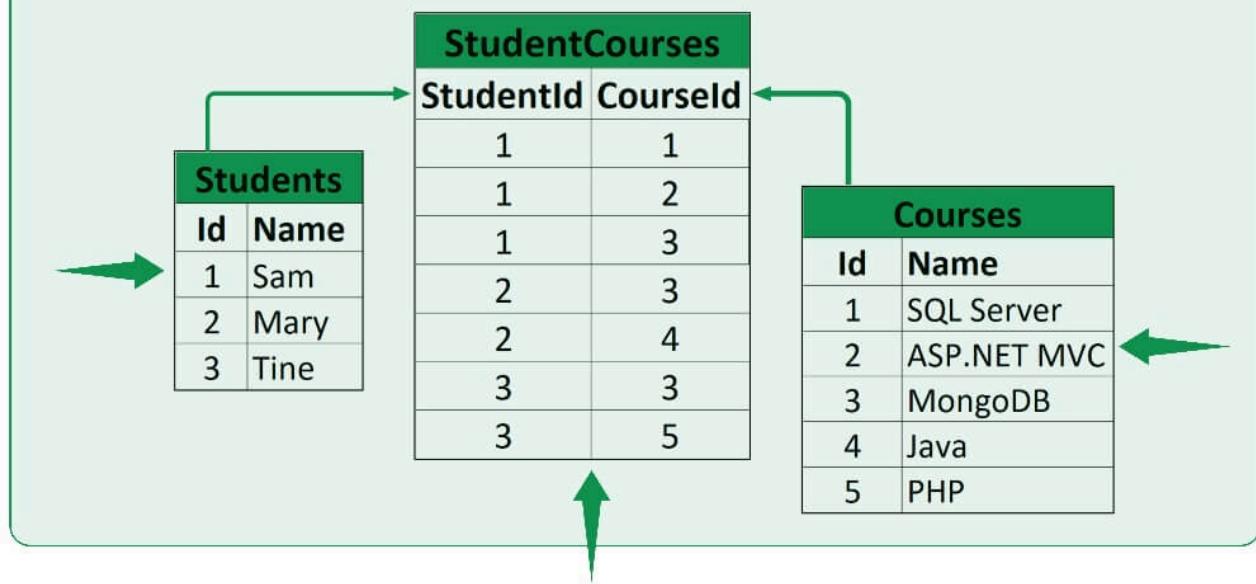


Fig. 10. Example of relational database structure

The relational model separates logical data structures (like tables and views) from physical storage structures. This separation permits flexibility in managing physical data without disrupting the logical data access. Database operations are divided into logical operations (defining what data is needed) and physical operations (how to access and process this data).

SQL (Structured Query Language) is the standard language for relational databases. It allows for data querying, updating, and management using commands like SELECT, INSERT, and DELETE. SQL is essential for tasks from web development to data analysis due to its versatility and wide adoption.

## 2.1.5 Knowledge base retrieval and reranking

A knowledge base retrieval and reranking system that integrates semantic search with document similarity scoring to enhance the accuracy of information retrieval. The process begins by converting a user query into an embedding using the sentence-transformers library, which enables semantic understanding beyond keyword matching. The system then performs a similarity search against a vectorized knowledge base using FAISS, retrieving the top relevant documents. To refine the relevance of these initial results, a reranking step is applied using a BGE-M3, which evaluates the query-document pairs for deeper contextual relevance. This two-stage pipeline:semantic retrieval followed by reranking ensures that the final ranked list of documents closely matches the user's intent, which makes it suitable for tasks such as FAQ answering, customer support, and intelligent search in knowledge-based systems.

## 2.1.6 Retrieval-Augmented Generation (RAG).

Retrieval-Augmented Generation (RAG) combines information retrieval with generative modeling to enhance NLP applications. While RAG's generative component synthesizes text, the retrieval layer independently serves as a mechanism for dynamically fetching relevant information from an external knowledge base (KB).

The retrieval layer operates by embedding both the input query and the KB entries into a shared vector space using dense embeddings (e.g., BERT-based models). Cosine similarity or dot product is then used to identify the most relevant KB entries based on semantic closeness. [9]

In AURA, only the retrieval layer of RAG is utilized. After a call transcript is summarized, the retrieval mechanism identifies relevant KB entries for downstream classification. For example, a query about insurance reimbursement would retrieve KB articles related to reimbursement processes, ensuring that the classification models (BERT-based) operate on contextually relevant information. This approach enhances the precision of classification and minimizes irrelevant predictions.

By using the RAG retrieval layer, the system balances efficiency and accuracy, leveraging pre-existing knowledge bases to support real-time classification in NHSO's call center operations.

#### *2.1.7 LoRA (Low-Rank Adaptation of Large Language Models)*

Low-Rank Adaptation of Large Language Models (LoRA) is a fine-tuning technique designed to efficiently adapt large pre-trained language models (LLMs) to downstream tasks. Instead of updating all model parameters, which is computationally expensive and memory-intensive, LoRA freezes the original model's weights and introduces small trainable low-rank matrices into specific layers, such as the attention layers. These low-rank matrices capture task-specific adaptations by operating within a low-dimensional subspace, significantly reducing the number of parameters that need to be fine-tuned. During training, LoRA computes the original model's output and adds the learned updates from the low-rank matrices, allowing the model to retain its pre-trained knowledge while adapting efficiently to new tasks. This method reduces computational costs and memory usage, making it particularly beneficial for fine-tuning large models with billions of parameters on limited hardware. LoRA has been applied in various tasks, such as sentiment analysis, text summarization, and domain-specific adaptations, and is a practical solution for deploying LLMs in resource-constrained environments while maintaining competitive performance. [10]

#### *2.1.8 Supervised Learning*

Supervised learning is a core paradigm in machine learning where an algorithm is trained on a labeled dataset, meaning each input is paired with a correct output. The model iteratively adjusts its internal parameters to minimize a loss function that quantifies the difference between its predictions and the ground truth labels. Training typically involves gradient-based optimization and validation against a held-out dataset to prevent overfitting. Supervised learning is divided into classification tasks—where outputs are discrete categories—and regression tasks—where outputs are continuous values. It is widely used in applications such as spam detection, fraud classification, medical diagnosis, and speech recognition. The effectiveness of supervised learning heavily depends on the quality and representativeness of the labeled data provided.

#### *2.1.9 Data Synthesis*

Data synthesis is the process of generating artificial data that resembles real-world datasets in structure, semantics, and statistical properties. Synthetic data can be produced using rule-based generation, simulation environments, or generative models like Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), and large language models (LLMs). Data synthesis is especially useful when real data is scarce, expensive to collect, or subject to privacy constraints. It allows for augmentation of training sets, balancing of class distributions, and simulation of edge cases or rare events. However, ensuring the realism and diversity of synthesized data is critical to avoid introducing biases or distributional mismatches during downstream learning.

#### *2.1.10 Multimodal Models*

Multimodal models are a class of machine learning models designed to process and integrate multiple data modalities—such as text, audio, images, or video—into a unified representation. These

models leverage architectures like multimodal Transformers or dual encoders, which learn to align and correlate information from different sources. Attention mechanisms are typically used to cross-reference between modalities, enabling joint understanding. Multimodal models are foundational to tasks such as visual question answering, video captioning, audio-visual event detection, and medical diagnosis from multimodal scans. Their ability to understand context from multiple input types allows for richer interpretation and more robust decision-making compared to unimodal models.

#### *2.1.11 Vision-Language Models (VLMs)*

Vision-language models are neural architectures that combine computer vision and natural language processing to jointly interpret image and textual data. These models are typically trained on aligned image-text pairs using objectives such as contrastive learning, masked language modeling, or image captioning. Transformer-based models like CLIP, BLIP, and Flamingo map both modalities into a shared embedding space, enabling cross-modal tasks such as image retrieval, caption generation, and visual question answering. VLMs can be extended with instruction tuning or chain-of-thought prompting to handle tasks requiring reasoning grounded in both visual and linguistic cues. Their capacity to reason about visual context makes them particularly powerful in domains like digital accessibility, content moderation, and document understanding.

#### *2.1.12 Speech-to-Text Models*

Speech-to-text models, or automatic speech recognition (ASR) systems, convert spoken language in audio signals into written text. These models typically involve a frontend that extracts acoustic features (e.g., Mel spectrograms) and a neural backend—often based on RNNs, CNNs, or Transformers—that decodes the features into textual tokens. Modern ASR models like Whisper and Conformer-Transducer utilize attention mechanisms and large-scale pretraining to improve robustness across accents, noise levels, and speaking styles. They are used in applications such as voice assistants, subtitle generation, accessibility tools, and transcription of legal or medical records. Key challenges include handling domain-specific vocabulary, speaker variation, and background noise.

#### *2.1.13 Speaker Diarization Models*

Speaker diarization is the task of segmenting an audio stream into distinct speaker turns and assigning a speaker label to each segment—effectively answering "who spoke when?" Diarization systems generally consist of voice activity detection (VAD), speaker embedding extraction (e.g., x-vectors or ECAPAs), and clustering algorithms such as agglomerative hierarchical clustering (AHC) or spectral clustering. Some end-to-end diarization models integrate these components into a single neural network. Accurate diarization is critical in applications like meeting transcription, conversational analysis, and call center monitoring. The main challenges involve overlapping speech, rapid speaker turns, and short utterances, which can lead to poor separation and label switching.

#### *2.1.14 Agentic LLMs*

Agentic large language models represent a shift from passive text generators to autonomous reasoning agents capable of goal-oriented behavior. These models combine LLMs with architectural patterns that support decision-making, self-reflection, tool use, and memory persistence. Agentic capabilities are often implemented using multi-turn prompting (e.g., ReAct or Chain-of-Thought), external memory modules, planning heuristics, or frameworks like AutoGPT and LangGraph. Agentic LLMs can decompose tasks into subtasks, query APIs, revise outputs, and adapt their behavior over time. This enables advanced use cases such as autonomous research agents, multi-step workflow execution, and long-context document synthesis. Their design aims to emulate higher-level cognition by blending language understanding with iterative reasoning and structured action.

### *2.3.15 Asynchronous Processing*

Asynchronous processing is a programming paradigm that allows tasks to be executed in a non-blocking manner, enabling a system to perform multiple operations concurrently without waiting for each task to finish before starting the next. In contrast to synchronous (blocking) execution, asynchronous processing improves the responsiveness, throughput, and efficiency of applications—especially in I/O-bound or network-intensive workflows.

In backend systems, asynchronous processing is commonly implemented using event loops, callbacks, futures, or `async/await` syntax (e.g., in Python’s `asyncio` or JavaScript’s `Promise` constructs). Frameworks such as FastAPI and Node.js are inherently designed to support `async` operations, allowing developers to build REST APIs and services that handle multiple requests simultaneously without spawning new threads for each client.

Use cases include:

- Handling simultaneous API requests in web servers
- Streaming real-time data (e.g., audio transcription or token-by-token LLM responses)
- Coordinating communication between microservices (e.g., model inference pipelines)
- Managing long-running background tasks (e.g., document preprocessing or embedding generation)

By leveraging asynchronous patterns, systems can avoid bottlenecks, reduce memory usage, and scale more effectively under high loads, which is critical for modern AI applications involving multi-stage inference, storage, and retrieval processes.

### *2.3.16 Bi-Encoder Architecture*

A bi-encoder is a dual-encoder neural architecture used primarily in semantic similarity and retrieval tasks. In this setup, a query and a document are independently encoded into dense vector representations using the same or similar encoder models. This architecture enables efficient approximate nearest-neighbor (ANN) search using vector similarity (e.g., cosine or L2 distance) and is commonly used in Siamese retrieval systems. In the AURA pipeline, WangchanBERT, PhayathaiBERT, and BGE-M3 are deployed in bi-encoder mode to embed queries and KB chunks into the same vector space for real-time semantic retrieval using FAISS or Milvus.

### *2.1.17 Cross-Encoder Architecture*

A cross-encoder architecture jointly encodes a pair of inputs—typically a query and a candidate document—using a single transformer model. Unlike bi-encoders, which process inputs separately, cross-encoders concatenate both inputs and allow full attention between them, enabling richer interaction patterns. This architecture is especially effective for reranking top-k candidates retrieved by a fast bi-encoder or BM25 stage. In AURA, the BGE-M3 reranker acts as a cross-encoder to score query-chunk pairs, optimizing retrieval precision at the cost of latency.

### *2.1.18 Contrastive and Triplet Loss*

Contrastive and triplet losses are distance-based learning objectives used to fine-tune embedding models like Siamese bi-encoders. Contrastive loss pushes apart dissimilar examples and pulls similar examples closer in embedding space, based on binary similarity labels. Triplet loss uses

anchor-positive-negative triplets and enforces that the anchor is closer to the positive than the negative by a given margin. These objectives are essential for improving fine-grained semantic discrimination, especially in dense retrieval systems. In AURA, triplet loss was used with hard negative sampling to fine-tune WangchanBERT and BGE-M3 embeddings.

### 2.1.19 Hard Negative Mining

Hard negative mining is a training technique where the most challenging negative examples—those that are semantically close but incorrect—are selectively included during model fine-tuning. This exposes the model to borderline cases that it would otherwise confuse with true positives. In both reranker and bi-encoder fine-tuning for AURA, top-k retrieved but irrelevant KB chunks were used as hard negatives to improve boundary decision-making, increasing recall and robustness in downstream tasks.

### 2.1.20 BM25 Ranking Function

BM25 (Best Matching 25) is a probabilistic ranking function widely used for lexical information retrieval. It ranks documents based on the presence of query terms, document length normalization, and term frequency, using the formula:

$$\text{BM25}(q, d) = \sum_{i=1}^{|q|} \text{IDF}(q_i) \cdot \frac{f(q_i, d) \cdot (k_1 + 1)}{f(q_i, d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{\text{avgdl}})}$$

where  $f(q_i, d)$  is the term frequency,  $|d|$  is the document length, and  $k_1, b$  are tunable hyperparameters. BM25 is used in Elasticsearch as the core retrieval method before applying neural rerankers in AURA’s hybrid pipeline.

### 2.1.21 ColBERT (Contextualized Late Interaction over BERT)

ColBERT is a retrieval architecture that generates multiple contextualized embeddings per query and document token, enabling fine-grained similarity matching via late interaction. Instead of pooling token embeddings into a single vector (as in bi-encoders), ColBERT computes token-wise similarity scores and aggregates them via max-pooling. This preserves token-level context and enhances retrieval accuracy, especially in long or information-dense documents. In AURA, BGE-M3 supports ColBERT-style retrieval via its multi-vector mode, enabling robust hybrid scoring when fused with BM25 and dense embeddings.

## 2.2 Evaluation Metrics

Text Summarization:

### 2.2.1 Rouge Score

The ROUGE Score (Recall-Oriented Understudy for Gisting Evaluation) is a widely used evaluation metric in natural language processing tasks such as summarization and machine translation. It measures the overlap between the generated text (candidate) and the reference text (ground truth) by comparing n-grams, word sequences, and the longest common subsequences. ROUGE has several variants, including ROUGE-N, which focuses on n-gram overlap (e.g., ROUGE-1 for unigrams and ROUGE-2 for bigrams), and ROUGE-L, which evaluates the longest common subsequence to capture sentence-level structure similarity. It calculates recall, precision, and F1 score, with recall being particularly important for capturing all critical content from the reference text. ROUGE is language-independent and simple to compute, making it a versatile and widely accepted metric. However, it is limited by its reliance on surface-level matching, which may overlook semantic

similarities when paraphrasing occurs, and its inability to account for deeper contextual understanding. Despite these limitations, ROUGE remains a practical and interpretable benchmark for evaluating text generation quality, especially in tasks like summarization. [11]

### 2.2.2 Loss Function

A loss function is a mathematical formula used in machine learning to quantify the difference between the predicted output of a model and the actual target values. It serves as a critical component in training models by providing feedback on how well the model is performing on a given task. During training, the model optimizes its parameters by minimizing the loss function, typically using algorithms like gradient descent. Loss functions can be categorized into different types depending on the task: for regression problems, common loss functions include Mean Squared Error (MSE) or Mean Absolute Error (MAE), while classification tasks often use Cross-Entropy Loss or Hinge Loss. The value of the loss function represents the error, with lower values indicating better model performance. A well-chosen loss function is essential for guiding the model to learn the desired patterns and achieve high accuracy on unseen data. It not only evaluates the model's predictions during training but also influences how efficiently the model converges to an optimal solution. [12]

## Topic Classification

### 2.2.3 Precision

Precision measures the proportion of correctly predicted positive instances out of all instances predicted as positive. It focuses on the relevance of predictions and is calculated as:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

High precision ensures that irrelevant results are minimized, making it a critical metric when false positives are costly.

### 2.2.4 Recall

Recall evaluates the proportion of correctly predicted positive instances out of all actual positive instances. It is particularly important when it is critical to identify all relevant cases. Recall is given by:

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

A high recall ensures that most relevant instances are identified, which is vital in scenarios like medical classification.

### 2.2.5 F1 Score

The F1 Score provides a balanced measure between precision and recall, making it particularly useful when dealing with imbalanced datasets. It is the harmonic mean of precision and recall:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

A high F1 score indicates a good balance between precision and recall, crucial for tasks where both metrics are equally important.

#### 2.2.6 Accuracy

Accuracy measures the proportion of correct predictions out of all predictions made. It is calculated as:

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}}$$

While accuracy provides an overall performance metric, it must be interpreted carefully in imbalanced datasets, as it may not fully reflect the performance of minority class predictions. [15]

#### 2.2.7 Word Error Rate (WER)

WER is a critical metric for evaluating the accuracy of speech-to-text systems. It calculates the error rate based on insertions (I), deletions (D), and substitutions (S) in the transcribed text compared to the reference text. WER is defined as:

$$\text{WER} = \frac{\text{Insertions (I)} + \text{Deletions (D)} + \text{Substitutions (S)}}{\text{Total Words (N)}}$$

A lower WER indicates more accurate transcription, directly impacting the quality of downstream tasks such as summarization and classification. [16]

#### 2.2.8 Diarization Error Rate (DER)

DER measures how well the speaker diarization system separates and labels different speakers in an audio recording. It is based on three types of mistakes: missed speech (when speech is not detected), false alarms (when silence or noise is detected as speech), and speaker confusion (when the wrong speaker label is assigned).

$$\text{DER} = \frac{\text{False Alarms} + \text{Missed Detections} + \text{Speaker Confusions}}{\text{Ground Truth Duration}}$$

A lower DER means the system is better at telling who is speaking and when.

#### 2.2.9 Mean Reciprocal Rank (MRR)

The Mean Reciprocal Rank (MRR) is an evaluation metric commonly used in information retrieval systems to measure the effectiveness of ranked outputs. It calculates the average of the reciprocal ranks of the first relevant item in the result list for a set of queries. If the correct knowledge base (KB) appears at the top of the list, it contributes more to the final score. MRR is defined as:

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}$$

where  $rank_i$  is the rank position of the first relevant result for the  $i$ th query. MRR values range from 0 to 1, with higher values indicating better performance. It is especially useful when only the top results are expected to be relevant, making it ideal for evaluating KB retrieval quality in real-time pipelines.

#### 2.2.10 Token Intersection-over-Union (Token-IoU)

Token-IoU is a semantic evaluation metric that measures the overlap between retrieved document chunks and ground-truth chunks at the token level. It is particularly relevant for tasks involving retrieval of semantically similar segments from long documents. The IoU is calculated by dividing the number of overlapping tokens by the number of tokens in the union of the predicted and reference spans:

$$\text{Token-IoU} = \frac{|\text{tokens}_{\text{pred}} \cap \text{tokens}_{\text{true}}|}{|\text{tokens}_{\text{pred}} \cup \text{tokens}_{\text{true}}|}$$

This metric captures fine-grained semantic recall and is used to benchmark the coverage quality of chunk-based retrieval systems like NHSO's hybrid KB retriever. In the AURA system, a Token-IoU score of at least 0.80 is targeted for high-quality retrieval.

#### 2.2.11 Latency (P95)

Latency at the 95th percentile (P95) measures the maximum time it takes to return retrieval results for 95% of requests, giving a realistic upper bound on user wait time under typical system load. It is critical in real-time systems where response speed directly affects user experience. For the AURA system, the latency is measured from the end of a call to the point when KB results are returned.

P95 latency is calculated by sorting all measured response times and selecting the value below which 95% of them fall. This metric reflects performance under peak load while ignoring extreme outliers, making it a robust indicator for production readiness. The NHSO pipeline targets a maximum P95 latency of 300 milliseconds for retrieval and reranking on GPU-backed infrastructure.

### 2.3 Technologies

#### 2.3.1 Whisper

Whisper is an advanced Speech-To-Text model developed by OpenAI. It is designed to accurately transcribe speech in multiple languages. Whisper is known for its ability to handle different accents and noisy environments, making it a great choice for challenging transcription tasks like those in call centers. It works by analyzing audio input and breaking it down into smaller phonetic units. These units are known as "phonemes," which represent the smallest sounds in speech. The model then uses patterns in these phonemes to match them to corresponding words in its trained vocabulary. Speech-to-text models like Whisper, rely on large datasets containing phoneme-to-word mappings, helping them understand how speech sounds correspond to written language. They use deep learning algorithms to continuously improve their accuracy by learning from a variety of speech patterns, accents, and background noise. [17]

#### 2.3.2 pyannote-audio

pyannote-audio is a Python library for speaker diarization, designed to process audio recordings and identify different speakers in a conversation. It uses deep learning models to segment and label audio

into speaker-specific chunks. This technology is useful in tasks like transcribing multi-speaker audio, where it's important to distinguish between speakers. pyannote-audio excels in handling noisy or overlapping speech, making it ideal for applications such as call center transcription or meeting minutes. It can be integrated with other speech-to-text systems like Whisper to improve transcription accuracy in multi-speaker scenarios. [18]

#### 2.3.3 DeepCut

DeepCut is a tokenization tool designed specifically for the Thai language, which handles the challenge of word segmentation in Thai text. Unlike many languages with spaces between words, Thai text often lacks clear delimiters, making tokenization a critical task for processing speech-to-text transcriptions. DeepCut utilizes machine learning techniques to accurately segment a sequence of Thai characters into meaningful words, ensuring that the transcribed text is properly structured for further processing, such as error rate calculation. [19]

#### 2.3.4 JiWER

JiWER is a widely used tool for evaluating the accuracy of speech-to-text models. It compares the transcribed text with a reference, measuring three key aspects: insertions, deletions, and substitutions. By calculating the Word Error Rate (WER), JiWER provides an objective metric that reflects the quality of transcription. Lower WER values indicate more accurate models. This tool is essential in assessing the performance of speech recognition systems, especially when dealing with different accents, noisy environments, and complex speech patterns. [20]

#### 2.3.5 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained transformer-based model designed for NLP tasks such as text classification, question answering, and sentiment analysis. It processes text bidirectionally, understanding context by analyzing both preceding and succeeding words simultaneously. BERT has transformed NLP by enabling fine-tuning on task-specific datasets, which significantly reduces the need for large task-specific training datasets. Its ability to generalize across various domains makes it a foundational technology for text classification .[22]

#### 2.3.6 ThaiBERT

ThaiBERT is a BERT-based model pre-trained specifically for Thai language processing. Given our project's focus on Thai data, ThaiBERT ensures that NLP tasks are more accurate by capturing the unique linguistic features of the Thai language, such as word segmentation and tonal markers. ThaiBERT has been widely used for classification and sentiment analysis tasks in Thai, making it a critical tool for improving AURA's system performance on domain-specific tasks.

#### 2.3.7 WangchanBERTa

WangchanBERTa, a BERT-based model developed specifically for the Thai language, enhances Thai text processing by leveraging large-scale pretraining on diverse Thai datasets. It excels in tasks like topic modeling, classification, and summarization, aligning well with our project's requirements. WangchanBERTa's efficiency and accuracy in handling Thai text make it ideal for your classification and retrieval-focused tasks. [23]

#### 2.3.8 Llama 3

LLaMA 3 (Large Language Model Meta AI, version 3) is an advanced Transformer-based large language model designed for natural language processing tasks such as text generation, summarization, and classification. It builds on previous iterations with enhanced capabilities and

optimizations, including support for multiple languages, making it suitable for diverse domains. LLaMA 3 emphasizes parameter efficiency by offering models in varying sizes, such as 1 billion or 3 billion parameters, allowing adaptability for resource-constrained environments without compromising performance. Its flexibility enables fine-tuning on domain-specific datasets, producing contextually relevant and coherent outputs. Additionally, LLaMA 3 focuses on reducing inference latency, making it practical for real-time applications. With its ability to process long contexts effectively and generate human-like text, LLaMA 3 is a powerful and versatile tool for modern NLP tasks. [24]

### *2.3.9 Node JS*

Node.js is an open-source, cross-platform javascript runtime environment. [nodejs] It is designed for scalable network applications and runs the V8 JavaScript engine, allowing for high performance. The use of Node JS enables developers to efficiently build back-end, front-end, and full-stack applications with the JavaScript language. Node.js was created by Ryan Dahl in 2009 and is currently developed by the OpenJS Foundation. [25]

### *2.3.10 Express js*

Express.js is a Node.js web framework for web and mobile applications. [express js] It simplifies the development of web applications and APIs by providing a number of HTTP utility methods and middleware, making it easy and efficient to create robust APIs and web services.

Express.js, a project of the OpenJS Foundation, is open-source and licensed under the Creative Commons Attribution-ShareAlike 3.0 United States License. It is widely used due to its performance advantages and fundamental web application features, without obscuring the features [26]

### *2.3.11 MySQL*

MySQL is an open-source relational database management system (RDBMS) that is highly recognized for its efficiency in storing, organizing, and retrieving data.[mysql]Developed by MySQL AB and supported by Oracle Corporation. As a relational database, it structures data in tables, thereby enabling the efficient management of complex data relationships and queries.

Created in 1995, MySQL has an open-source licensing under the GPL (GNU General Public License), which allows users to freely download, use, and modify the software. This aspect of MySQL makes it accessible for a wide range of applications, from small individual projects to large-scale corporate uses. For those requiring different licensing terms, especially in commercial applications, MySQL also offers a commercially licensed version.

In terms of capabilities, MySQL is designed to be scalable and flexible. It operates efficiently across various platforms, whether it is running on a simple desktop or on high-powered servers, and can even be scaled up to work in clustered environments for more demanding tasks. MySQL boasts a comprehensive set of features, including multiple storage engines, robust support for transactions, and extensive application programming interfaces (APIs) for various programming languages. Its versatility is further showcased in its ability to function in a client/server architecture or as an embedded database, making it a versatile choice for different software development scenarios. [27]

### *2.3.12 Accelerate*

Accelerate is a Python library developed by Hugging Face that simplifies the process of training and fine-tuning machine learning models on distributed systems, including multi-GPU and multi-node setups. It provides a seamless interface for leveraging multiple hardware resources, such as GPUs or

TPUs, without requiring extensive modifications to the original code. Accelerate handles the complexities of distributed training by managing data parallelism, gradient synchronization, and hardware communication efficiently, allowing researchers to focus on model development and experimentation. It is framework-agnostic, meaning it works with PyTorch and other deep learning libraries, making it versatile for various use cases. Additionally, Accelerate includes features like mixed precision training, which reduces memory usage and speeds up training by using lower-precision computations where applicable. With its user-friendly API, Accelerate empowers developers to train large-scale models efficiently, maximizing hardware utilization while minimizing engineering overhead. [28]

### *2.3.13 Nvidia Triton Inference Server*

NVIDIA Triton Inference Server is an open-source platform designed to streamline the deployment, serving, and scaling of machine learning models in production environments. It supports multiple machine learning frameworks, including TensorFlow, PyTorch, ONNX, and XGBoost, enabling flexibility in deploying models from various ecosystems. Triton provides advanced features such as dynamic batching, model versioning, and multi-model deployment on CPUs and GPUs, optimizing resource utilization and inference performance. It is highly scalable, making it suitable for real-time applications and high-throughput environments. Triton also supports HTTP/gRPC endpoints, allowing seamless integration with applications and RESTful APIs. Additionally, it includes tools for model monitoring and metrics collection, which help maintain performance and reliability. With its ability to efficiently serve models on diverse hardware configurations, NVIDIA Triton Inference Server is a powerful solution for deploying AI models in enterprise and cloud environments. [29]

### *2.3.14 PyTorch*

PyTorch is an open-source deep learning framework developed by Facebook's AI Research lab (FAIR). It is designed to provide flexibility and speed for building and deploying machine learning models, especially those based on neural networks. PyTorch is widely recognized for its dynamic computational graph, which allows developers to modify the graph on the fly during runtime, making it particularly well-suited for research and experimentation.

Key features of PyTorch include:

- Dynamic Computational Graphs:

Unlike static computation graphs used in some frameworks, PyTorch allows for more intuitive model building and debugging by enabling operations to be defined dynamically.

- Tensor Operations:

PyTorch provides a comprehensive suite of tensor operations, including matrix manipulation and mathematical functions, optimized for use on both CPUs and GPUs.

- Autograd:

The framework's automatic differentiation engine computes gradients for tensor operations, which simplifies the process of backpropagation during neural network training.

- Support for GPU Acceleration:

PyTorch integrates seamlessly with CUDA, enabling efficient computation on GPUs to accelerate model training and inference.

- Integration with Machine Learning Libraries:

PyTorch works well with libraries like Hugging Face Transformers, torchvision (for computer vision tasks), and torchaudio (for audio processing).

PyTorch is highly versatile, catering to applications such as natural language processing, computer vision, and reinforcement learning. Its flexibility, coupled with robust performance, has made it a popular choice among researchers and developers for creating state-of-the-art machine learning and deep learning models. [30]

### 2.3.15 Hugging Face Transformers

Hugging Face Transformers is an open-source library that provides a comprehensive framework for working with state-of-the-art natural language processing (NLP) models based on Transformer architectures. It offers pre-trained models for a wide range of tasks, including text generation, classification, summarization, translation, and question answering, supporting architectures like BERT, GPT, T5, and PEGASUS. The library is highly flexible and integrates seamlessly with PyTorch and TensorFlow, allowing developers to fine-tune models on custom datasets or use them directly for inference. Hugging Face Transformers simplifies complex tasks with its user-friendly API, enabling quick access to powerful models and tools. It also supports distributed training, mixed precision, and efficient tokenization, making it suitable for both research and production. With a vast repository of pre-trained models and extensive documentation, Hugging Face Transformers has become a go-to resource for building and deploying NLP solutions. [31]

### 2.3.16 Kubernetes

Kubernetes is an open-source platform for automating the deployment, scaling, and management of containerized applications. It organizes containers into pods, providing features like automatic scaling, load balancing, self-healing, and rolling updates for seamless application management. Kubernetes abstracts infrastructure complexities, ensuring applications run reliably across on-premises, cloud, or hybrid environments. With declarative configurations, it maintains the desired state of applications while supporting advanced features like persistent storage, service discovery, and networking, making it a versatile solution for scalable and resilient deployments. [32]

### 2.3.17 BGE-M3

BAAI/bge-m3 is a cutting-edge multilingual embedding model developed by the Beijing Academy of Artificial Intelligence (BAAI) that represents a significant advancement in cross-lingual representation learning. Building on the success of previous BGE (BAAI General Embeddings) models, bge-m3 demonstrates exceptional performance across multiple languages, supporting over 100 languages with particularly strong results in high-resource languages like English, Chinese, and major European languages. The model employs a sophisticated transformer architecture optimized for semantic similarity tasks, which enables it to capture nuanced relationships between texts regardless of their source language. Its versatility makes it valuable for multilingual information retrieval, cross-lingual document matching, semantic search applications, and various natural language understanding tasks. BAAI/bge-m3's ability to generate consistent embedding spaces across languages has made it a preferred choice for organizations handling multilingual content, which allows for unified vector representations that bridge linguistic barriers and facilitate more effective cross-lingual knowledge transfer[BAAI/bge-m3 · Hugging Face,” [huggingface.co](https://huggingface.co/BAAI/bge-m3). <https://huggingface.co/BAAI/bge-m3>].

### 2.3.18 FAISS

FAISS (Facebook AI Similarity Search) is a prominent open-source library developed by Facebook AI Research that enables efficient similarity search and clustering of dense vectors. When implementing vector search with FAISS using L2 distance and the IndexFlatL2 index type, developers gain access to a powerful exact search mechanism optimized for Euclidean distance

computations. IndexFlatL2 provides the foundation for precise nearest neighbor search operations when accuracy is paramount.

The L2 distance (Euclidean distance) measures the straight-line distance between two points in vector space, which makes it particularly suitable for applications where spatial relationships between embeddings directly correlate with semantic similarity. Unlike approximate indices that trade accuracy for speed, IndexFlatL2 performs exhaustive searches, computing the exact L2 distance between the query vector and every vector in the database. This guarantees optimal search results without compromising accuracy, though at the cost of increased computational resources as the dataset grows.

Implementation of FAISS with IndexFlatL2 follows a straightforward workflow: first, an IndexFlatL2 instance is initialized with the dimensionality of the embedding vectors; next, vectors are added to the index; finally, search operations return the k-nearest neighbors based on L2 distance calculations. While IndexFlatL2 does not employ sophisticated data structures for acceleration, it benefits from FAISS's highly optimized matrix operations and parallel processing capabilities, including GPU acceleration when available, which makes it substantially faster than naive implementations.

IndexFlatL2 serves as an excellent baseline for search quality evaluation and is particularly suitable for applications with modest dataset sizes (up to millions of vectors) where search precision cannot be compromised. For larger-scale deployments, it often serves as a building block for more complex FAISS indices that incorporate quantization or hierarchical structures to improve efficiency while maintaining acceptable accuracy levels. [“Faiss: A library for efficient similarity search,” *Facebook Engineering*, Mar. 29, 2017.

<https://engineering.fb.com/2017/03/29/data-infrastructure/faiss-a-library-for-efficient-similarity-search/>]

### 2.3.19 Qwen2.5

Qwen2.5 is a family of multilingual language and vision-language models developed by Alibaba DAMO Academy, designed to support a broad range of tasks in natural language understanding (NLU), generation, and multimodal reasoning. The Qwen2.5 architecture is based on the Transformer model and has been released in various sizes, including base and instruct-tuned variants, with capabilities extending from pure text understanding to vision-language processing in multimodal settings. The “Instruct” versions are fine-tuned using alignment techniques to follow human instructions more reliably.

In the language-only domain, Qwen2.5 supports high-quality instruction following, multilingual comprehension, summarization, and classification. For multimodal tasks, Qwen2.5-VL (Vision-Language) integrates image and text processing through a vision encoder (typically a ViT variant) and cross-attention mechanisms. This allows it to handle tasks like image captioning, document OCR explanation, and question answering grounded in visual context. Qwen2.5 models also support extended context lengths, making them suitable for scenarios requiring long-document comprehension or multiple image-text interactions.

### 2.3.20 VLLM

VLLM (Very Large Language Model Serving) is an open-source inference framework developed to optimize serving of large Transformer-based language models, particularly for deployment in latency-sensitive applications. It is designed to support OpenAI-compatible APIs and leverages continuous batching and paged attention to enable efficient handling of multiple inference requests in parallel.

VLLM introduces “paged attention,” a memory-efficient mechanism that reduces redundant computation by reusing cached key-value pairs across decoding steps. It also supports asynchronous

request handling, enabling models to process streaming or multi-user tasks concurrently. These features make VLLM highly performant for applications involving high-throughput summarization, chat, and text generation. VLLM supports various backends (CUDA, ROCm) and can be deployed with models from Hugging Face Transformers, including those converted with vLLM's own loader utilities.

### 2.3.21 Docker

Docker is a platform that enables developers to package applications and their dependencies into standardized containers. A Docker container is a lightweight, standalone executable unit that includes everything needed to run an application—code, runtime, libraries, and system tools—isolated from the host environment.

Docker containers run using a shared kernel but maintain their own isolated process space, making them ideal for reproducible environments, CI/CD pipelines, and cross-platform deployment. Docker images are built using a Dockerfile that defines the application setup, and images can be versioned and distributed via container registries like Docker Hub. This allows for rapid testing and deployment of software in consistent environments, minimizing "it works on my machine" issues.

### 2.3.22 Docker Compose

Docker Compose is a tool for defining and managing multi-container Docker applications. It allows developers to configure services, networks, and volumes using a `docker-compose.yml` file. Each service is described as a container that can depend on others—for example, a web server that depends on a database and a cache service.

Compose streamlines orchestration by providing simple commands to start, stop, and rebuild entire container ecosystems. This is especially useful in development, where complex microservice architectures can be brought up with a single command. Compose also enables easy scaling and configuration across environments (development, staging, production) through variable substitution and environment files.

### 2.3.23 FastAPI

FastAPI is a modern, high-performance web framework for building APIs with Python 3.7+ based on standard Python type hints. Built on top of Starlette for the web handling and Pydantic for data validation, FastAPI delivers both speed and developer ergonomics.

It supports automatic generation of OpenAPI (Swagger) documentation, dependency injection, and asynchronous request handling with full support for `async/await` syntax. FastAPI is ideal for building ML-powered APIs because it allows easy integration with background workers, WebSocket endpoints, and model inference pipelines. It also provides strong validation, making it suitable for handling structured JSON input and output in microservices.

### 2.3.24 PostgreSQL

PostgreSQL is a powerful, open-source relational database management system known for its reliability, feature richness, and extensibility. It supports advanced data types (e.g., arrays, JSONB, HSTORE), full ACID compliance, and complex queries involving joins, subqueries, and window functions.

PostgreSQL includes support for procedural languages like PL/pgSQL, indexing strategies like GIN and GiST, and extensions such as PostGIS for geospatial data and pgvector for vector similarity search. It is widely used in enterprise systems, web applications, and analytics platforms where consistency, concurrency, and scalability are required.

### *2.3.25 Elasticsearch*

Elasticsearch is a distributed, RESTful search and analytics engine built on top of Apache Lucene. It is designed for horizontal scalability, real-time performance, and full-text search on structured or unstructured data.

Elasticsearch organizes data into indices composed of documents and fields. It uses inverted indices and BM25 (or TF-IDF) ranking to efficiently retrieve relevant documents. Common use cases include log aggregation (e.g., ELK Stack), site search, and metadata indexing. It also supports fuzzy search, filtering, and faceted queries, and is often combined with Kibana for visualization.

### *2.3.26 LLaMA Factory*

LLaMA Factory is an open-source training and inference framework for fine-tuning and deploying LLaMA (Large Language Model Meta AI) models. It supports instruction tuning using Supervised Fine-Tuning (SFT), Reinforcement Learning from Human Feedback (RLHF), and other methods.

The framework is built on PyTorch and integrates with Hugging Face Transformers. It supports LoRA (Low-Rank Adaptation), PEFT (Parameter-Efficient Fine-Tuning), and DeepSpeed to enable fine-tuning large models on modest GPU resources. It also includes configuration templates, checkpoint resumption, and metrics logging, making it suitable for research and production fine-tuning of LLaMA and related models.

### *2.3.27 Gemini 2.0*

Gemini 2.0 is a proprietary multimodal large model developed by Google DeepMind, designed for high-fidelity reasoning across text and image modalities. It is part of Google's Gemini model family and is accessible via the Google Vertex AI API platform.

Gemini 2.0 supports complex tasks including OCR-based document understanding, instruction-following image captioning, and visual QA. It is fine-tuned with a mixture of supervised learning and reinforcement learning from human feedback (RLHF). In addition to text generation, Gemini can interpret image layouts, extract tabular data, and provide Thai-language outputs, making it practical for multilingual and multimodal document processing. The model supports input limits exceeding 32k tokens and features throttled rate limits for API-based deployments.

### *2.3.28 Gemma 3*

Gemma 3 is an open-source family of Transformer-based language models released by Google as part of their open LLM initiative. Inspired by Gemini, Gemma models are optimized for multilingual and instruction-following tasks and are fine-tuned using human preference data.

Gemma 3 models are available in various sizes (e.g., 2B, 7B, 27B) and support quantization, LoRA-based fine-tuning, and efficient inference with Hugging Face Transformers and vLLM. They provide high performance on reasoning, summarization, and conversational tasks. Unlike Gemini, which is API-based, Gemma models can be fully self-hosted and customized, making them suitable for privacy-preserving deployments.

### *2.3.29 FlagEmbeddin*

FlagEmbedding is an efficient embedding library optimized for sentence and document-level semantic encoding. It provides APIs and pretrained models for multilingual and monolingual semantic search and ranking tasks.

Built upon optimized Transformer backbones like BGE and MiniLM, FlagEmbedding supports dense vector generation suitable for similarity search, clustering, and reranking. It includes GPU-accelerated batching and quantization options for real-time applications. The library is often

used with FAISS or Elasticsearch for hybrid retrieval pipelines and supports loading models from Hugging Face Transformers with minimal overhead.

#### *2.3.30 OpenAI API Compatibility*

OpenAI API Compatibility refers to the ability of third-party model serving frameworks and tools—such as vLLM, Triton, or LLaMA Factory—to mimic or support the interface conventions established by OpenAI's API. This includes replicating endpoint structures (`/v1/chat/completions`, `/v1/embeddings`), request/response schemas, and token usage semantics, allowing developers to seamlessly swap OpenAI models with custom-hosted or open-source alternatives without rewriting client-side code.

Frameworks with OpenAI API compatibility provide drop-in replacement capabilities, enabling existing applications built around GPT-3.5/4 to interface with alternative LLMs like LLaMA, Mistral, or Gemma using the same HTTP endpoints and JSON format. This abstraction layer is especially useful in enterprise settings where developers want to maintain feature parity while reducing API costs, improving latency, or operating in privacy-sensitive contexts.

#### *2.3.31 HNSW Index*

The Hierarchical Navigable Small World (HNSW) index is a graph-based structure used for approximate nearest-neighbor (ANN) search in high-dimensional vector spaces. It builds a multi-layered proximity graph where each node represents a vector and edges link nearby vectors. HNSW offers logarithmic search complexity, balancing speed and recall. In AURA, HNSW is used via FAISS and Milvus, configured with parameters like `M=48`, `efConstruction=200`, and `efSearch=64`, enabling sub-100 ms vector search for up to millions of embeddings.

#### *2.3.32 SentencePiece*

SentencePiece is a language-agnostic subword tokenizer used to process raw text into token sequences without relying on language-specific pre-tokenization. It enables better handling of rare words and subword units, which is critical for training and inference in models like BGE-M3. SentencePiece operates by learning a vocabulary of subword units using algorithms like Byte-Pair Encoding (BPE) or Unigram LM. In AURA, SentencePiece with a vocabulary size of 50k is used for chunking and embedding long Thai knowledge base documents.

#### *2.3.33 ICU Thai Analyzer*

The ICU Thai Analyzer is a text segmentation tool built into Elasticsearch that supports Thai-language tokenization using the Unicode Common Locale Data Repository (CLDR). Unlike Latin-based languages, Thai lacks spaces between words, so accurate segmentation is essential for lexical search. In AURA's BM25 pipeline, the ICU Thai Analyzer ensures meaningful term extraction for document indexing and query matching in ElasticSearch, improving the effectiveness of traditional IR components.

#### *2.3.34 ColBERT Multi-Vector Retrieval*

The ColBERT retrieval approach produces a matrix of contextualized token embeddings for both query and document. Instead of collapsing them into a single vector (like traditional dense retrieval), it uses token-level max-similarity to compute relevance. This technique allows better fine-grained matching and is especially useful when applied with longer documents or semantically dense KB chunks. AURA's hybrid retriever leverages this via BGE-M3's colbert-style mode to improve top-k accuracy during semantic search.

#### *2.3.35 Triton gRPC Inference (Cross-Encoder Serving)*

Triton Inference Server supports serving ML models via high-throughput REST and gRPC APIs. In the AURA pipeline, it is used to host the BGE-M3 reranker as a gRPC service. Each request passes a

query and a set of candidate chunks to the server, which returns relevance scores computed via a cross-encoder. Triton supports batching, GPU inference, and model versioning, making it a key part of AURA’s production-grade reranking stack. This setup ensures low-latency, scalable reranking in real time.

## **Chapter 3**

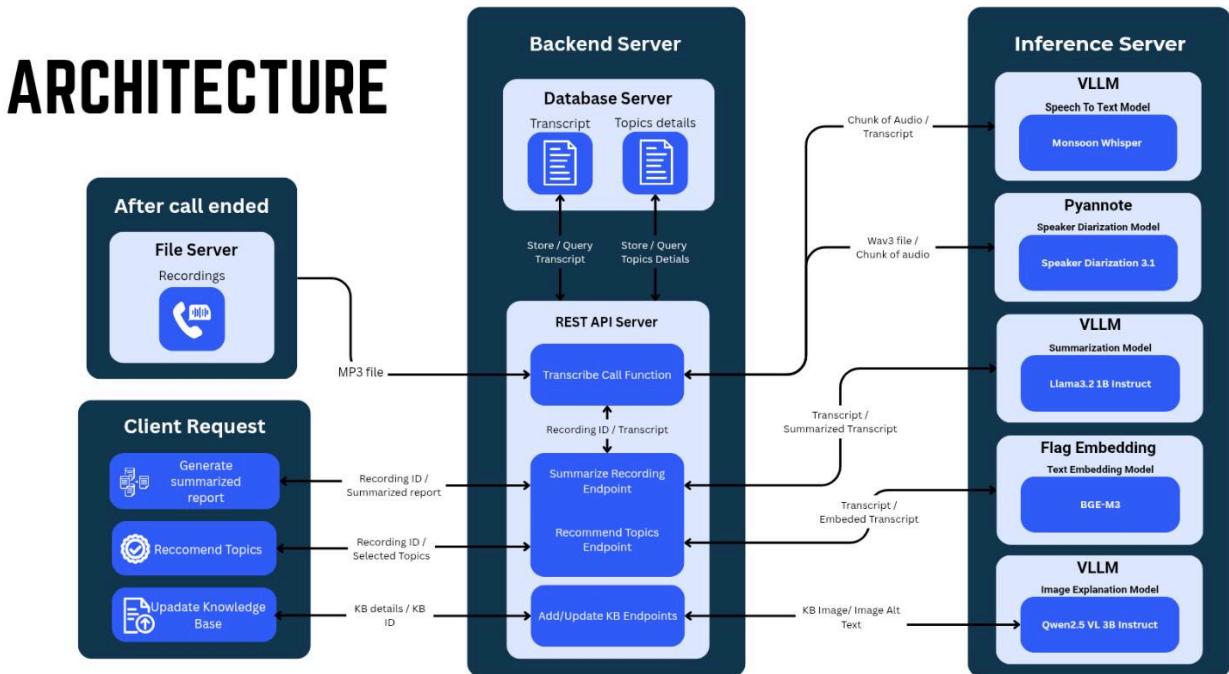
### **Methodology**

#### **3.1 User Research**

We had a virtual meeting with the NHO where we discussed the organisation’s pain points and needs, and developed solutions, which we had presented to the Contact Center and IT personnel on our NHO Head Office on-site visit, on the 7th of November 2024. There, we had the opportunity to meet the relevant personnel in person, and visit the Contact Center in person to get a better grasp of the workflow and struggles

of an agent, from which we received our revised scope of Automatic Call Data Analysis according to CRM data, in which we would recommend the correct entries of CRM fields based on processed data from call recordings. We would design our solution from then on. [33]

## 3.2 System Design



### 3.2.1 System Artifacts and Components

#### 3.2.1.1 Inputs

##### 3.2.1.1.1 Call Recording from CRM System

When a call ends in the CRM system, an MP3 file of the recording is generated and sent to the backend server for transcription and further processing. This serves as the primary input for tasks such as summarization and classification.

##### 3.2.1.1.2 Client Requests from CRM System

Clients interact with the system through the CRM interface, sending various requests, such as:

##### 3.2.1.1.3 Generate Summarized Report:

Request for a concise summary of the call recording

##### 3.2.1.1.4 Auto-Select Knowledge Base (KB):

Request to retrieve the most relevant knowledge base for a specific context.

#### 3.2.1.2 Database Server

##### 3.2.1.2.1 Transcripts:

Text generated from the transcribed call recordings.

*3.2.1.2.2 Knowledge Bases (KBs):*

Pre-defined knowledge bases used for classification and retrieval tasks.

### **3.2.1.3 REST API Server**

The REST API Server acts as the central coordinator between client requests and the inference server, providing multiple endpoints:

*3.2.1.3.1 Transcribe Call Endpoint:*

Converts MP3 recordings into text using the inference server's speech-to-text model.

*3.2.1.3.2 Summarize Recording Endpoint:*

Produces a summarized version of the transcript.

*3.2.1.3.3 Classify Knowledge Base Endpoint:*

Identifies and retrieves the most relevant KB for the given transcript.

*3.2.1.3.4 Add/Update/Delete Knowledge Base Endpoint:*

Add new knowledge base or update and delete current knowledge base

### **3.2.1.4 Inference Server**

The Inference Server houses advanced models to perform computationally intensive tasks, including:

*3.2.1.4.1 Speech-to-Text:*

Transcribes MP3 recordings into text using models such as Whisper TH and pyannote-audio.

*3.2.1.4.2 Summarization Model:*

Summarizes transcripts using PEGASUS-X-large or LLaMA3.2-1B-Instruct models.

*3.2.1.4.3 Knowledge Base Classification Model:*

Employs Retrieval-Augmented Generation (RAG) to classify transcripts into the most relevant KB.

*3.2.1.4.4 Vision Language Model:*

Explain the image into text using Qwen2.5 VL 3B Instruct and Gemeni2.0 API

## *3.2.2 Pipeline Process Explanation*

### **3.2.2.1 Receive Call Recording from CRM System**

When a call ends in the CRM system, the system generates an MP3 file of the recording. This file is sent to the backend server, specifically to the Transcribe Call Endpoint, which

initiates the transcription process. The call recording serves as the primary input for all subsequent steps in the pipeline.

### **3.2.2.2 Transcribe the Call Recording**

The Transcribe Call Endpoint forwards the MP3 file to the Inference Server, where the Speech-to-Text Model (e.g., Whisper TH or pyannote-audio) processes the audio and converts it into a transcript. The generated transcript is then stored in the Transcript Repository of the database server for future use.

### **3.2.2.3 Summarize the Transcript**

If the client requests a summarized report, the Summarize Recording Endpoint retrieves the transcript from the database server and sends it to the Inference Server. Here, the Summarization Model (e.g., PEGASUS-X-large or LLaMA3.2-1B-Instruct) generates a concise summary of the transcript. The summary is returned to the backend server and sent to the client as a summarized report.

### **3.2.2.4 Classify Knowledge Base (KB)**

For KB classification requests, the Classify Knowledge Base Endpoint retrieves the transcript from the database server and sends it to the Inference Server, where the Knowledge Base Classification Model (e.g., RAG) determines the most relevant knowledge base. The selected KB is sent back to the backend server and then to the client.

### **3.2.2.5 Add, Update and Delete Knowledge Base**

When any add or update new knowledge base, it will pass to send to clean the text and process the image into text with VLM and save into the database server.

### **3.2.2.6 Client Receives Results**

The client receives the requested outputs (e.g., summarized report, selected KB, disease, or policy) through the CRM interface.

## *3.2.3 Technology Stack*

The system is designed as a modular and scalable architecture composed of loosely coupled services, each containerized using Docker and orchestrated for efficient deployment. The core components span four major layers: the API backend, model inference engines, databases, and deployment infrastructure.

### **3.2.3.1 Backend/API Server – FastAPI**

The backend is implemented using FastAPI, a high-performance Python web framework ideal for building APIs. FastAPI provides automatic validation, type checking, and OpenAPI documentation, enabling fast development while supporting asynchronous processing. It serves as the central coordinator, handling user requests (e.g., transcription, summarization, knowledge base management) and orchestrating inference calls to other microservices.

### **3.2.3.2 Model Inference Engine – VLLM, FlagEmbedding, Pyannote**

The inference layer hosts various models, each deployed as an independent microservice:

#### *3.2.3.2.1 VLLM*

VLLM is used to serve large language models such as LLaMA and Qwen for tasks like summarization and natural language generation. It introduces paged attention, an innovative memory management strategy that enables efficient GPU utilization. This allows the system to process multiple requests concurrently with minimal latency. VLLM is also designed to be OpenAI-compatible, meaning it can expose endpoints that follow the same API format as OpenAI's services, making it easy to integrate with existing applications that expect that interface.

#### *3.2.3.2.2 FlagEmbedding*

FlagEmbedding is a lightweight, production-ready embedding server that encodes textual input into dense vector representations in real time. It is especially effective for powering semantic retrieval and reranking pipelines, where relevance is determined by vector similarity rather than keyword matching. FlagEmbedding is often paired with models like BGE-M3 to ensure multilingual and domain-adaptable embeddings for tasks such as knowledge base search, intent classification, and clustering.

#### *3.2.3.2.3 Pyannote-audio*

Pyannote-audio is deployed as a standalone REST service that performs speaker diarization, the process of detecting and labeling distinct speakers in an audio stream. It segments a conversation into speaker-specific time intervals, enabling the system to differentiate between speakers even in overlapping or noisy conditions. This functionality is critical for structuring call center transcripts and supports better readability and downstream analysis such as speaker-specific intent recognition or role tagging.

Each model service is isolated in its own container, allowing for independent scaling and updates.

### **3.2.3.3 Database – PostgreSQL, Elasticsearch, FAISS**

The database layer combines three complementary storage systems to support different types of querying:

#### *3.2.3.3.1 PostgreSQL*

PostgreSQL stores structured data such as transcripts, metadata, and processed KB entries. It supports reliable querying and is well-suited for transactional operations in the system.

#### *3.2.3.3.2 Elasticsearch*

Elasticsearch enables fast, keyword-based search over knowledge base documents using full-text indexing, making it ideal for direct phrase or term matching.

#### *3.2.3.3.3 FAISS*

FAISS performs similarity search over dense vector embeddings, enabling retrieval of semantically relevant KB entries even without exact keyword overlap.

This hybrid architecture allows both fast keyword lookups and semantic relevance scoring to operate in parallel.

### **3.2.3.4 Deployment – Docker**

All services are containerized using Docker, allowing for isolated, reproducible, and portable deployments. Docker Compose is used to define and manage multi-service configurations, enabling local development and cloud deployment consistency. This setup simplifies scaling individual components (e.g., inference servers) and ensures version control and dependency management across the full stack.

## 3.2 Data Synthesis **Kasidis Manasurangkul**

Since last semester, we have not received any real-world conversation data from NHSO. Given the continued uncertainty and the low likelihood of gaining access to this data in time, we decided to synthesize a set of conversation transcripts to ensure continued progress in other components of the project. These synthetic dialogues simulate interactions between call center agents and callers, and were generated using the knowledge base documents provided by NHSO during the previous term.

To enhance the realism of the dataset and support a broader range of downstream tasks—such as summarization and classification—we structured each synthetic transcript to include between 1 and 4 distinct knowledge base (KB) entries. This design enabled evaluation of model performance under varying levels of complexity and information density. As part of the synthesis process, we performed knowledge base preprocessing and generated transcripts across seven distinct sub-datasets.

### 3.2.1 Knowledge Base Processing

In this process, we experimented with various models to iterate over the preprocessed KB, identify images, and feed them into a vision-language model to generate explanatory text. We tested multiple models, as outlined below.

Since this model needs to be implemented as an API in the NHSO system and used every time a new KB is introduced, we prioritized smaller models. Using a large model would lead to unnecessary waste of computational resources.

#### 3.2.1.1 Knowledge Base text processing

In the first version of KB text processing, there were still some errors, and the output was quite messy. The table and image tags were not correctly ordered, causing all tables and image tags to appear before the text. This could lead to misinterpretation by the classification model, as it might fail to understand the correct context.

To address this, we corrected the order by sorting text, images, and tables to match their actual placement in the document. Additionally, we removed all formatting and styles, such as shapes in DOCX, LaTeX syntax, and highlights, which appeared as unreadable text. We also shortened excessive dots to reduce token consumption when reading the KB and performed another round of text cleaning.

#### 3.2.1.2 Generating image explanation with vision language models

In this process, we experimented with various models to iterate over the preprocessed KB, identify images, and feed them into a vision-language model to generate explanatory text. We tested multiple models, as outlined below.

##### 3.2.1.2.1 Deepseek-vl2-tiny

This was the first model we tested. Although the model could read Thai text in images, it failed to generate output in Thai. To solve this issue, we had the model generate output in English first and then used Llama 3.2 3B Instruct to translate it into Thai. This approach was successful.

[รูปภาพ] ชื่อความแสดงภาพแสดงสองช่องที่ชื่อว่า "Document 1" และ "Document 2"

โดยที่แต่ละช่องมีข้อมูลที่จะอธิบายถึงความเกี่ยวข้องกับภาษาไทยต่างๆ Document 1 มีข้อมูลที่แสดงถึงความเกี่ยวข้องของคำศัพท์ต่างๆ เช่น พาหะ, ใจ, จิต, ใจ, จิต ซึ่งอาจเป็นรายชื่อของข้อมูลที่เกี่ยวข้องกับภาษาไทย หรือรายละเอียดอื่นๆ ของคำศัพท์ ที่ใช้ในภาษาไทยอยู่ Document 2 ยังมีข้อมูลที่แสดงถึงความเกี่ยวข้องของคำศัพท์ต่างๆ เช่น พาหะ, ใจ, จิต, ใจ, จิต ซึ่งอาจเป็นรายชื่อของข้อมูลที่เกี่ยวข้องกับภาษาไทย หรือรายละเอียดอื่นๆ ของคำศัพท์ ที่ใช้ในภาษาไทยอยู่ ทั้งสองช่องนี้มีข้อมูลที่เกี่ยวข้องกับภาษาไทย แต่ไม่ได้ระบุรายละเอียดเพิ่มเติม ทำให้ต้องใช้เวลาอ่านและทำความเข้าใจเพิ่มเติม โดยมีความซ้ำซ้อนและไม่ชัดเจน [รูปภาพ]

However, as seen in the results above, the model still produces redundant text and lacks detailed descriptions of the images. This can result in overly long explanations, making it more difficult for the classification model to utilize, as it requires additional computational resources and processing time.

### 3.2.1.2.2 Qwen2.5-VL-3B-Instruct

This model produced much better results but still failed to generate output in Thai. Therefore, we continued using Llama 3.2 3B Instruct for translation. However, hosting this model required additional preprocessing steps since it could not process images with a transparent background. To address this, we removed all such images, as our analysis showed that most transparent-background images in the KB were unrelated icons.

[รูปภาพ]ภาพประกอบมีส่วนที่อธิบายถึงความเกี่ยวข้องในภาษาไทย ส่วนแรกมีชื่อรูปว่า "ตัวอย่างที่ 1" และส่วนที่สองมีชื่อรูปว่า "ตัวอย่างที่ 2" ทั้งสองส่วนมีเนื้อหาในภาษาไทย โดยส่วนแรกนำเสนอตัวอย่างเอกสารที่มีข้อมูลเฉพาะ เช่น วันที่ เวลา และหมายเหตุแบบจำลอง ส่วนที่สองยังมีข้อมูลที่คล้ายกัน แต่มีข้อมูลที่แตกต่างกัน ส่วนทั้งสองนี้เครื่องหมายสำคัญที่ใช้ไปที่ส่วนของข้อความที่สำคัญ ส่วนข้อความในภาพประกอบเป็นภาษาไทย และดูเหมือนจะอธิบายถึงความเกี่ยวข้องกับเอกสารหรือแบบฟอร์ม อาจใช้สำหรับวัตถุประสงค์ทางการบริหารจัดการหรือทางกฎหมาย [รูปภาพ]

As seen in the results above, this model performs much better in OCR text extraction and generates more concise yet informative descriptions. However, one concern is its lack of robustness, as it sometimes tends to repeat the same words repeatedly.

To address this, I attempted an agentic approach by assigning Llama 3.2 3B not only to translate but also to fix typos and handle repeated words. However, since this Llama model is not very large, it was only able to translate the text but failed to correct typos.

### 3.2.1.2.3 Gemini 2.0 Flash

Since Qwen2.5-VL-3B-Instruct still could not achieve satisfactory results and required two separate models—one for extracting data from images and another for translation—we decided to try an API-based model like Gemini 2.0 Flash instead. This model is considerably cheaper and is reported to outperform current open-source vision-language models. Additionally, Gemini 2.0 Flash can generate output directly in Thai without requiring a separate translation model, simplifying the pipeline. Given that the Knowledge Base (KB) does not contain confidential data, using an API-based model should be an acceptable solution.

For the Gemini 2.0 Flash free tier, there is a usage limitation of 1,000,000 tokens per day and 15 requests per minute. To work within these constraints, I configured my

code to send only 15 requests per minute and switched to another account when the limit was reached. Below are the results.

[รูปภาพ] ก้าพนีแบ่งออกเป็นสองส่วนหลัก, ส่วนและขวา แต่ละส่วนแสดงข้อมูลที่คล้ายกันแต่มีรายละเอียดที่แตกต่างกัน

ส่วนซ้าย:

1. หัวช้อ: "(ตัวอย่างที่ 1)"
2. ข้อความ: "หากที่นี้ขอความตั้งกล่าวต้องรอประมาณ 2 วัน จึงจะสามารถย้ายสิทธิ์ใหม่ได้ในทันที"
3. ส่วนข้อมูล: ประกอบด้วยรายการข้อมูลต่อไปนี้:
  - สิทธิ์ที่ใช้เบิก: สิทธิ์หลักประกันสุขภาพแห่งชาติ (ยกเว้นการร่วมจ่ายค่าบริการ 30 บาท)
  - ประเภทสิทธิ์อย: เด็กอายุไม่เกิน 12 ปีบริบูรณ์
  - รหัสบัตรประกันสุขภาพ: N719427392271
  - วันที่เริ่มใช้หน่วยบริการ: 14 กุมภาพันธ์ 2566 เวลา 14:41 น.
  - วันหมดสิทธิ์อย: 8 กุมภาพันธ์ 2578 เวลา 23:59 น.
  - จังหวัดที่ลงทะเบียนรักษา: ปัตตานี
  - หน่วยบริการปฐมภูมิ: รพ.สต.บ้านป่าตาบูด (09988)
  - หน่วยบริการที่รับส่งต่อ: รพ.ยะหรีง (11429)
- Model: 1
- จำนวนครั้งที่เปลี่ยนหน่วยบริการประจำ: 1
- หน่วยบริการประจำ: รพ.ยะหรีง (11429)
4. ข้อความในกรอบ: "ลงทะเบียนสิทธิ์ว่างรายใหม่เกิดสิทธิ์ทันที"
5. ข้อความด้านล่าง: "รอประมาณ"

ส่วนขวา:

1. หัวช้อ: "(ตัวอย่างที่ 2)"
2. ข้อความ: "หากไม่มีข้อความข้างบนนี้ การณ์สามารถย้ายสิทธิ์ได้ทันที ไม่ต้องรอประมาณ 2 วัน"
3. ส่วนข้อมูล: ประกอบด้วยรายการข้อมูลต่างๆ ดังนี้:
  - สิทธิ์ที่ใช้เบิก: สิทธิ์หลักประกันสุขภาพแห่งชาติ
  - ประเภทสิทธิ์อย: ช่วงอายุ 12-59 ปี
  - รหัสบัตรประกันสุขภาพ: N891024447747
  - วันที่เริ่มใช้หน่วยบริการ: 18 กันยายน 2565 เวลา 18:18 น.
  - วันหมดสิทธิ์อย: 28 กุมภาพันธ์ 2585 เวลา 23:59 น.
  - จังหวัดที่ลงทะเบียนรักษา: กรุงเทพฯ
  - หน่วยบริการปฐมภูมิ: พร้อมมงคลคลินิกเวชกรรม สาขารามคำแหง 2 (41990)
  - หน่วยบริการที่รับส่งต่อ: รพ.สปริง (15049)
- Model: 5
- จำนวนครั้งที่เปลี่ยนหน่วยบริการประจำ: 0
- หน่วยบริการประจำ: ศูนย์บริการสาธารณสุข 57 บุญร่อง สันเตต (13700)
4. ข้อความด้านล่าง: "ไม่ต้องรอประมาณ"

[รูปภาพ]

Gemini 2.0 Flash performed excellently in extracting data from images, producing text that perfectly matches the content in the image while maintaining a readable format. Although the text length is slightly longer than ideal, it does not contain unnecessary parts and retains more key keywords from the image compared to Qwen2.5-VL-3B-Instruct.

	Free Tier	Paid Tier, per 1M tokens in USD
Input price	Free of charge	\$0.10 (text / image / video) \$0.70 (audio)
Output price	Free of charge	\$0.40
Context caching price	Free of charge	\$0.025 / 1,000,000 tokens (text/image/video) \$0.175 / 1,000,000 tokens (audio) Available March 31, 2025
Context caching (storage)	Free of charge, up to 1,000,000 tokens of storage per hour Available March 31, 2025	\$1.00 / 1,000,000 tokens per hour Available March 31, 2025
Tuning price	Not available	Not available
Grounding with Google Search	Free of charge, up to 500 RPD	1,500 RPD (free), then \$35 / 1,000 requests
Used to improve our products	Yes	No

For the price of Gemini 2.0 Flash, it costs only \$0.10 per 1 million input tokens and \$0.40 per 1 million output tokens. On average, a KB contains only two images, with each image generating approximately 2,048 output tokens. Therefore, the cost for processing one KB is approximately \$0.2016 or 6.81 THB, which is considered very inexpensive since this process is done only once.

### 3.2.2 Transcript Synthesis

To support a variety of experimental settings, we synthesized transcripts with different configurations, varying the number of knowledge base (KB) entries per conversation. In total, we created seven distinct versions of the dataset

#### 3.2.2.1 First version of synthetic dataset

In our initial attempt at generating synthetic conversation data with 1 knowledge base, we employed DeepSeek R1 685B, a large language model, to simulate phone call dialogues between call center agents and citizens. These conversations were based on real topics from NHSO knowledge bases provided to us in the previous semester.

To guide the model's output, we crafted a detailed system prompt instructing DeepSeek to produce realistic, Thai-language conversations between callers and agents. Since the NHSO examples we received were summaries rather than actual dialogue transcripts, we manually wrote three full example conversations derived from those summaries and included them in the prompt to help shape the model's response style.

Given the model's scale (685 billion parameters), we deployed it using eight H200 GPUs with the vLLM framework to serve inference requests efficiently. Conversation generation was done via Python scripts, which provided the model with prompts and knowledge base documents. However, due to the model's context limit of 128,000 tokens, it was not feasible to include all KB entries at once. Instead, we rotated through different KBs every 10 requests to balance coverage and token constraints. A temperature of 0.9 was used to encourage output diversity and reduce repetition when generating dialogues from the same KB.

Here is an example of a generated conversation based on KB1346:

--- Conversation Record #11 (KB=KB1346.txt) ---

[Output]:

ตัวอย่างบทสนทนาจำลองระหว่างผู้ป่วยครองกับเจ้าหน้าที่หน่วยบริการ

ผู้ป่วยครอง: สวัสดีค่ะ ลูกชายเรียนอยู่ชั้นป.4 มีอาการตามมัวบอยๆ บอกว่าม่องกระดานในห้องไม่ชัด  
อย่างพากเพรา

เจ้าหน้าที่: สวัสดีครับ คุณแม่มีความสามารถอ่านชื่อของมาให้สิทธิ์บัตรทองได้แล้วครับ แนะนำให้  
คิดต่อหน่วยบริการที่คุณแม่สังกัดอยู่ หรือสถานพยาบาลใกล้บ้านที่เข้าระบบสปสช. เช่น รพ.สต.

หรือคลินิกชุมชนอุบลฯได้ครับ

ผู้ป่วยครอง: ถ้าตราชลาภว่าสายตาสั้น จะได้วันตราฟรีเลยใช่ไหมคะ?

เจ้าหน้าที่: ใช่ครับ หากจักษุแพทย์ยืนยันว่ามีสายตาผิดปกติ เช่น สั้น/ยาว/เอียง ทาง

โครงการจะครอบคลุมค่าเลนส์สายตาและแวนคนละ 1 อันต่อปี รวมถึงนัดติดตามผลหลังใช้แวน 6  
เดือนด้วยครับ

ผู้ป่วยครอง: แล้วถ้าแวนหายหรือแตกละคะ? ขออันใหม่ได้ไหม?

เจ้าหน้าที่: กรณีแบบนี้ต้องปรึกษาหน่วยบริการที่ดัดแวนให้ก่อนนะครับ เพราะสปสช. เหมาจ่ายค่า<sup>แวนปีละ 1 อัน</sup> แต่ถ้ามีเหตุจำเป็นจริงๆ อาจพิจารณาเป็นรายกรณีครับ

ผู้ป่วยครอง: ลูกชาย 13 ปีแต่ยังเรียนป.6 อยู่ ลูกยังเข้าข่ายโครงการนี้ได้อยู่ไหมคะ?

เจ้าหน้าที่: ได้ครับ! เกณฑ์โครงการคุณต้องอายุและระดับชั้น ถ้าน้องอายุไม่เกิน 12 ปี \*หรือ\* กำลัง<sup>เรียนอยู่ชั้นป.6</sup> แม้จะอายุ 13 ก็ยังมีสิทธิ์อยู่ครับ

ผู้ป่วยครอง: โครงการนี้ดัดแวนในโรงพยาบาลด้วยใช่ไหมคะ?

เจ้าหน้าที่: ใช่ครับ ทั้งโรงพยาบาลและเอกชน ครุณคัดกรองสายตาเบื้องต้น หากพบปัญหา ก็ส่งต่อ<sup>รพ.สต.</sup> และหน่วยบริการที่มีจักษุแพทย์ตรวจยืนยันอีกครั้งครับ

ผู้ป่วยครอง: ขอบคุณมากนะคะ เดียวจะไปเตรียมสูติบัตรลูกเพื่อไปขอใช้บริการเลย!

เจ้าหน้าที่: ไม่มีปัญหาเลยครับ หากมีข้อสงสัยเพิ่มเติม โทรมาสอบถามได้ตลอดเวลานะครับ

After generation, we applied post-processing to clean the outputs. This included removing headers like “ตัวอย่างบทสนทนาจำลอง...”, deleting any non-verbal formatting (e.g., []), and stripping emojis or irrelevant metadata. Despite this, the initial version of the dataset remained noisy, with several persistent issues:

- Chinese phrases occasionally appeared in labels or dialogue.
- Speaker tags were inconsistent (e.g., ผู้ป่วยครอง, แม่, ผู้โทร).
- Some conversations contained non-dialogue content like topic titles or summaries.

To address these quality concerns, we later transitioned to an agentic two-pass synthesis approach. In this updated method, the first model generated the conversation, and a second model acted as a validator, cleaning up inconsistencies, correcting speaker names, and refining dialogue for coherence. This substantially improved the clarity and realism of the generated transcripts, bringing them closer to what real-world speech-to-text systems would produce.

Out of 4,720 initially generated entries across 472 KBs, we discarded 434 entries during post-processing, leaving 4,286 usable conversations.

### 3.2.2.2 Second version of synthetic dataset

In this second version, we aim to synthesize conversations consisting of two KBs, making them more realistic since, in the real world, there are cases where an agent and caller discuss multiple related topics in a single conversation.

However, this time, we no longer have access to the SiamAI cluster with H200, meaning we cannot use DeepSeek R1 (685B parameters). Instead, we experimented with two smaller models: Llama 3.3 70B Instruct and Qwen2.5 72B Instruct.

For Llama 3.3, although the model is highly robust in the Thai language, it performed poorly in role-playing scenarios. It struggled to generate informal Thai speech, making the conversation feel more like a structured document rather than a natural dialogue. After testing the first 10 entries, we decided to stop using Llama and switched to Qwen2.5 72B Instruct, which yielded much better results.

Qwen 2.5 performed exceptionally well in role-playing—it could naturally generate informal Thai speech, mimicking real human conversations with pauses and natural phrasing. However, it was not as robust as Llama and occasionally introduced foreign languages, topic headers, or summaries at the end, similar to DeepSeek R1. Surprisingly, despite being much smaller than DeepSeek R1, Qwen 2.5 produced significantly cleaner outputs, reducing the need for an agentic method.

Since we need two KBs per conversation in this version, we constructed KB pairs by selecting KBs sequentially and then randomly choosing 50 KBs from a total of 468 to pair with each. This resulted in 23,400 KB pairs.

However, generating these KB pairs required additional processing due to hardware limitations. Using 8 A100 GPUs, we could handle a maximum context length of 32,678 tokens for the 72B model. However, token analysis on the KBs revealed that the highest token count for a single KB was around 40,000 tokens, with 40 KBs exceeding 12,000 tokens. Since we needed to feed two KBs along with the system prompt into the model, this would have caused an out-of-memory (OOM) error. To prevent this, we had no choice but to truncate KBs that exceeded the model's token limit.

After 15 hours of synthesized conversation, we completed the process and performed post-processing to clean the results. In the end, we successfully generated 21,310 entries. Below is an example of the results.

--- Conversation Record #21306 (KB=KB1393.txt,KB1142.txt) ---

[Output]:

ผู้พูด1:

สวัสดีค่ะ สายด่วนหลักประกันสุขภาพแห่งชาติ 1330 มีอะไรให้ช่วยเหลือคะ?

ผู้พูด2:

สวัสดีค่ะ อยากรู้ว่าสามารถเรื่องศูนย์คุ้มครองสิทธิบัตรทองค่ะ หนูไม่แน่ใจว่าจะต้องติดต่ออย่างไร แล้วท่านจะได้รับเอกสารใดเมื่อได้สิทธิในการใช้บริการบัตรทอง

ผู้พูด1:

สวัสดีค่ะ ศูนย์คุ้มครองสิทธิบัตรทอง หรือที่เรียกว่าหน่วย 50(5) เป็นหน่วยงานที่รับเรื่องร้องเรียนและช่วยเหลือผู้ที่ถูกกล่าวหาว่าได้ลักทรัพย์ในกระบวนการรับบริการบัตรทองค่ะ ไม่ทราบว่าคุณอยู่จังหวัดไหนคะ? จะได้แนะนำสำนักงานที่รับผิดชอบให้

ผู้พูด2:

หนูอยู่กรุงเทพค่ะ และถ้าต้องการร้องเรียน จะต้องทำยังไงบ้างคะ?

ผู้พูด1:

ค่ะ ถ้าคุณอยู่กรุงเทพฯ สามารถติดต่อศูนย์คุ้มครองสิทธิบัตรทอง ที่สำนักงาน สปสช. เขต 13 ได้ค่ะ ที่อยู่คือ 123 ถนนวิภาวดีรังสิต แขวงลาดพร้าว เขตลาดพร้าว กรุงเทพมหานคร 10230 หรือโทรศัพท์ที่ 02-123-4567

ผู้พูด2:

แล้วถ้าจะร้องเรียน ต้องเตรียมอะไรบ้างคะ?

ผู้พูด1:

เมื่อคุณต้องการร้องเรียน ให้เตรียมข้อมูลต่างๆ ดังนี้ค่ะ ชื่อ-นามสกุล ที่อยู่ บัตรประชาชน เลขบัตรประจำตัวประชาชน รายละเอียดของเหตุการณ์ที่เกิดขึ้น รวมถึงวัน เวลา และสถานที่ที่เกิดเหตุ สุดท้ายคือช่องทางติดต่อกลับ ทั้งเบอร์โทรศัพท์และที่อยู่อีเมล

**ผู้พูด2:**

เข้าใจแล้วค่ะ แล้วถ้าเข้าสถานะเรื่องร้องเรียน ต้องทำยังไงค่ะ?

**ผู้พูด1:**

ค่ะ หลังจากร้องเรียนเรียบร้อยแล้ว ศูนย์คุ้มครองสิทธิบัตรทองจะมีการบันทึกเรื่องร้องเรียนและจะติดต่อกลับเพื่อแจ้งผลการดำเนินการภายใน 30 วัน คุณสามารถสอบถามสถานะการร้องเรียนได้ที่เบอร์โทรศัพท์ที่ให้ไว้ หรือติดต่อผ่านอีเมลได้ค่ะ

**ผู้พูด2:**

ขอบคุณมากค่ะ แล้วถ้าเรื่องหนึ่งค่ะ หมูเห็นว่ามีบริการตู้ห้องเย็น ที่ให้บริการตรวจสุขภาพและปรึกษาแพทย์ออนไลน์ อยากรู้ว่ามีอยู่ที่ไหนบ้าง และใช้งานยังไง?

**ผู้พูด1:**

ค่ะ ตู้ห้องเย็น เป็นบริการสุขภาพทางเลือกใหม่ที่ให้คุณสามารถตรวจสุขภาพและปรึกษาแพทย์ออนไลน์ได้ค่ะ ปัจจุบันตู้ห้องเย็นมีการนำร่องในพื้นที่กรุงเทพฯ ที่ 2 แห่ง คือ 1. ศูนย์ฯ ฯ และ 2. อาคารทัช บลัดดิ้ง เชตท้ายหวาน

**ผู้พูด2:**

โอเคค่ะ และวิธีการใช้งานตู้ห้องเย็นล่ะคะ?

**ผู้พูด1:**

เมื่อคุณไปยังจุดให้บริการตู้ห้องเย็น ขั้นตอนการใช้งานมีดังนี้ค่ะ ขั้นตอนแรก ให้คุณเสียบบัตรประชาชนที่ตู้ห้องเย็น ตู้จะตรวจสอบสิทธิบัตรทองของคุณ หากพบว่าคุณมีสิทธิ ประตูตู้จะเปิดออก ค่ะ

**ผู้พูด2:**

แล้วถ้ามาต้องทำอะไรต่อคะ?

**ผู้พูด1:**

ต่อไป คุณจะสามารถตรวจสุขภาพเบื้องต้นได้ค่ะ อาทิเช่น วัดความดันโลหิต วัดส่วนสูง น้ำหนัก และวัดค่าอوكซิเจนในเลือด ผลการตรวจจะใช้ประกอบการปรึกษาแพทย์ออนไลน์ค่ะ ขั้นตอนต่อไปคือ การปรึกษาแพทย์ออนไลน์ผ่านการวิดีโอดล อุณจะได้รับคำแนะนำจากแพทย์ และสามารถขอรับยาได้ 2 แบบ คือ 1. ให้รีดอีริปส์ยาหัวหิน หรือ 2. เลือกรับยาที่ร้านยาที่เข้าร่วมโครงการ ซึ่งปัจจุบันมีเพียงร้านยากรุงเทพฯ เท่านั้น

**ผู้พูด2:**

เข้าใจแล้วค่ะ แล้วถ้าต้องการดูผลตรวจและข้อมูลยังไงล่ะคะ?

**ผู้พูด1:**

ค่ะ หลังจากปรึกษาแพทย์เรียบร้อยแล้ว คุณสามารถสแกน QR Code เพื่อรับข้อมูลผลตรวจและข้อมูลยาที่ได้รับค่ะ นอกจากนี้ยังสามารถดูผลตรวจและคำแนะนำจากแพทย์ได้ผ่านแอปพลิเคชัน LINE หรือโทรศัพท์มือถือของคุณ

**ผู้พูด2:**

ขอบคุณมากค่ะ นี่เป็นข้อมูลที่ดีมากเลย

**ผู้พูด1:**

ยินดีค่ะ ถ้ามีคำถามเพิ่มเติมหรือต้องการความช่วยเหลือ สามารถติดต่อเราได้ที่ 1330 ได้ตลอด 24 ชั่วโมงนะคะ

**ผู้พูด2:**

ค่ะ ขอบคุณมากค่ะ สวัสดีค่ะ

**ผู้พูด1:**

สวัสดีค่ะ ยินดีที่ช่วยเหลือค่ะ

The results from this dataset are usable but not perfect, as there are still some issues. When generating more than 20,000 entries, the model tends to reuse the same scenario, changing only the questions. Additionally, the model often generates simple or straightforward questions, which may not accurately represent real-world situations.

Moreover, the model still struggles to truly integrate two topics into a single conversation. Instead, it often connects them in a rigid and unnatural way, using sentences like:

"ขอบคุณมากค่ะ แล้วอีกเรื่องหนึ่งค่ะ ที่นี่มีบริการตู้ห้องเย็น ที่ให้บริการตรวจสอบสภาพและปรึกษาแพทย์ออนไลน์ อยากรู้ว่ามีอยู่ที่ไหนบ้าง แล้วใช้งานยังไง?"

Even though we set the system prompt to combine both topics into a seamless conversation, it did not work well. This is likely because the model cannot effectively multitask within a single prompt, and in some cases, the randomly paired KBs do not naturally fit within the same context.

### 3.2.2.3 Third version of synthetic dataset

In this version, we continued generating conversations using two KBs per conversation, but we made major improvements to the data synthesis process.

In the previous version, we used Qwen2.5-72B-Instruct to generate over 20,000 conversations. However, the results were not as good as expected — the conversations were too simple and didn't sound natural. Another issue was the context length limit: each KB document could be up to 15,000 tokens, and using two KBs almost filled the model's memory. Since we plan to support three or four KBs in the future, we needed a new method.

So in this version, we redesigned the pipeline that split the task into smaller parts to reduce memory usage and improve quality instead of using one large model to read all KBs, create a scenario, and write the conversation all at once:

#### 3.2.2.3.1 Find similar KB pairs

For each KB, we treated it as a primary KB and compared it with all other KBs using BM25 to measure similarity. We then selected the top 20 most similar KBs as candidate pairs. This process was repeated for every KB, allowing us to build conversation groups based on related content. To ensure balance in the dataset, we applied a penalty to the similarity score—reducing the score of KBs that had already been used too many times. This helped prevent certain KBs from appearing too frequently and reduced the risk of bias in downstream tasks like classification or summarization. The adjusted similarity score was calculated using the following formula:

$$\text{AdjustedScore}_{p,c} = \text{BM25}_{p,c} - (\alpha \times U_c)$$

Where:

- $\text{AdjustedScore}_{p,c}$ : The final adjusted score between primary KB  $p$  and candidate KB  $c$
- $\text{BM25}_{p,c}$ : The original BM25 similarity score between the two KBs
- $\alpha$ : A constant controlling the penalty for overused KBs
- $U_c$ : The total number of times candidate KB  $c$  has already been used in any group

However, this penalty system also introduced a challenge. If we selected candidate KBs for all primaries in a single pass, the KBs processed earlier would get access to the most similar and popular candidates first. As a result, KBs that were processed

later in the sequence would be left with lower-scoring options, even if they originally had high similarity matches. This imbalance occurred because the penalty made overused KBs less favorable over time, especially when used too early.

To solve this, we split the candidate selection process into multiple rounds. In each round, we went through all KBs one by one and selected only one candidate group per primary KB. After finishing one full round, we moved on to the next. We repeated this process until each KB had been grouped into 20 sets. This round-based approach helped spread out the usage of high-similarity KBs more evenly, resulting in a fairer and higher-quality dataset.

### 3.2.2.3.2 Generate scenario from paired KBs

To handle longer inputs and reduce memory usage, we used Qwen2.5 14B, a smaller model that supports up to 1 million tokens. Instead of asking a large model to read all KBs and generate the conversation directly, we used this smaller model to create a scenario that connects two related KBs into a single story as seen in the figure below.

ผู้โทร: โทรเข้ามาสอบถามเกี่ยวกับการใช้บริการสร้างเสริมสุขภาพและป้องกันโรค (PP) ที่ร้านยา และอยากร้าบวิธีการลงทะเบียนเปลี่ยนหน่วยบริการผ่านแอปพลิเคชัน สปสช. เพื่อให้สามารถใช้สิทธิ์ได้อย่างสะดวก

เจ้าหน้าที่: สวัสดีค่ะ ร้านยาที่ให้บริการสร้างเสริมสุขภาพและป้องกันโรค (PP) สามารถให้บริการแก่ประชาชนสัญชาติไทยที่มีบัตรประจำตัวประชาชน 13 หลัก โดยมีบริการ 9 รายการ รวมถึงบริการตรวจสอบสุขภาพ ตรวจประเมินปัจจัยเสี่ยง และการให้คำปรึกษา ขึ้นอยู่กับอายุและกลุ่มเป้าหมายค่ะ ท่านสามารถติดตามสิทธิ์ของท่านผ่านแอปพลิเคชัน NHSO ได้เลย

ผู้โทร: แต่ถ้าหากฉันย้ายที่อยู่ไปแล้ว ฉันจะต้องทำอย่างไรเพื่อเปลี่ยนหน่วยบริการผ่านแอปฯ และของ NHSO ใช่ไหมคะ?

เจ้าหน้าที่: ใช่ค่ะ ท่านสามารถลงทะเบียนเปลี่ยนหน่วยบริการผ่านแอปพลิเคชัน NHSO ได้ โดยในกรณีที่ท่านย้ายที่อยู่ ท่านต้องเลือก \"ไม่ตรง\" เมื่อระบบถามว่าที่อยู่ปัจจุบันของท่านตรงกับที่อยู่ในบัตรประจำตัวประชาชนหรือไม่ จากนั้นท่านจะต้องถ่ายรูปบัตรประชาชนและเซลฟี่กับบัตรประชาชน พร้อมแนบหลักฐานการพักอาศัยปัจจุบัน เช่น ทะเบียนบ้านเจ้าบ้าน สัญญาเช่า หรือบิลค่าน้ำค่าไฟค่ะ

ผู้โทร: และหลังจากนั้นฉันจะต้องรอสักกี่วันถึงจะสามารถใช้สิทธิ์ได้ตามหน่วยบริการแห่งใหม่?

เจ้าหน้าที่: หลังจากที่ท่านลงทะเบียนเปลี่ยนหน่วยบริการผ่านแอปฯ แล้ว สิทธิ์ใหม่จะถูกสร้างขึ้นให้ทันที แต่เพื่อความถูกต้องของระบบ เราแนะนำให้รอประมาณ 1-2 วัน หลังจากนั้นท่านสามารถย้ายหน่วยบริการได้อีกด้วย หากจำเป็นค่ะ

ผู้โทร: ขอบคุณค่ะ และฉันจะติดตามสถานะการเปลี่ยนหน่วยบริการได้ที่ไหน?

เจ้าหน้าที่: ท่านสามารถติดตามสถานะการเปลี่ยนหน่วยบริการได้ที่แอปพลิเคชัน NBSO ภายใต้เมนู \"ตรวจสอบสิทธิ\" หรือท่านสามารถโทรติดต่อ 1330 เพื่อสอบถามสถานะการเปลี่ยนหน่วยบริการได้ที่เวลาราชการค่ะ

ผู้โทร: ขอบคุณมากค่ะ\ก\กเจ้าหน้าที่: ยินดีช่วยเหลือค่ะ ถ้ามีคำถามเพิ่มเติม ท่านสามารถติดต่อ 1330 ได้ตลอดเวลาค่ะ

**Figure 4.** Example of Conversation Generated with KB pariring and Qwen 2.5V1

This change helped avoid out-of-memory (OOM) errors and gave us much more room to work with longer KBs. Since the scenario generation task was light, it didn't need a large model — freeing up resources for the actual conversation generation later on. This setup also prepared us for future plans to include 3 or 4 KBs per conversation.

### 3.2.2.3.3 Generate conversation from the scenario

We switched to Gemini 2.0 Flash (via Google API) because it produced much better results than Qwen2.5-72B-Instruct. The conversations sounded more natural, the roleplay was smoother, and there were no issues with mixed languages or messy formatting. Gemini also provided clean and consistent structured outputs, which made post-processing much easier.

However, Gemini 2.0 comes with usage limits — it allows only 15 requests per minute and 1,500 requests per day on the free tier. Because of these limits, the generation process took more time to complete.

With this improved process, we successfully generated 7,000 entries in this version. The overall quality is much higher, with conversations that are more realistic, detailed, and human-like.

ผู้พูด1:  
สวัสดีค่ะ สายด่วน สปสช. 1330 ดิฉันอรัญญา ยินดีให้บริการค่ะ "ไม่ทราบว่าวันนี้มีอะไรให้ดิฉันช่วยเหลือค่ะ?"  
ผู้พูด2:  
สวัสดีค่ะ คือว่าดิฉันอยากระบบสอบถามเรื่องสิทธิการรักษาพยาบาลหน่อยค่ะ พอดีว่าช่วงนี้เห็นมีข่าวเรื่องร้านขายยาที่เข้าร่วมโครงการอะไรสักอย่างของ สปสช. นี่แหล่ะค่ะ แล้วก็มีบริการตรวจสุขภาพฟรีด้วย ดิฉันก็เลยสนใจค่ะ อยากจะรู้ว่ามันมีบริการอะไรบ้าง แล้วดิฉันสามารถใช้สิทธิได้ไหมคะ  
ผู้พูด1:  
ค่ะ โครงการที่คุณลูกค้าพูดถึงคือบริการสร้างเสริมสุขภาพและป้องกันโรค หรือที่เรียกว่า PP ที่ร้านขายยาค่ะ โครงการนี้เปิดโอกาสให้ประชาชนไทยที่มีบัตรประชาชน 13 หลักสามารถเข้ารับบริการต่างๆ ที่ร้านขายยาที่เข้าร่วมโครงการได้ค่ะ โดยบริการที่มีให้ก็มีหลายอย่างเลยค่ะ ตั้งแต่การตรวจตัวกรองสุขภาพเบื้องต้น การประเมินปัจจัยเสี่ยงต่างๆ ไปจนถึงการให้คำปรึกษาด้านสุขภาพโดยเภสัชกรค่ะ  
ผู้พูด2:  
อ้อ แล้วมันมีบริการอะไรบ้างคะ? แล้วดิฉันจะรู้ได้ยังไงว่าดิฉันใช้สิทธิอะไรได้บ้าง?  
ผู้พูด1:

บริการที่มีให้จะแตกต่างกันไปตามช่วงอายุและกลุ่มเป้าหมายคือ ยกตัวอย่างเช่น ถ้าคุณลูกค้าอายุ 35 ปี อาจจะสามารถรับบริการตรวจวัดความดันโลหิต หรือตรวจคัดกรองเบาหวานได้ค่ะ ส่วนการที่จะทราบว่าคุณลูกค้ามีสิทธิอะไรบ้างนั้น สามารถตรวจสอบได้ง่ายๆ ผ่านแอปพลิเคชัน สปสช. ค่ะ

ผู้พูด2:

แอป สปสช.? คือแอปอะไรหรือคะ? และต้องโหลดที่ไหน?

ผู้พูด1:

แอปพลิเคชัน สปสช. เป็นแอปพลิเคชันที่ สปสช. จัดทำขึ้นเพื่อให้ประชาชนสามารถตรวจสอบสิทธิการรักษาพยาบาลของตนเอง รวมถึงข้อมูลต่างๆ ที่เกี่ยวข้องกับระบบหลักประกันสุขภาพแห่งชาติค่ะ สามารถดาวน์โหลดได้ฟรี ทั้งใน App Store สำหรับผู้ที่ใช้โทรศัพท์ Android ค่ะ

ผู้พูด2:

อ้อ เข้าใจแล้วค่ะ แต่ว่าตอนนี้ดิจิทัลย้ายบ้านมาอยู่ที่ใหม่แล้วค่ะ และหน่วยบริการเดิมที่เคยลงทะเบียนไว้มันไม่สามารถย้ายบ้านใหม่มาได้ ถ้าจะเปลี่ยนหน่วยบริการ ต้องทำยังไงคะ? ต้องไปที่สำนักงาน สปสช. เลยหรือเปล่า?

ผู้พูด1:

ไม่ต้องไปที่สำนักงาน สปสช. ค่ะ คุณลูกค้าสามารถเปลี่ยนหน่วยบริการได้ง่ายๆ ผ่านแอปพลิเคชัน สปสช. ได้เลยค่ะ สะดวกใหม่ค่ะ เดียวดิจิทัลแนะนำขั้นตอนให้ค่ะ

ผู้พูด2:

สะดวกค่ะ รบกวนด้วยนะคะ

ผู้พูด1:

ค่ะ ขั้นตอนแรก หลังจากที่คุณลูกค้าดาวน์โหลดและติดตั้งแอปพลิเคชัน สปสช. เรียบร้อยแล้ว ให้เปิดแอปพลิเคชันขึ้นมาค่ะ จากนั้นระบบจะให้คุณลูกค้ายืนยันตัวตนก่อนนะคะ

ผู้พูด2:

ยืนยันตัวตน? ต้องทำยังไงคะ?

ผู้พูด1:

ระบบจะให้คุณลูกค้าถ่ายรูปบัตรประชาชน และถ่ายรูปเซลฟี่คู่กับบัตรประชาชนค่ะ เพื่อยืนยันว่าเป็นตัวคุณลูกค้าจริงๆ ค่ะ หลังจากยืนยันตัวตนเสร็จแล้ว จะมีเมนูต่างๆ ให้เลือกค่ะ

ผู้พูด2:

อ้อ ค่ะ แล้วต้องเลือกเมนูไหนคะ?

ผู้พูด1:

ให้คุณลูกค้าเลือกเมนู "ลงทะเบียนเปลี่ยนหน่วยบริการ" ค่ะ เมื่อกดเข้าไปแล้ว ระบบจะถามว่า "ที่อยู่ปัจจุบันของคุณตรงกับที่อยู่ในบัตรประชาชนหรือไม่" เนื่องจากคุณลูกค้าย้ายบ้านมาอยู่ที่ใหม่แล้ว ให้เลือก "ไม่ตรง" นะคะ

ผู้พูด2:

อ้อ ต้องเลือก "ไม่ตรง" ใช่ไหมคะ? และหลังจากนั้นล่ะคะ?

ผู้พูด1:

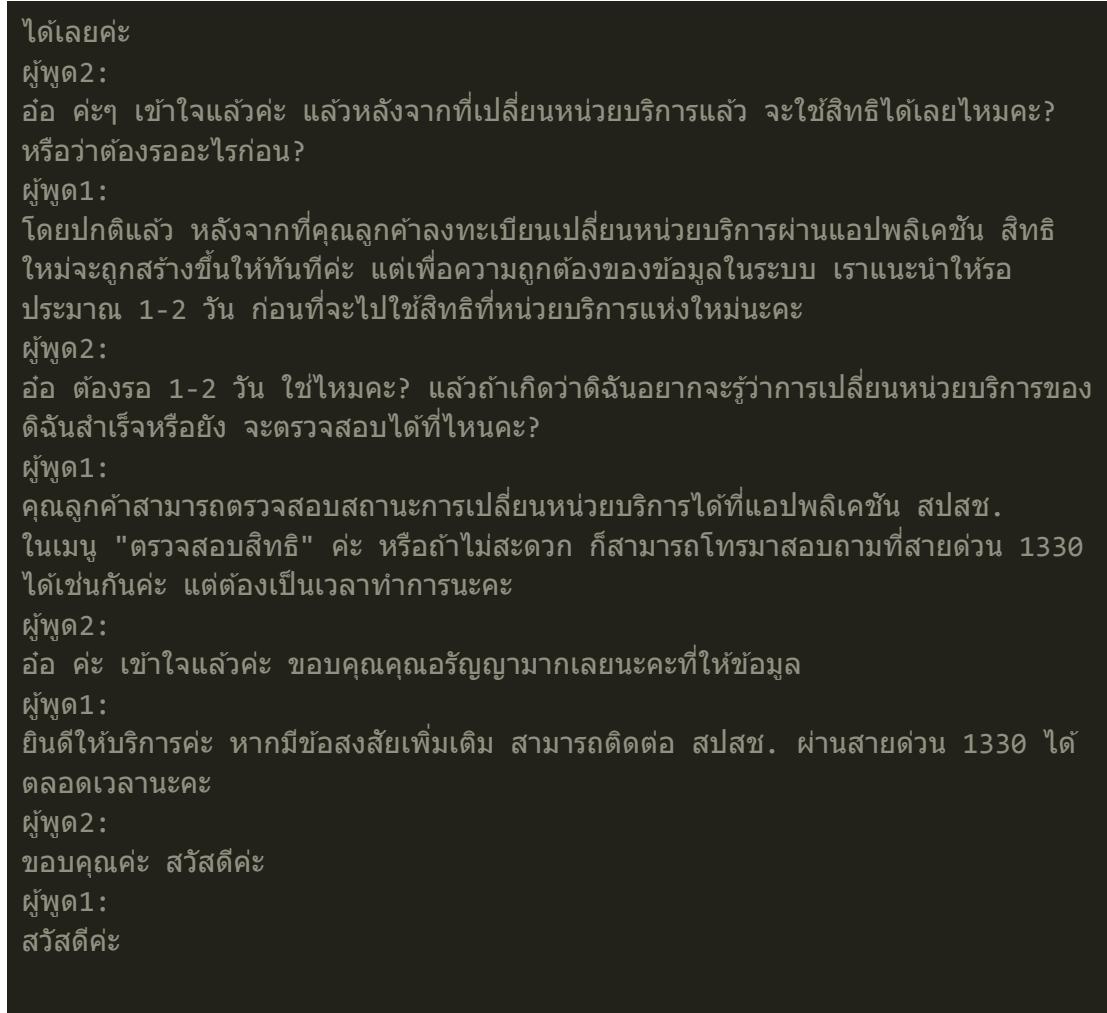
หลังจากนั้น ระบบจะให้คุณลูกค้าแนบหลักฐานการพักอาศัยปัจจุบันค่ะ เช่น ทะเบียนบ้านเจ้าบ้าน สัญญาเช่า หรือบิลค่าน้ำค่าไฟ ที่มีชื่อของคุณลูกค้าค่ะ

ผู้พูด2:

อี๊ม เข้าใจแล้วค่ะ แต่ว่าดิจิทัลยังไม่ได้เปลี่ยนทะเบียนบ้านเลยค่ะ ใช้สัญญาเช่าได้ไหมคะ?

ผู้พูด1:

ใช้สัญญาเช่าได้ค่ะ แต่ต้องเป็นสัญญาเช่าที่มีชื่อของคุณลูกค้า และมีที่อยู่ปัจจุบันระบุไว้อย่างชัดเจนนะคะ หลังจากแนบหลักฐานเรียบร้อยแล้ว ก็ให้เลือกหน่วยบริการใหม่ที่ต้องการ



**Figure 5.** Example Conversation Generated with Gemini 2.0 Flash

#### 3.2.2.4 Fourth version of synthetic dataset

In the fourth version, we continued using the same pipeline as in version 3, but this time we aimed to generate conversations using three KBs per entry instead of two. Our goal for this version was to produce more than 15,000 entries.

However, using three KBs introduced a new issue. Since each group included two candidate KBs, the one with the highest similarity score was selected repeatedly. This often resulted in the same pattern: a group made up of the primary KB, the top-scoring candidate, and one other. This happened because the previous penalty mechanism became less effective—both candidates were penalized at the same time, and the top-scoring one often dropped only from first to second place, which still kept it in the group. As a result, some high-scoring KBs appeared in over 80% of the groups for certain primary KBs.

To solve this, we introduced an additional penalty ratio, designed to reduce the score more aggressively when the same KB is repeatedly paired with the same primary. This penalty is based on the number of times a candidate KB has already been used with a specific primary KB. The more often the same pair appears, the heavier the penalty becomes—encouraging more variety in candidate selection.

The updated formula for the adjusted similarity score is:

$$\text{AdjustedScore}_{p,c} = \left( BM25_{p,c} \times \max \left( 0, 1 - \left( \beta \times \frac{T_{p,c}}{R} \right) \right) \right) - (\alpha \times U_c)$$

Where:

- AdjustedScore<sub>p,c</sub>: Final score between primary KB and candidate KB
- BM25<sub>p,c</sub>: Original BM25 similarity score between KB and KB
- T<sub>p,c</sub>: Number of times candidate KB has been grouped with primary KB in earlier rounds
- U<sub>c</sub>: Total number of times candidate KB has been used in any group
- R: Total number of rounds
- α: Global usage penalty factor
- β: Pair penalty ratio factor

The process after this followed the same steps as in the previous version. However, generating scenarios with three KBs became more challenging, as the model often forgot to include information from all KBs. This issue was gradually resolved through trial and error by carefully adjusting both the system prompt and user prompt.

Since we aimed to generate over 15,000 entries, the scenario generation process became a bottleneck—it took over 7 seconds per request, which added up to a significant amount of time. Around this time, Gemma 3 was released. Based on Gemini 2.0 but fully open-source, it offered a promising alternative. We decided to switch from using the Gemini 2.0 API to hosting Gemma-3-27B-IT ourselves on 4 A100 GPUs using APEX.

The results were impressive. While there was only a slight drop in output quality, the inference speed improved by nearly 10 times. This allowed us to scale scenario generation much more efficiently without relying on external APIs or hitting request limits.

```
-- Conversation (KB=KB36.txt, KB37.txt, KB519.txt) --
ผู้พูด1:
สวัสดีครับ ติดต่อสายด่วน สปสช. 1330 มีอะไรให้ช่วยแนะนำหรือสอบถามได้เลยครับ
ผู้พูด2:
สวัสดีครับ คือผมมีเรื่องอยากระบุกษาหน่อยครับ พอดีว่าผมอยากระบุกษาในโรงพยาบาล
ที่ใช้สิทธิบัตรทองนะครับ แต่ไม่แน่ใจว่าต้องทำยังไงบ้าง และก็ที่บ้านผมก็มีลูกกับแม่อีก
สองคน  ผมอยากระบุกษาให้ทุกคนเลย มันต้องทำยังไงบ้างครับ
ผู้พูด1:
ครับ เข้าใจแล้วครับ ไม่ทราบว่าคุณลูกค้าจะเด็กให้ผมสอบถามข้อมูลส่วนตัวของคุณ
ลูกค้าและสมาชิกในครอบครัว เพื่อตรวจสอบสิทธิและแนะนำขั้นตอนที่ถูกต้องได้ไหม
ครับ?
ผู้พูด2:
ได้ครับ คือของผมเนี่ย เลขบัตรประชาชน... (แจ้งหมายเลขบัตรประชาชน) และก็
สิทธิบัตรทองครับ ส่วนลูกชายอายุ 12 ก็มีสิทธิบัตรทองเหมือนกัน ส่วนคุณแม่เนี่ย มี
สิทธิคุณพิการด้วยครับ แต่ผมไม่แน่ใจว่าต้องใช้ข้อมูลอะไรบ้าง
ผู้พูด1:
ขอบคุณครับที่แจ้งข้อมูลเบื้องต้นให้ทราบ ไม่ทราบว่าคุณลูกค้าต้องการเปลี่ยนหน่วย
บริการด้วยเหตุผลอะไรเป็นพิเศษ?  เช่น ต้องการความสะดวกในการเดินทาง
หรือเหตุผลอื่น ๆ ครับ? และไม่ทราบว่าคุณแม่มีสิทธิคุณพิการ (DIS) ด้วยใช่ไหมครับ?
ผู้พูด2:
ใช่ครับ คือผมอยากระบุกษาในมันใกล้บ้านกว่าเดิมหน่อยนะครับ เพราะว่า
```

ตอนนี้มันเดินทางลำบาก แล้วก็เลยอย่างจะทำให้ทุกคนในครอบครัวเลยทีเดียว แล้วก็อย่างจะตามว่า สิทธิ公民การของคุณแม่เนี่ย มันจะเปลี่ยนผ่านช่องทางเดียวกันได้ไหมครับ

ผู้พูด 1:

ครับ เข้าใจแล้วครับ สำหรับการเปลี่ยนหน่วยบริการนั้น คุณลูกค้าสามารถทำได้ผ่านแอปพลิเคชัน สปสช. หรือ Line @nhso ได้ครับ สะดวกใช้ช่องทางไหนมากกว่ากันครับ?

ผู้พูด 2:

เออ... แล้วมันต่างกันยังไงหรือครับ?

ผู้พูด 1:

ทั้งสองช่องทางสามารถทำรายการเปลี่ยนหน่วยบริการได้เหมือนกันครับ แต่แอปพลิเคชัน สปสช. จะมีฟังก์ชันอื่น ๆ เพิ่มเติม เช่น การตรวจสอบสิทธิ การค้นหาหน่วยบริการ หรือการรับข่าวสารจาก สปสช. ครับ ส่วน Line @nhso จะเน้นที่การทำธุกรรมที่รวดเร็วและง่ายครับ

ผู้พูด 2:

อืม... ยังลองทำในแอปดูก่อนก็ได้ครับ

ผู้พูด 1:

ได้ครับ สำหรับขั้นตอนการลงทะเบียนและเปลี่ยนหน่วยบริการผ่านแอปพลิเคชัน สปสช. มีดังนี้ครับ ขั้นตอนแรก ให้คุณลูกค้าดาวน์โหลดแอปพลิเคชัน สปสช. จาก Google Play Store หรือ App Store ก่อนนะครับ โดยพิมพ์คำว่า "สปสช." ในช่องค้นหาครับ

ผู้พูด 2:

อ่อ โอดีครับ เดี๋ยวผมลองหาดูก่อน

ผู้พูด 1:

เมื่อดาวน์โหลดแอปพลิเคชันเรียบร้อยแล้ว ให้คุณลูกค้าเปิดแอปพลิเคชัน และคลิกที่เมนู "ลงทะเบียน" นะครับ

ผู้พูด 2:

ครับฯ เห็นแล้วครับ ปุ่มลงทะเบียน

ผู้พูด 1:

จากนั้น ระบบจะให้คุณลูกค้ากรอกข้อมูลส่วนตัว เช่น หมายเลขบัตรประชาชน วันเดือนปีเกิด และเลขหลังบัตรประชาชนครับ

ผู้พูด 2:

โอดีครับ เดี๋ยวผมกรอกตาม

ผู้พูด 1:

หลังจากการกรอกข้อมูลส่วนตัวเรียบร้อยแล้ว ระบบจะให้คุณลูกค้าตั้งรหัส PIN 6 หลัก เพื่อใช้ในการเข้าสู่ระบบครั้งต่อไปครับ

ผู้พูด 2:

อ่อ ตั้งรหัสผ่านไข่ใหม่ครับ

ผู้พูด 1:

ใช่ครับ เมื่อตั้งรหัส PIN เรียบร้อยแล้ว ระบบจะให้คุณลูกค้ายืนยันตัวตน โดยการถ่ายรูปบัตรประชาชน และถ่ายรูปเซลฟี่คู่กับบัตรประชาชนครับ

ผู้พูด 2:

อืม... ต้องถ่ายรูปด้วยหรือครับ?

ผู้พูด 1:

ใช่ครับ เพื่อเป็นการยืนยันตัวตนของคุณลูกค้าครับ เมื่อยืนยันตัวตนเรียบร้อยแล้ว คุณลูกค้าจะสามารถเลือกหน่วยบริการใหม่ที่ต้องการได้เลยครับ

ผู้พูด 2:

อ้อ เข้าใจแล้วครับ แล้วของคุณแม่ที่ถือสิทธิ公民พิการนี่ต้องทำยังไงครับ?

ផែទ 1:

สำหรับคุณแม่ที่มีสิทธิคนพิการ (DIS) จะไม่สามารถเปลี่ยนหน่วยบริการผ่านแอปพลิเคชัน หรือ Line @nhso ได้นะครับ เนื่องจากสิทธินี้ไม่รองรับการลงทะเบียนผ่านช่องทางดังกล่าวครับ

៤២៩

ອ້າວ ເຫຣອຄຮັບ ແລ້ວຕ້ອງທໍາຍັງໄຟລ່ຄຮັບ?

៨១

คุณแม่จะต้องติดต่อหน่วยบริการใกล้บ้านโดยตรง เพื่อดำเนินการเปลี่ยนหน่วยบริการครรภ์โดยนำบัตรประชาชนและเอกสารที่เกี่ยวข้องไปติดต่อที่หน่วยบริการได้เลยครับ

ຜົດ 2:

อ้อ เช้าใจแล้วครับ แล้วของลูกชายอายุ 12 นี่ทำผ่านแอปได้เลยใช่ไหมครับ?

៤១

ใช้ครับ สำหรับบุตรชายอายุ 12 ปี สามารถทำผ่านแอปพลิเคชัน สปสช. ได้เลยครับ โดยใช้ข้อมูลของบุตรชายในการลงทะเบียนและยืนยันตัวตนครับ หลังจากยืนยันข้อมูล เรียบร้อยแล้ว สิทธิการรักษาพยาบาลใหม่จะมีผลทันทีครับ

សំណើលេខ ២:

ခုခံ ကရာပ်၊ သော အဲလောကရာပ် ခုခံကရာမာကနီရာပ်

ផែទេស

ยินดีให้บริการครับ หากมีข้อสงสัยเพิ่มเติม สามารถติดต่อสายด่วน 1330 หรือแอด Line @nhso ได้ทุกวันทำการนะครับ

ធនធាន ២:

ครับ ขอบคุณมากครับ สวัสดีครับ

ผู้ดูแล: สวัสดีครับ

**Figure 6.** Example Scenario Generated Using 3 KBs with Gemma 3'

### 3.2.2.5 Fifth version of synthetic dataset

In the fifth version, we generated conversations using four KBs per entry, producing a total of 6,500 entries. The overall process remained the same as in the previous version, including KB pairing, scenario generation, and conversation synthesis.

However, generating scenarios with four KBs introduced a new challenge: the combined context length often reached around 50,000 tokens, which made the scenario generation process significantly slower—even when using the smaller Qwen2.5 14B (1M) model. Despite its smaller size, Qwen was still slow due to poor inference optimization.

To address this, we experimented with Gemma-3 12B IT, which we found to be better optimized for inference. As a result, we were able to generate scenarios twice as fast compared to Qwen, without sacrificing output quality. This speed improvement allowed us to complete scenario generation in a more reasonable timeframe, even when working with large input contexts.

Conversation (KB=KB36.txt, KB984.txt, KB1061.txt, KB73.txt)

ຝັ້ມ 1

ស្ថាបន្ទី សាស្ត្រ សាធារណៈ សាខាអាជីវកម្ម សាខាអាជីវកម្ម សាខាអាជីវកម្ម សាខាអាជីវកម្ម

ຟັບ 2:

สวัสดีค่ะ คือว่า...เอ่อ...คือหนูเพิ่งรู้ตัวว่ากำลังตั้งครรภ์ค่ะ แล้วก็อยากรจะไปฝากครรภ์ที่โรงพยาบาลใกล้บ้านมากกว่าที่เดินนี่ค่ะ แต่พอเปิดแอป สปสช. ดู มันไม่เห็นโรงพยาบาลที่หุ้นอย่างไว้เลยค่ะ ไม่รู้ว่าต้องทำยังไงดี

ผู้พูด1:

ติณัณเข้าใจค่ะ "ไม่ทราบว่าคุณลูกค้าส่วนใดได้ตั้งส่วนใดมูลส่วนตัวบ้าง" ใหม่ค่ะ เพื่อตรวจสอบสิทธิและให้คำแนะนำที่ถูกต้องได้ค่ะ?

ผู้พูด2:

อืม...ก็ได้ค่ะ คือหนูชื่อ...เอ่อ...ชื่อนารีรัตน์ ศรีวิไลค่ะ

ผู้พูด1:

ขอบคุณค่ะคุณนาเรียรัตน์ ติณัณขออนุญาตตรวจสอบข้อมูลสักครู่นึงค่ะ...ค่ะ จากการตรวจสอบ คุณนาเรียรัตน์มีสิทธิหลักประกันสุขภาพแห่งชาติถูกต้องนะค่ะ "ไม่ทราบว่าโรงพยาบาลที่คุณนาเรียรัตน์ต้องการเปลี่ยนไปฝากครรภ์คือโรงพยาบาลใดค่ะ?

ผู้พูด2:

คือโรงพยาบาลสมเด็จพระนราธิราชนนครศรีธรรมราชค่ะ มันใกล้บ้านหนูมากเลย แต่ในแอปมันไม่ขึ้นมาให้เลือกเลยค่ะ

ผู้พูด1:

ติณัณเข้าใจค่ะ บางครั้งข้อมูลในแอปพลิเคชันอาจยังไม่ได้รับการอัปเดต หรืออาจมีข้อจำกัดบางประการค่ะ ก่อนอื่นคุณนาเรียรัตน์ลองตรวจสอบอีกครั้งนะค่ะ ว่าพิมพ์ชื่อโรงพยาบาลถูกต้องหรือไม่ และลองเลื่อนดูในรายการทั้งหมดแล้วหรือยังค่ะ?

ผู้พูด2:

ก็ลองแล้วค่ะ พิมพ์ก็ถูกต้องแล้ว เลื่อนดูก็แล้วค่ะ ก็ไม่เห็นเลยค่ะ แล้วหนูจะทำยังไงดีค่ะ?

ผู้พูด1:

ถ้าไม่พบโรงพยาบาลที่ต้องการในแอปพลิเคชัน คุณนาเรียรัตน์สามารถติดต่อโรงพยาบาลสมเด็จพระนราธิราชนนครศรีธรรมราชโดยตรงได้เลยค่ะ เพื่อสอบถามว่าโรงพยาบาลได้ลงทะเบียนเข้าร่วมในระบบหลักประกันสุขภาพแห่งชาติแล้วหรือยังค่ะ?

ผู้พูด2:

อืม อย่างนั้นเหรอค่ะ? แล้วถ้ามันลงทะเบียนแล้วละคะ?

ผู้พูด1:

หากโรงพยาบาลลงทะเบียนแล้ว แต่ยังไม่ปรากฏในแอปพลิเคชัน คุณนาเรียรัตน์สามารถติดต่อเจ้าหน้าที่แอดมินของ สปสช. ผ่านช่องทาง Line @nhso (พื้นที่ กทม.) หรือติดต่อโรงพยาบาลส่งเสริมสุขภาพตำบล (รพ.สต.) หรือโรงพยาบาลรัฐ (พื้นที่ ต.จว.) เพื่อดำเนินการเรื่องการเปลี่ยนหน่วยบริการได้ค่ะ

ผู้พูด2:

อืม...แล้วต้องใช้เวลานานไหมค่ะ กว่าจะเปลี่ยนได้? แล้วหนูจะฝากครรภ์ได้เลยไหมค่ะ หลังจากเปลี่ยนหน่วยบริการแล้ว?

ผู้พูด1:

โดยทั่วไปแล้ว หลังจากดำเนินการเปลี่ยนหน่วยบริการแล้ว สิทธิใหม่จะเริ่มมีผลทันทีค่ะ แต่เพื่อความแน่ใจ แนะนำให้คุณนาเรียรัตน์รอประมาณ 1-2 วันทำการ เพื่อให้ระบบประมวลผลเรียบร้อยก่อนนะค่ะ และจึงค่อยไปติดต่อโรงพยาบาลสมเด็จพระนราธิราชนนครศรีธรรมราชเพื่อฝากครรภ์ค่ะ

ผู้พูด2:

อืม...โอดีค่ะ และเรื่องสามีหนูบ้าง สามีหนูอยากรจะไปฟังยาคุมค่ะ ต้องทำยังไงบ้างค่ะ? หนูสามารถลงทะเบียนให้สามีได้ไหม?

ผู้พูด1:

ได้ค่ะ คุณนาเรียรัตน์สามารถลงทะเบียนให้สามีได้ผ่านแอปพลิเคชัน สปสช. ค่ะ โดยมี

ข้อกำหนดว่า สามีของคุณนารีรัตน์จะต้องมีนามสกุลและที่อยู่ตามทะเบียนบ้านเดียวกัน กับคุณนารีรัตน์นั่นแหละ

ผู้พูด2:

อ่อ อย่างนั้นหรือคะ? แล้วต้องใช้ออกสารอะไรบ้างคะ?

ผู้พูด1:

ในการลงทะเบียนให้สามี คุณนารีรัตน์จะต้องมีบัตรประจำตัวประชาชนของสามี และสามี สมุดบันทึกสุขภาพแม่และเด็ก ก็สามารถเตรียมไปได้ด้วยได้ค่ะ แต่การลงทะเบียนผ่านแอป อาจมีข้อจำกัดนะค่ะ หากไม่สามารถลงทะเบียนได้ คุณนารีรัตน์สามารถติดต่อที่จุดรับลงทะเบียนได้ค่ะ

ผู้พูด2:

แล้วจุดรับลงทะเบียนอยู่ที่ไหนบ้างคะ?

ผู้พูด1:

สำหรับพื้นที่กรุงเทพฯ สามารถติดต่อแอดมินไลน์ @nhro ได้เลยค่ะ ให้บริการทุกวัน ตั้งแต่เวลา 8 โมงเช้าถึง 2 ทุ่มค่ะ ส่วนพื้นที่ต่างจังหวัด สามารถติดต่อที่โรงพยาบาล ส่งเสริมสุขภาพตำบล หรือโรงพยาบาลรัฐใกล้บ้านได้ค่ะ แต่ต้องเป็นวันและเวลาราชการ นะค่ะ

ผู้พูด2:

อืม...เข้าใจแล้วค่ะ แล้วหลังจากลงทะเบียนให้สามีแล้ว สามีหนูจะไปฝังยาคุมได้เลย ไหนคะ? ต้องเตรียมอะไรไปบ้าง?

ผู้พูด1:

การฝังยาคุมเป็นบริการที่อยู่ในสิทธิ BCB ค่ะ ดังนั้น สามีของคุณนารีรัตน์สามารถเข้ารับบริการได้ทันทีหลังจากลงทะเบียนเรียบร้อยแล้วค่ะ สิ่งที่ต้องเตรียมคือ บัตรประจำตัวประชาชนของสามีค่ะ และควรโทรศัพท์นัดหมายกับทางโรงพยาบาลล่วงหน้า เพื่อความสะดวกนะค่ะ

ผู้พูด2:

ขอบคุณมากเลยนะค่ะที่ให้คำแนะนำละเวียดขนาดนี้ค่ะ

ผู้พูด1:

ยินดีให้บริการค่ะ หากมีข้อสงสัยเพิ่มเติม หรือต้องการให้ดีฉันติดตามเรื่องการเปลี่ยนหน่วยบริการให้ ก็สามารถติดต่อกันได้ตลอด 24 ชั่วโมงนะค่ะ ดีฉันจะช่วยติดตามสถานะให้ค่ะ

ผู้พูด2:

ขอบคุณค่ะ เดียวหนูจะลองทำตามที่บอกนะค่ะ สวัสดีค่ะ

ผู้พูด1:

สวัสดีค่ะ ขอให้คุณนารีรัตน์และครอบครัวมีสุขภาพแข็งแรงค่ะ

*Figure 7. Example Scenario Generated Using 4 KBs*

### 3.2.2.6 Sixth and Seventh version of synthetic dataset

In the sixth version, we generated 6,260 entries using two KBs per conversation. This version was created to help balance the overall dataset, as the number of 2-KB entries was still lower than the intended ratio compared to other variants. The entire pipeline remained the same as in previous versions—using BM25 for KB pairing, adjusted similarity scoring with penalty mechanisms, scenario generation, and conversation synthesis. No structural changes were made; this version served primarily to fill the gap and ensure better distribution across KB group sizes.

The seventh version focused on single-KB conversations, generating a total of 7,620 entries. These conversations are meant to simulate more focused, real-world inquiries where users

ask about one topic at a time. Since only one KB was used per entry, the pairing step was removed, simplifying the process. This version adds important variety to the dataset by offering a mix of short, topic-specific conversations alongside the more complex multi-KB interactions.

### 3.2.2.7 Merge Datasets

After completing all versions from v1 to v7, we merged the datasets into a single unified set—excluding version 2, which originally contained 21,240 entries. While version 2 played an important role in scaling up the dataset early on, its overall quality was noticeably lower compared to later versions. To maintain consistency, we included only 7,000 carefully selected entries from version 2 in the final dataset. These entries were chosen based on the appearance frequency of each KB—favoring underrepresented KBs to ensure better balance and reduce the standard deviation in KB usage.

## 3.3 Model Pipeline Development

### 3.3.1 Speech To Text model

#### 3.3.1.1 Research about STT model

Through our research, we found various STT models such as Whisper, PyThaiNLP, AWS Transcribe, and Google Speech Recognition. Each model has unique strengths and limitations. Whisper is great for multiple languages, especially with its Thai-focused versions, Whisper TH [34], WhisperX [35], and Monsoon Whisper []. PyThaiNLP is good for Thai language tasks but has limited STT features. AWS Transcribe and Google Speech Recognition are reliable, cloud-based options that work well for general transcription but they are very expensive.

#### 3.3.1.2 Evaluation Criteria

We prioritized four critical factors for our STT models: accuracy, processing time, speaker diarization support, and cost.

##### 3.3.1.2.1 Accuracy

To assess the transcription accuracy of each model, we used Word Error Rate (WER), a metric that quantifies the differences between the model's output and the correct transcription. WER accounts for insertions, deletions, and substitutions, with a lower value indicating higher accuracy. Among the evaluated models, Monsoon Whisper consistently achieved the lowest WER, making it the most accurate model for Thai language speech recognition. It outperformed other alternatives, including Whisper TH.

##### 3.3.1.2.2 Processing Speed

The speed of the model processing the audio is very important for large datasets. We evaluated the model using audio files ranging from 5 seconds to 50 minutes. We measured the time by starting the clock before loading the Speech-To-Text models and ending the time after the model had fully transcribed the text.

##### 3.3.1.2.3 Speaker Diarization

Speaker diarization is very important for our STT models. It is the ability to differentiate between multiple speakers. The current models that support speaker diarization are AWS Transcribe, WhisperX, and Google Speech Recognition.

##### 3.3.1.2.4 Cost

Cost is a huge factor in our Speech-To-Text model. Since we want NHSO to minimize expenses while maintaining high-quality transcription services, we

prioritize models that are free to operate. By using open-source models like Whisper and their fine models, NHSO can save a significant amount of money. These models can be scaled to handle high-volume processing efficiently without requiring extra costs for each transcription.

### Google Speech Recognition [37]

Standard recognition models					
Category	Model	0 minute to 500,000 minute	500,000 minute to 1,000,000 minute	1,000,000 minute to 2,000,000 minute	2,000,000 minute and above
Recognition (sku:3099-B70F-0949)	Standard	\$0.016 / 1 minute, per 1 month / account	\$0.01 / 1 minute, per 1 month / account	\$0.008 / 1 minute, per 1 month / account	\$0.004 / 1 minute, per 1 month / account
Recognition (Logged) (sku:4292-8666-5DBB)	Standard	\$0.012 / 1 minute, per 1 month / account	\$0.0075 / 1 minute, per 1 month / account	\$0.006 / 1 minute, per 1 month / account	\$0.003 / 1 minute, per 1 month / account
Medical models					
Category	Model	0 minute to 60 minute		60 minute and above	
Medical Dictation (sku:6649-62EF-CB8F)	Medical <sup>2</sup>	\$0 (Free) / 1 minute, per 1 month / account		\$0.078 / 1 minute, per 1 month / account	
Medical Conversation (sku:7247-19E1-FB4D)	Medical <sup>2</sup>	\$0 (Free) / 1 minute, per 1 month / account		\$0.078 / 1 minute, per 1 month / account	

Whisper (Including Whisper TH, WhisperX, Monsoon Whisper) and PythaiNLP are free.

#### 3.3.1.3 Model Information

Model Name	Parameters Size	Model Size	Required VRAM
Whisper Large [38]	1550 M	5 GB	~10 GB
Whisper TH	1610 M	3.22 GB	~8 GB
WhisperX	~1550 M	~5 GB	~10 GB
Monsoon Whisper	764 M	3.06 GB	~6 GB
PythaiNLP	105 M	419 MB	~4 GB
Google Speech Recognition	Undisclosed	Undisclosed	Undisclosed

##### 3.3.1.3.1 Implementation Details of WhisperX

WhisperX was integrated into the system with GPU acceleration enabled through APEX. We use the large model for maximum accuracy and performance. Audio files were preprocessed by converting them to a compatible format such as WAV, and cleaning them to reduce background noise and distortion. To effectively process

audio with multiple speakers, WhisperX was paired with PyAnnote’s speaker diarization pipeline, which greatly improved its speaker identification capabilities. Finally, the transcribed outputs were post processed to correct minor errors and format the text for better readability. However, we tried to run WhisperX on apex and it won’t run due to the mismatch between the version of cuda and whisper itself.

### 3.3.1.3.2 Implementation Details of Monsoon Whisper

We decided to opt for an alternative model that is also fine tuned for Thai language. The test of the result model shows the best results for our use case with fast transcription speed and high accuracy.

### 3.3.1.3.1 Audio Datasets Used for Testing

We used real-world data relevant to call center operations, including recordings from podcasts and public health conversations. Additionally, we leveraged data from the [ZeroSickTH](#) [39] hugging face [thaigov-radio-audio](#) [40], and [thai-elderly-speech](#) [41], which offers diverse audio content for testing. This data will help evaluate the speaker diarization and transcription accuracy of our models, ensuring they perform effectively in varied and noisy environments typically found in real call center scenarios.

### 3.3.1.4 VLLM with STT models

We tested the VLLM library on both Whisper and Whisper TH to evaluate potential performance improvements. Additionally, we segment audio files into smaller chunks and use batch processing to transcribe them efficiently.

**Generated transcription:** ประเทศไทยเรานี่ย พดถึงเรื่องของน้ำหนักเกินอะ ในอาเซียนเนี่ย เราได้ระดับสองนะ ระดับสองเลยหรือ รองจากมาเลเซียอะ เพราะว่ามาระเขี้ยวอ้วนที่สุด แล้วต่อมาก็ คือเรา ต่อมาก็คือเราแล้ว คนม้าหนักเกินนะ แล้วพอรามาดูประชากรของคนไทยที่มีภาวะน้ำหนักเกินเนี่ย อุญี่ที่ 32% ของประชากร ก็หมายถึงว่าคนไทย 70 ล้านคน 32% นี่คืออ้วน ไม่ๆ 32% เนี่ยคือน้ำหนักเกิน อ่อนน้ำหนักเกิน แต่ 9% อะเป็นโรคอ้วน 9% เป็นโรคอ้วน นอกจานั้นเนี่ย โรคเบาหวาน

**Generated transcription:** มันมากับโรคอ้วนเนี่ย เมื่อปี 2000 คนไทยเป็นโรคเบาหวาน ประมาณ 1.5 ล้านคน แต่ตอนนี้มีให้หายคนเป็นเบาหวานกี่คน ค่าค้าที่แล้ว 1.5 ล้านคน อ่ะ กี่เท่าดี ขึ้นกี่เท่า ประมาณ 4-5 เท่าได้ปะ 8.5 ล้านคน ในปัจจุบัน โอ๊荷 เยอะมากนะ เยอะมากเนอะ เพราะว่า 7-8 เท่า เพราะว่าฉะนั้นเนี่ยโรคอ้วนทำไม่ได้สักที จริงๆก็เป็นคำเตือนที่นำเสนอใจ เพราะว่าจริงๆแล้วปัจจุบันเราก็แบบมีเทคโนโลยีช่วยเหลือและพยายามนะ แต่ว่าคนไข้ก็อยาก

**Generated transcription:** เรื่องมหาเราระยะอุญี่ดี เออ บอกว่า มหาว่า เออ หมอดะ ช่วยลดน้ำหนักหน่อยค่ะ บุนนี่นั่น อะไรมาย่างเงี้ย เออ ยังเยอะอุญี่ จริงๆแล้วเนี่ย ก็คือก่อนอื่นอื่นเลยนะ เราต้องมาเข้าใจว่า ความทิวต่ออะไรก่อน ก็คือว่า ความทิวอะ มันเป็นเกี่ยวข้องกับ ต้องเข้าใจว่า ร่างกายคณเรามันค่อนข้างซับซ้อน ใชปะ เรายังทึ้งสาร เคเมียอะและเลย แล้วก็มีไซโนนด้วย แล้วก็โดยส่วนมากอะ ความทิวอะ เกี่ยวข้องกับไซโนนของเราโดยตรงเนอะ เมื่อเรารู้ว่า เมื่อห้องเราวางใชปะ แล้วก็แบบไม่มีอาหารอย่างเงี้ย เราก็จะเริ่มนี่เสียงแบบ กูกก กูกก กูกก

**Generated transcription:** เริ่มทิวแล้ว เริ่มทิวแล้วมันอยากกิน อันนี่เกิดขึ้นจากไซโนนที่ ชื่อว่าเกรลิน เกรลินเป็นไซโนนที่เข้าเรียกว่าอะไร ผลิตจากในพวงกล้าไส์ กระเพาะอาหารด้วย

กระเพาะอาหาร ล่าใส่ต่างๆ มันก็จะส่งสัญญาณไปที่สมอง แล้วบอกว่าเข้ย เออไม่มีอาหารแล้วนะ เออไม่มีแคลอรี่แล้วนะ ฉันต้องการกิน และก็เลยรู้สึกว่า เข้ย หิวเวย หิวต้องกิน พอกินเข้าไป

ประมาณนึง ที่มันจะบอกว่าเริ่มอิ่ม มันก็จะเริ่มส่งสัญญาณว่า เข้ยๆ ไข้มันเราจะผลิตไซโนนอะไร

**Generated transcription:** ที่ชื่อว่าเลปดินออกมา ตรงกันข้ามกัน ตรงกันข้ามกัน ไซโนน เลปดินนี่จะเป็นไซโนนให้เราแบบบว่า เข้ย อิ่ม勃勃 พอกิน ก็จะแล้วนะ เออ ไม่เงินเราจะกิน กินแบบไม่มีวันหยุดใชปะ จะนั้นไซโนนสองตัวนี้มีเป็นไซโนนที่ว่า เออ หิวนะ แล้วก็ไซโนนที่ว่า เออ อิ่มได้แล้วนะ เออ ชึ่งในงานวิจัยอะกูพนเจอว่าคนปัจจุบันเนี่ยมีการตื้อต่อเลปดินค่อนข้าง

เย่อ เหมือนคล้ายๆดื้อต่ออินซูลิน คนที่เป็นเบาหวาน ใช่ แต่มีการดื้อ ต้อเลปตินแทน ใช่ ก็แบบว่าเลปตินอาจจะสร้างแล้วนะ แต่ก็ยังหิวอยู่ ซึ่งอาจจะเจอ

Generated transcription: มากับคนที่เป็นโรคอ้วนจริงจัง แบบว่ากินแล้วแต่ยังกินอีกจะ  
เพราะว่า yังกินต่อไม่หยุด อะไรอย่างเช่น ก็ เพราะว่า สอร์โมนเลปตินจะ เข้ามีบัญชา แล้วบอกจาก  
สอร์โมนสองตัวแล้วเนี่ย ยังมีสอร์โมนอีกสองตัวที่สำคัญมาก ก็คือสอร์โมนอินซูลินที่เรารู้ไปก่อน  
หน้านี้เน้อ แล้วก็สอร์โมนคอร์ติโซล สอร์โมนคอร์ติโซลเนี่ยเป็นสอร์โมนความเครียดใช่ปะ แต่ว่ามัน  
ก็เป็นนาสันใจว่า เออทำไม่แบบตอนกลางคืนอะ เรานอนอะ เรานอนประมาณหกถึงแปดชั่วโมงต่อวัน

ใช้ปะ แล้วทำไม่เราะแบบ ท่าไม่ระหว่างหกถังแปดชั่วโมงถังไม่กิน อาใช้เพระว่าเราอยู่กลางวัน  
Generated transcription: ยัง 6-8 ชั่วโมง ไม่กินนี่คือหิวมาก ไม่ต้อง 6-8 2-3 ชั่วโมงก็  
กินแล้ว ไม่ต้อง 6-8 แต่คือเมื่อเวลาเรานอนอย่างนี้ เราก็ไม่รู้สึกหิวใช่ปะ เขาก็มาทำงานวิจัยแล้ว  
ว่าเวลาเรานอนเนี่ย เราแม่ก็ต้องรู้โน่นที่สูงนະ ที่เราก็ให้ 4 ทุ่มถังตี 2 อยู่ต้องนอน พะว่าพอเรา  
นอนแล้วเนี่ย เราก็จะมีก็ต้องรู้โน่นหลัง ก็ต้องรู้โน่นเราก็จะไม่ค่อยเห็น แล้วพอเราตื่นขึ้นมาตอน  
เช้าเนี่ย ประมาณ 7-8 โมงเนี่ย ก็จะเป็นช่วงที่สอร์โนนคอร์ติโซ่สูงสุด ซึ่งพ่อสอร์โนนคอร์ติโซ่สูงจะ  
ทำความบันยันคลุ่มหัวงง

Generated transcription: ເພຣະນຳຕາລອຄ່ອນຫ້າງສູງ ເຮາລຍໄປ່ເທິວ

Duration: 63.1507465839386

### 3.3.1.5 Fine Tuning Speech to text model

3.3.1.5 Fine Tuning Speech to text model  
From our extensive research we decided to use a fine tuned model of whisper called monsoon whisper. It's a model of whisper that has been trained with thai voice datasets. We chose this model due to the time efficiency and the accuracy of the transcription. However, we believe that the model can be further fine tuned to increase its performance. We used CMKL/Porjai-Thai-voice-dataset-central as the dataset to train our model. After fine tuning the model, we tested it out on a small sample data that consists of audio that last from (30-5 minutes) to test the accuracy between the models and the fine tuned version performs better with 4% less word error rate

Audio	Distilled Whisper TH	Monsoon	Fine Tuned Monsoon
Medical Audio1	0.3822	0.3337	0.3222
Medical Audio2	0.3005	0.2989	0.3217
Medical Audio3	0.2488	0.283	0.2146
Medical Audio4	0.3157	0.3337	0.3333
TestAudio1	0.1299	0.0909	0.1299
TestAudio2	0.2222	0.1852	0.2037
TestAudio3	0.1447	0.1447	0.0263
TestAudio4	0.1818	0.0364	0.0
TestAudio5	0.3611	0.0972	0.25
TestAudio6	0.1047	0.1395	0.0465
TestAudio7	0.1042	0.1042	0.0625
TestAudio8	0.0678	0.2373	0.0847
TestAudio9	0.0676	0.1081	0.0676
TestAudio10	0.122	0.0854	0.061
v2TestAudio1	0.2824	0.3294	0.3294
v2TestAudio2	0.2613	0.1532	0.1441
v2TestAudio3	0.1471	0.1275	0.1373
v2TestAudio4	0.4478	0.194	0.2687
v2TestAudio5	0.4074	0.6852	0.2778
Average	0.2274	0.2033	0.1602

### 3.3.1.6 Speaker Diarization

We used the PyAnnote library for speaker diarization, which segments audio and groups portions spoken by the same speaker. To evaluate the model's performance, we used Diarization Error Rate (DER), a metric that measures the accuracy of speaker segmentation. The model performed well in differentiating speakers with distinct voices but had a higher DER when the voices were very similar, indicating challenges in segmenting and correctly identifying speakers in such scenarios.

### 3.3.1.7 Fine Tuning Speaker Diarization

To improve the performance of the diarization model, we generated synthetic conversation data consisting of roughly 10,000 audio files from [THAI SER](#) and created matching RTTM and UEM files for training and evaluation. These synthetic conversations were created by systematically combining audio clips from different actors across studio and Zoom recordings, arranging them in alternating order to simulate two speakers talking to each other. The code collects audio files from multiple recording sessions (studio21-80 and zoom1-20), organizing them by actor and script, then ensuring each actor has all three required script recordings before pairing them for conversation synthesis. We then fine-tuned the model using the dataset. This fine-tuning led to notable improvements in performance, but we observed potential overfitting to the synthetic patterns. As a result, we opted to use the original PyAnnote model for deployment to ensure better generalization on unseen data.

### 3.3.1.8 Connecting Speaker Diarization and Speech To Text Model

To connect the speaker diarization model with the speech-to-text (STT) model, the diarization system first identifies and segments the audio into distinct speaker portions. Each segment is labeled with a speaker identifier. The transcribed text from the Monsoon Whisper STT model is then aligned with these speaker segments. This process involves associating each transcription output with the corresponding speaker, based on the diarization results. By doing so, we ensure that the transcription is accurately linked to the correct speaker for each segment, improving the overall clarity and structure of the final output.

## 3.3.2. Topic Classification *Niracha Janavatara*

### 3.3.2.1 Objective and success criteria

Our objective is to retrieve the top 10 most relevant Knowledge Base (KB) entries for a given call transcript immediately after the call ends. Unlike a streaming or real-time setup, our implementation performs batch retrieval after the full transcript has been generated from the MP3 recording.

Each transcript may be associated with 1–5 KBs (sometimes more), depending on the topics discussed. While precision matters, our top priority is multi-label recall, especially Recall@3 and Recall@5.

#### Evaluation Targets

Metric	Target	Rationale
Exact KB Recall@3 for 1 ground truth KBs	$\geq 95\%$	Ensure critical KBs are among the first few.
Exact Recall@5 for 1-3 ground truth KBs		
Exact Recall @10 for 1-4 (or more) ground truth KBs		
Token IoU (KB $\leftrightarrow$ Transcript)	$\geq 0.80$	Ensures semantic match, even across paraphrased or partial KB matches
Latency (transcript $\rightarrow$ KB results)	$\leq 2\text{s P95}$	So that the agents are able to feel that KB classification is almost instantaneous, and they still remember as much of the call as possible to fact check the KBs.

We optimize not only for top-1 accuracy, but also for multi-label coverage, since many conversations involve 2–5+ KBs. As such, our evaluation pipeline computes both Recall@K and Token-IoU for multiple KB tags per query.

### 3.3.2.2. Dataset Preparation: KBs, Transcripts and Ground Truth

#### 3.3.2.2.1 Knowledge Base Documents

The KB corpus comprises **468 documents** ( $\approx 9$  million tokens). Each `.txt` file contains structured health and policy information but only the KB ID is embedded in the filename (e.g., `KB1346.txt`). As such:

1. Regex-based parsing was implemented to extract the title and name of each KB.
2. The final KB entry format was standardized to JSONL:

```
{ "kb_id": "KB1346", "title": "ตรวจสายด้าดีก", "body": "...", "updated_at": "..." }
```

#### 3.3.2.2.2 Synthetic Conversations (v1–v7)

- 57,142 conversations generated by prompt templates.
- Each conversation is labeled with 1 to 5 ground truth KB IDs ("kb\_pair" field in JSON).
- Transcripts include turns like:

ผู้พูด00: สวัสดีค่ะ อยากรถเปลี่ยนหน่วยบริการ  
ผู้พูด01: ค่ะ ขอทราบเลขบัตรประชาชน...

#### 3.3.2.2.3 Ground Truth Labels for Evaluation

- Each query's label is a set of relevant KB IDs.
- During evaluation, we score a hit if any chunk from the correct KB is retrieved.

### 3.3.2.3 KB Normalization and Preprocessing

Text Cleaning Steps:

1. DOCX Artifacts Removal – Redundant line breaks, bullets, hidden tags like `<<TABLE>>`, `<<IMG>>`.
2. Unicode Normalization (NFKC) – Ensures Thai diacritics are consistently rendered.
3. Title Extraction (Regex) – The first bold or uppercase line matching Thai sentence structure is set as the title.
4. Gemini 2.0 Flash for OCR/IMG KBs – Used for any visual PDF-style entries. Generates fluent Thai with diagrams described textually.
5. All entries were finalized in versioned `.jsonl` and uploaded to MinIO

### 3.3.2.4 Transcript Preprocessing and Chunking

Cleaning Process:

1. **Turn Splitting**

regex: `\d+`: used to separate speaker turns.

2. **Removal of trivial prefixes**

greetings, hold messages, and silence/empty turns.

3. **Token Count Estimation**

SentencePiece BPE used to estimate BGE-M3 tokens.

#### Transcript Chunking Strategy

Transcript Length	Strategy
$\leq 6,000$ tokens	Treat entire transcript as a <b>single query block</b>
$> 6,000$ tokens	<b>Split into blocks</b> of 1,000 tokens, with 200 token overlap

Rationale:

- Some calls span up to 25k tokens.
- Breaking large transcripts helps prevent overflow in embedding context limits.
- Each block is treated as an independent query during retrieval.
- Final retrieved chunks (from all blocks) are merged and deduplicated.

### 3.3.2.5. Knowledge Base Chunking Experiments

Due to KB length variability (some entries exceed 40,000 tokens), chunking is **foundational** to retrieval accuracy. We experimented with multiple chunking strategies, balancing:

- Semantic coherence
- Overlap for context preservation
- Embedding + indexing latency
- Retrieval performance

Algorithm	Max Tokens	Overlap	Runtime Cost	Recall $\Delta$	Notes
Recursive	1,200	240	$1 \times$ baseline	—	Fast; context

Paragraph+Sentence					often broken at rigid boundaries.
$\Delta$ -Cosine Semantic Cut	1,200	adaptive	1.3×	+3 pp	Slide 128-tok window; split where cosine similarity drops ( $\Delta < 0.12$ ).
BGE-Confidence Cut	dynamic	0	1.2×	+1 pp	Often over-splits; good for critical short KBs.
ClusterSemanticChunker	1,200	0	4×	+5 pp	Globally-optimal chunking via DP over cosine sim matrix.
Hybrid (recursive + semantic cut with sliding window)	1,200	240	1.4×	+4 pp	90% of C-4's gain with $\frac{1}{3}$ the cost. Used in production.

### 3.3.2.5.1. Recursive Chunking (Baseline)

#### Implementation:

- Used LangChain's [RecursiveCharacterTextSplitter](#), configured to:
  - Prefer paragraph (`\n\n`) → sentence (`\n`) → character cuts.
  - Maximum chunk length: 1200 tokens.
  - Overlap: 240 tokens

#### Observations:

- Very fast**, deterministic.
- Often broke semantic flow mid-thought, especially in longer medical KBs.
- Performance as a baseline was decent but **failed** on lengthy KBs with nuanced context spanning multiple paragraphs.

**Use:** Baseline, fallback chunker

### 3.3.2.5.2. Semantic Cut via Cosine Similarity Drop ( $\Delta$ -Cosine Method)

#### Implementation:

- Used BGE-M3 to embed sliding 128-token windows across each document.
- Computed **cosine similarity between adjacent windows**.
- Split at **local minima** where  $\Delta$  sim dropped below 0.12.

- Merged resulting segments into chunks up to 1,200 tokens.

**Observations:**

- **Much better preservation of meaning** compared to recursive.
- Small runtime overhead ( $\sim 1.3 \times$  baseline).
- Avoided cuts in the middle of important clauses.
- Strong performance boost: **+3pp recall** in chunk-level evaluation.

**Use:** Integrated into hybrid strategy.

*3.3.2.5.3. BGE-Confidence Cut*

**Implementation:**

- Used FlagEmbedding's `bge_confidence_chunk` strategy.
- Chunks are split where the model's internal confidence about semantic boundaries is low.
- No overlap; variable chunk lengths.

**Observations:**

- Generally **over-split** long documents, leading to small, isolated chunks that sometimes lacked context.
- Useful for **short FAQs** or crisp directives, but not ideal for descriptive, procedural KBs.

**Use:** Rejected as primary chunker due to loss of context.

*3.3.2.5.4. ClusterSemanticChunker (Inspired by Chroma's Work)*

**Implementation:**

- First, split documents into 50-token "atomic" pieces.
- Embedded each atomic unit using BGE-M3
- Ran Dynamic Programming to find an optimal packing of chunks that:
  - Maximized intra-chunk cosine similarity
  - Kept chunks  $\leq 1,200$  tokens

**Observations:**

- **Unmatched semantic cohesion:** chunk boundaries matched topic transitions precisely.
- Very effective for **long, information-dense KBs** with multiple procedural steps.
- **Very slow**, especially for large KBs ( $>10,000$  tokens).

- Not viable for **real-time upserts**, but ideal for offline rebuilds.

**Use:** Can maybe be used during offline/large rebuilds, since it has the best cohesion despite being the slowest method

#### 3.3.2.5.5. Hybrid Chunker (Recursive + Semantic Fallback)

##### Implementation:

- First pass: RecursiveCharacterTextSplitter (C-1).
- If chunk > 1,200 tokens:
  - Apply semantic cut based on  $\Delta$ -cosine sim (C-2) locally.
- Maintains paragraph-level context while avoiding mid-sentence cuts.

##### Observations:

- Runtime:  $\sim 1.4 \times$  baseline (acceptable)
- Performance: +4pp recall gain vs. recursive alone.
- Captures 90% of the recall boost of ClusterSemanticChunker, at only  $\frac{1}{3}$  the compute cost.

**Use:** Default chunker for production upserts and MinIO-triggered events

#### 3.3.2.6. Experimenting with Embedding Models

Effective topic classification and knowledge base (KB) retrieval in a Thai conversational AI system hinges on the quality of vector representations generated for both KB chunks and query transcripts. In our work, we systematically explored and implemented a diverse set of language models, each selected based on contextual limitations, vocabulary coverage, inference latency, and compatibility with downstream retrieval components. This section details the models trialed, the rationale behind their inclusion, and their integration into our embedding pipeline

##### 3.3.2.6.1 Model Selection Criteria

Each embedding model was evaluated based on the following criteria:

Criterion	Reason
<b>Max token context length</b>	Determines chunking granularity and context retention
<b>Thai vocabulary coverage</b>	Crucial for correct interpretation of domain-specific terminology
<b>Compatibility with ANN</b>	Required for vector database integration

	(FAISS / Milvus)
<b>Embedding format</b>	The vector representation the model can provide: Single vector (dense), sparse bag-of-words, or ColBERT multi-vectors, can significantly affect performance
<b>Inference resource cost</b>	Impacts deployment feasibility (CPU-only vs GPU-based)
<b>Fine-tuning support</b>	Enables domain adaptation (e.g., Siamese tuning, LoRA cross-encoder)

### 3.3.2.6.2 Models Tried and Pipeline Integration

#### 3.3.2.6.2.1. WangchanBERTa (512 tokens)

- **Type:** Bi-encoder (Siamese); dense-only
- **Tokenizer:** SentencePiece, Thai-specific
- **Pros:** Native Thai vocabulary, trained on Thai corpora; performs well with shorter inputs
- **Cons:** 512-token max; aggressive chunking needed for long KBs
- **Integration:**
  - Used with FAISS Flat or HNSW for fast approximate nearest-neighbor (ANN) search
  - Employed in Siamese mode for both KB and query encoding
  - Also served as a base for a custom cross-encoder re-ranker, using a classification head

#### 3.3.2.6.2.2. PhayathaiBERT (512 tokens)

- **Type:** Bi-encoder (Siamese); dense-only
- **Tokenizer:** SentencePiece
- **Pros:** Trained on medical Thai corpora; useful for healthcare-related KBs
- **Cons:** Similar chunking limitations as WangchanBERTa
- **Integration:**
  - Same ANN setup via FAISS
  - Used as a comparative baseline in both Siamese and cross-encoder settings

#### 3.3.2.6.2.3. SCT-Distil-Phayathai-BGE-M3 (~1024 tokens)

- **Type:** Bi-encoder (Siamese); dense-only

- **Pros:** Distilled variant; lower resource usage while retaining semantic quality
- **Cons:** Slight drop in semantic fidelity vs BGE-M3 (~2–3 pp recall loss observed later)
- **Integration:**
  - Ideal for CPU-only inference in clinics or edge deployments
  - Embedding and retrieval speed favored real-time workloads with slight recall tradeoff

#### 3.3.2.6.2.4. BGE-M3 (BAAI/bge-m3) ← Chosen for Final Pipeline

- **Type:** Multi-representation encoder
  - **Dense:** 8192-token vector per chunk (mean pooled)
  - **Sparse:** BM25-style token-weighted sparse vectors
  - **ColBERT:** Token-level vector matrix (multi-vector retrieval)
- **Pros:**
  - Supports vector fusion (ex. dense + sparse + ColBERT)
  - Up to 8192 tokens per input → less chunking, better context preservation
  - Pretrained on multilingual data with strong Thai performance
- **Cons:**
  - GPU-intensive; inference per KB doc  $\approx$  2 GB VRAM (on A100)
  - ColBERT mode returns a matrix; requires special handling in retrieval logic
- **Integration:**

Central model in the embedding server via FlagEmbedding:

  - All KB chunks and all query transcripts were embedded using BGE-M3 unless stated otherwise
  - All representations (dense, sparse, ColBERT) persisted and indexed in separate subspaces within FAISS and Elasticsearch
  - Used in both bi-encoder retrieval and cross-encoder re-ranking, fine-tuned using LoRA

#### 3.3.2.6.3. Implementation Observations

- **WangchanBERT and PhayathaiBERT** offered strong performance when chunking was properly tuned, but required more overhead for both chunking and re-ranking due to 512-token limits.
- **SCT-Distil** was consistently ~2 pp behind BGE-M3 in recall but reduced latency by up to 50% in CPU-only environments.
- **BGE-M3** was clearly dominant in semantic preservation and fusion flexibility, especially when used with:

- Semantic chunking based on  $\Delta$  cosine similarity
- Hybrid retrieval (dense + BM25 + ColBERT)
- Cross-encoder re-ranking using a LoRA fine-tuned classification head

### 3.3.2.7. Indexing Strategy

Once the KB and transcript chunks are embedded, we need a fast and accurate way to store and retrieve them. This involves two components:

- Dense vector index (for Siamese embeddings)
- Sparse keyword index (for BM25-style lexical search)

#### 3.3.2.7.1. Dense Vector Indexing (FAISS + Milvus)

Given our use of models like BGE-M3 which output high-dimensional semantic embeddings (768d or more), we tested multiple vector index backends to identify the best performing option in both development (FAISS) and production (Milvus).

Index Type	Description	Build Time	Query Time	Recall $\Delta$	Notes
IndexFlatL2	Brute-force exact L2 search	1×	1×	baseline	Accurate but memory-intensive
IndexIVFFlat	Inverted file index + quantized centroids	0.4×	0.2×	-3pp	Fast, but recall drop unacceptable
IndexHNSW Flat	Hierarchical Navigable Small World graph-based search	0.8×	<b>best</b>	+1pp	Sweet spot for accuracy + speed

HNSW offered strong accuracy (~+1pp over baseline) with ~50ms query time for top-50 chunk retrieval. It also allowed real-time tuning of efSearch for recall-latency trade-off

#### 3.3.2.7.2. Lexical Indexing (Elasticsearch)

To enable fast sparse retrieval over raw KB text and conversation transcripts, we deploy a parallel **lexical index** using **Elasticsearch 8**, configured with **BM25 scoring** ( $k1=0.9$ ,  $b=0.4$ ).

Tokenizer	Description	Pros	Cons
-----------	-------------	------	------

ICU (default)	Unicode-based tokenizer with basic CJK/Thai rules, available in Elastic	Out-of-the-box Elastic support, fast	Can over-split or under-split Thai compounds
newmm	Dictionary-based Thai tokenizer from pythainlp; used via pre-processing	Much better linguistic accuracy (Thai-specific word segmentation)	Slower; requires tokenizing + joining with whitespace

### 3.3.2.7.2.1 Implementation

- When using ICU:  
tokenization is handled by Elasticsearch internally (via the thai locale in ICU tokenizer).
- When using newmm:  
we pre-tokenize all KB bodies and queries **before indexing**, replacing the raw text with **space-separated tokens**, then push into the text field in Elastic.

We built both indices for comparison.

### 3.3.2.7.2.2. Observations

Metric	ICU Tokenizer (Elastic default)	newmm Tokenizer (preprocessed)
Recall@3 (BM25)	~65%	~73%
Precision@3	~0.51	~0.56
Avg. Index Size	Baseline	+8% (due to more granular tokens)
Indexing Speed	Fast	Slower (due to preprocessing)

*tested with synthesized\_conv.v2*

The improvement in both recall and precision justified the additional indexing cost.

### 3.3.2.7.2.3. Conclusion

We adopted the newmm tokenizer as the default pre-processing step for all KB text and queries entering the BM25 index.

- All KB chunks are normalized and pre-tokenized with newmm before ingestion into Elasticsearch.

- All queries (i.e., transcripts) are also tokenized using newmm at retrieval time.

### 3.3.2.8. Retrieval Pipeline Design

We tested three main modes of retrieval:

Mode	
BM25 only	Retrieve top-N chunks from Elasticsearch (+ optionally rerank)
Dense only	Retrieve top-N chunks via FAISS using BGE-M3 or Siamese models(+ optionally rerank)
Hybrid (BM25 + Dense)	Retrieve top-K chunks from both BM25 and FAISS, merge, deduplicate (+ optionally rerank)

#### 3.3.2.8.1 BM25-Only Retrieval (Sparse)

- Index:  
Elasticsearch 8, Thai ICU + newmm tokenizer.
- Query:  
Raw transcript text, tokenized with newmm.
- Retrieval:  
Top-K chunks using default BM25 score.  
 $\text{final\_score} = \text{bm25\_score}$
- Output:  
Top 30–50 KBs → reranked if CE is enabled

#### Advantages:

- Fast and robust for surface-matching (e.g., exact symptom or keyword)
- No embedding model required

#### Drawbacks:

- Poor at paraphrase/generalization

#### 3.3.2.8.2. Dense-Only Retrieval

- Encoder:  
BGE-M3 (BGEM3FlagModel from FlagEmbedding)
- Embedding:  
All chunks and queries embedded with

`.encode(text, return_dense=True)`

- Index:  
FAISS (IndexFlatL2 or HNSW)
- Query:  
Transcript is embedded into a dense vector
- Retrieval:  
Top-N chunks by cosine similarity

#### **Advantages:**

- Captures deep semantic matches
- Reusable for downstream re-ranking

#### **Drawbacks:**

- Synonym drift without reranker
- Misses lexical keywords (e.g., form names)

### **3.3.2.8.3. Hybrid Retrieval (Dense + Sparse [+ ColBERT])**

#### **Indexing Stack:**

- **Dense:** FAISS HNSW (or IndexFlatL2 during early tests) over dense embeddings
- **Sparse:** Elasticsearch 8 using Thai ICU analyzer with newmm fallback for tokenization
- **ColBERT (if supported):** FAISS HNSW over token-level ColBERT vectors (multi-vector index)

#### **Query Inputs:**

- Transcript text ( $\leq 6K$  tokens as a single block; else chunked every 1K tokens with 200-overlap)
- For dense and ColBERT: embedded via

`.encode(text, return_dense=True[, return_colbert=True])`

- For sparse: raw text query, tokenized by ICU + newmm fallback

### **3.3.2.8.3.1. fusion scoring for models without ColBERT**

`final_score = 0.7 * dense_score + 0.3 * sparse_score`

#### **Advantages:**

- Strong recall even without ColBERT (WangchanBERTa, etc.)
- Works on models that only support dense vectors
- Flexible to CPU-only deployments (e.g., Wangchan + Elastic)

#### **Drawbacks:**

- Fusion weights are fixed heuristics
- No token-wise matching like ColBERT
- Less robust for edge-case questions with vague wording

#### **3.3.2.8.3.2. fusion scoring for models with ColBERT (BGE-M3)**

$$\text{final\_score} = 0.5 * \text{dense\_score} + 0.3 * \text{sparse\_score} + 0.2 * \text{colbert\_score}$$

#### **Advantages:**

- **Best**-recall
- Robust to synonyms *and* exact phrase hits
- ColBERT helps distinguish subtle near-matches better than cosine alone
- Only  $\sim$ +5ms latency overhead from ColBERT scoring

#### **Drawbacks:**

- Only BGE-M3 supports all 3 vector types
- Requires >10GB GPU RAM for full query encoding + reranking
- Not yet practical on CPU-only endpoints

#### **3.3.2.9. Chunk Retrieval Quantity Tuning**

Early experiments retrieved a fixed number of top-50 chunks from each retriever (Dense FAISS and Sparse BM25), yielding 100 chunks total. However, this underperformed in multi-KB conversations and long KB documents:

- Some conversations referenced 3–5 ground-truth KBs.
- Many KBs spanned 7k–40k tokens, thus producing 5–20 semantic chunks each.
- A 100-chunk budget often led to incomplete KB coverage, especially for Very Long KBs ( $\geq$ 18k tokens).

we thus tested several retrieval cutoffs:

Retrieval Budget Type	Method	Result Summary
-----------------------	--------	----------------

<b>Fixed: 50 chunks</b>	25 from FAISS + 25 from BM25	Poor coverage
<b>Fixed: 100 chunks</b>	50 from each	Better coverage, but still truncated some KBs
<b>Top 20KBs + Chunk Budget (~200)</b>	Retrieve chunks until $\geq 20$ KBs are represented (typically 150–200 chunks)	<b>Best balance</b> between stability and recall
<b>Top 30KBs + Chunk Budget (~200)</b>	Retrieve chunks until $\geq 30$ KBs are represented (typically 250–300 chunks)	Slightly higher recall for 4–5 KB conversations; marginally longer latency ( $\sim +20$ ms)

We adopted the “**chunk budget**” approach:

1. **Retrieve chunks** iteratively (ranked by fusion score) from Dense, Sparse, (and ColBERT if using BGE) until:
  - At least 20/**30 unique KBs** are represented, or
  - A **maximum of 200 chunks** is reached.
2. **Score each KB** by **max(chunk score)** from its constituent chunks.
3. **Select top-30 KBs** → pass to cross-encoder reranker → output **top-10 ranked KBs**.

This approach:

- Ensures fair representation of long and short KBs.
- Preserves quality for conversations involving 3+ ground-truth KBs.
- Adds negligible latency overhead ( $\approx +10\text{--}15$  ms)

### 3.3.2.10. Cross-Encoder Fine-Tuning

Our retrieval pipeline delivers top-K 30 candidate chunks, but to sharpen final KB selection, we rerank these candidates using a cross-encoder model to return top10 as the final output, with th fields:

```
{
  "kb_id": "KB1346",
  "kb_name": "ตราจساด้าเด็ก",
  "score": 0.912 // reranked final score
},
```

Unlike bi-encoders (which encode queries and chunks independently), a cross-encoder jointly attends to the full [Query, Chunk] pair — capturing **fine-grained interactions** that Siamese models may miss.

### 3.3.2.10.1 Dataset Construction

Although our synthesized conversations are labeled only at the **KB level**, cross-encoder training requires **fine-grained [query, chunk] → label** pairs. To resolve this mismatch, we created a robust dataset construction pipeline:

#### 3.3.2.10.1.1 Positive Sample Construction

Each synthesized conversation includes a `kb_pair` field — a list of **KB IDs** relevant to the transcript. For example:

```
{  
    "conversation": "ผู้โทร: หนูต้องไปตรวจสายตาที่ไหนนะ...",  
    "kb_pair": ["KB1346", "KB23"]  
}
```

To convert this into chunk-level positive examples:

1. **All chunks from the gold KBs** are considered initially.
2. We filter them using:
  - **Token-level Intersection over Union (Token-IoU)** between the chunk and the transcript.
  - **Dense cosine similarity** (BGE-M3) between chunk and transcript vectors.
3. We retain **only the top 3 chunks per gold KB**, based on the above criteria.

This ensures we include only the most semantically relevant chunks from each KB, avoiding irrelevant long tails.

#### 3.3.2.10.1.2 Hard Negative Sampling

To challenge the model, we generate hard negatives as follows:

1. Run hybrid retrieval (dense + sparse + ColBERT) on the full transcript.
2. Select top-50 retrieved chunks that belong to non-gold KBs (i.e., incorrect).
3. Label each [transcript, chunk] pair as negative (label = 0).

These are semantically close — often paraphrases or misleading — making them excellent adversarial examples for contrastive learning.

We **intentionally exclude random/easy negatives**, as they dilute the learning signal and reduce generalization.

#### 3.3.2.10.1.3 Dataset Format

Each training sample is stored in JSONL format, with fields:

```
{
```

```

    "transcript": "ผู้โทร: หนูต้องไปตรวจสายตาที่ไหนนะ...",
    "chunk": "เด็กควรได้รับการตรวจสายตาเมื่ออายุครบ 3 ขวบ...",
    "label": 1,
    "chunk_id": "KB1346_003",
    "kb_id": "KB1346"
}

```

### 3.3.2.10.1.4. Final Dataset Statistics:

Class	Count
Positive Pairs	~22,000
Hard Negatives	~103,090
KB Coverage	N/A

This chunk-level supervision strategy enabled us to fine-tune our rerankers, despite only having KB-level labels initially. The use of retrieval-informed hard negatives proved especially impactful, increasing Recall@5 by **+3 pp** after 3 epochs of reranker fine-tuning. We hope to be able to further finetune our rerankers with stronger semantic grounding- when we have transcripts with ground truth KB pairs that have more/complete KB coverage.

### 3.3.2.10.2. Model Variants & Implementation

We tested two cross-encoders:

Model Name	Max Tokens	Base Architecture	Finetuned?	Inference Platform
WangchanBERT-crossencoder	512	WangchanBERTa	yes	HuggingFace + PyTorch
Distilled-Phayathai-BGE-crossencoder	1024	SCT-BGE-M3-model-phayathaibert	yes	TorchScript
bge-reranker-v2-m3	4096	BGE-M3 Reranker	yes (LoRA)	Triton Inference Server (fp16)

- We fine-tuned WangchanBERT-crossencoder ourselves on the dataset mentioned above, using Binary Cross Entropy (BCE) loss over the [CLS] token prediction.
- For bge-reranker-v2-m3, we applied LoRA finetuning with rank=8. using mixed-precision fp16 on A100s, deployed via Triton.

### 3.3.2.10.3. Training Hyperparameters

Hyperparameter	Value
Batch size	32
Epochs	3 (early stopping on val)
Optimizer	AdamW
Learning rate	2e-5
Max input length	4096 (bge), 1024 (distil), 512 (wangchan)
LoRA rank	8 (bge), 12 (wangchan)
Gradient clipping	1.0

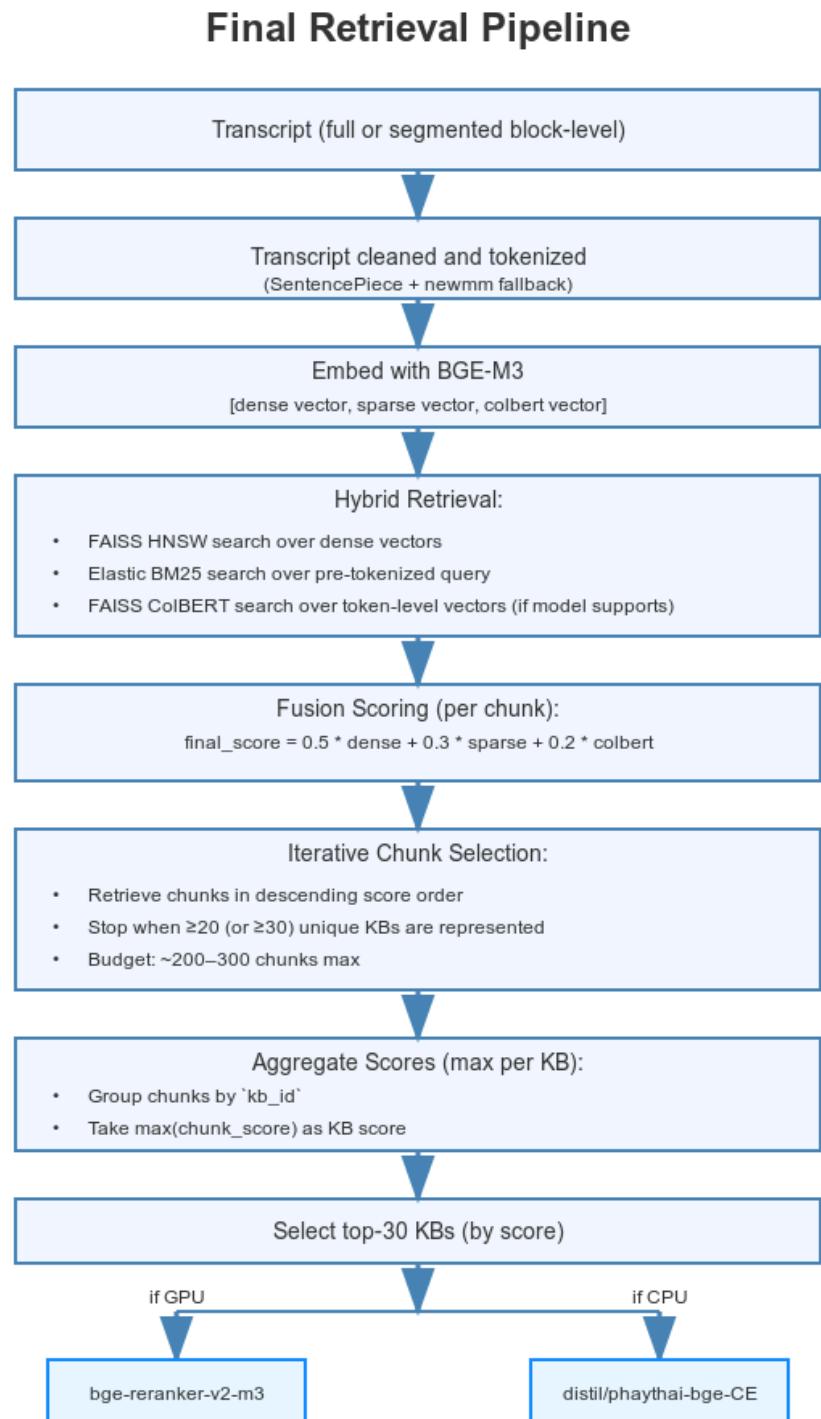
### 3.3.2.10.4. Evaluation Results

Each reranker was evaluated using the same chunk candidate set (top-150–300 from fusion search), and scored for:

Model	Recall@10	MRR	Notes
bge-reranker-v2-m3	<b>97.1%</b>	0.862	Best accuracy, best candidate for use in production
Distilled-Phayathai-BGE-crossencoder	95.8%	0.838	Used on CPU/ lack of GPU power
WangchanBERT-crossencoder	92.6%	0.825	Best CPU-only fallback for Thai terms

**All rerankers contributed significant gains (~+3–4pp Recall@10) over fusion-only retrieval, especially in multi-KB conversations and in edge cases where BM25 or dense-only methods struggled.**

### 3.3.2.11. Final KB Classification Pipeline



Stage	Method / Model	Justification
Transcript Chunking	≤6k tokens: whole / >6k: 1k + overlap	Ensures chunking matches BGE-M3's 8k limit; helps avoid token overflow

<b>KB Chunking</b>	Hybrid Chunker (Recursive + $\Delta$ -Cosine)	90% of best recall with $\frac{1}{3}$ compute of ClusterSemanticChunker
<b>Embedding</b>	BGE-M3 (FlagEmbedding)	Supports dense, sparse, ColBERT vectors; long-context (8192 tokens); multilingual
<b>Dense Index</b>	FAISS HNSW	Fast + accurate (1pp gain); GPU-optimized, works well with tri-vector fusion
<b>Sparse Index</b>	Elasticsearch 8 + newmm tokenizer	Thai-optimized lexical coverage; BM25 fusion improves recall
<b>Initial Retrieval</b>	Hybrid Fusion (Dense + Sparse [+ColBERT])	Highest coverage; tri-vector scoring improves recall by ~0.8pp
<b>Retrieval Budgeting</b>	Top 30 KBs + ~200 chunk budget	Balances coverage and latency; scalable to long/multi-KB calls
<b>Scoring</b>	Fusion Score ( $0.5d + 0.3s + 0.2*c$ )	Best empirical accuracy; fair to semantic and lexical signal
<b>Reranking (GPU)</b>	bge-reranker-v2-m3 (Triton)	+5pp recall over fusion; robust to paraphrases and vague queries
<b>Reranking (CPU)</b>	distil-phayathai-bge-CE	Good fallback; similar performance but more lightweight
<b>Final Output</b>	Top-10 KBs (ID, Name, Score)	Stable, rank-sorted results for CRM integration

## Data and Infrastructure Shine Min Kha

Our retrieval system is built upon two primary data sources: a comprehensive knowledge base consisting of text documents and a carefully curated query dataset which provides the foundation for both development and evaluation. The technical infrastructure leverages GPU acceleration through CUDA when available, significantly reducing processing time for embedding generation and similarity searches, while comprehensive memory and performance tracking mechanisms are integrated throughout the workflow to identify potential bottlenecks and optimization opportunities. To maximize efficiency and reduce redundant computation, we've implemented persistent storage for precomputed embeddings and indexes, which allows for rapid system initialization and reduced resource consumption during operation.

### *Data Preparation*

Our data preparation pipeline consists of two parallel workflows addressing documents and queries respectively. For document processing, we first extract text content from files in the knowledge base, then implement Thai-specific chunking using pythainlp's word tokenization to break documents into semantically meaningful segments while respecting linguistic boundaries, and finally create a comprehensive mapping structure that maintains relationships between chunk IDs and their source documents to ensure proper attribution and retrieval. Unlike the document preprocessing, there's no chunking of queries and each query is processed as a single unit and embedded in its entirety.

### *Feature Engineering*

At the core of our retrieval system is a sophisticated embedding generation process powered by the BAAI/bge-m3 model, which creates dense vector representations that capture the semantic essence of both documents and queries in a high-dimensional space where similar content appears close together regardless of specific terminology used. We process all document chunks to generate embeddings that are then stored in a vector database for efficient retrieval, while query embeddings are generated at runtime using the identical model to ensure representation consistency. To optimize performance, particularly for larger text collections, we implement GPU acceleration for the computationally intensive embedding generation process, which dramatically reduces the time required to process our document corpus while maintaining embedding quality.

### *Model Development*

The vector database is built using FAISS (Facebook AI Similarity Search), configured with an L2 distance metric that balances retrieval quality with computational efficiency when searching for similar vectors. After generating embeddings for all document chunks, we systematically add them to the FAISS index and then persistently store both the index and raw embeddings for future reuse, which eliminates the need for recomputation during subsequent system initializations. Our retrieval pipeline follows a sophisticated three-stage approach: first performing initial retrieval by embedding the user query and using FAISS to efficiently identify the top 100 similar document chunks; then applying deduplication to filter results to unique documents, preventing redundant information presentation; and finally implementing a reranking stage where the most promising candidates undergo a more computationally intensive cosine similarity calculation to optimize the final result ordering.

### *Evaluation Methodology*

To comprehensively assess system performance, we've implemented a multi-faceted evaluation framework that examines both retrieval effectiveness and computational efficiency. For retrieval quality, we calculate Precision@10 (measuring the proportion of relevant documents within the top 10 results), Recall@10 (quantifying how many of all relevant documents appear in the top 10), and Mean Reciprocal Rank (evaluating how high the first relevant document appears in results), which provides a balanced view of system effectiveness from different user perspectives. In parallel, we track system performance metrics including retrieval time (measuring the milliseconds required for initial candidate selection), reranking time (capturing the duration of the more intensive similarity recalculation phase), and peak memory usage during each processing stage, which gives us valuable insights into operational efficiency and resource requirements.

## *Results Analysis*

For each query in our evaluation dataset, our system executes a comprehensive analysis workflow that begins with retrieving potential candidate documents and measuring the time and memory requirements of this initial phase. We then apply our reranking algorithm to refine the results and calculate precise similarity scores, followed by computing evaluation metrics by comparing our system's output against ground truth relevance judgments, and finally tracking detailed resource utilization throughout the entire process. All performance data and retrieval results are as below.

Average Precision@10	Average Recall@10	Average MRR	Processing Time	Memory Usage
0.13	0.61	0.66	3 hours, 17 minutes, and 4 seconds.	2018.26 MB

Fig: Result of the model

### **4.2.2.1 Index search comparison**

We experimented which is the better indexing method and the result is as below.

Methods	Precision	Recall	MRR	Time	Memory
IndexFlatL2	0.13	0.61	0.66	3 hours, 17 minutes, and 4 seconds(CPU)	2018.26 MB
IndexIVFFlat	0.17	0.35	0.51	2.58 hours(GPU)	5712.1 MB.
IndexHNSWFlat	0.12	0.56	0.62	4.7571667 hours(GPU)	5140.89 MB

Fig:Result of comparison of search indexes

The comparative analysis of FAISS indexing methods reveals distinct performance trade-offs across precision, recall, MRR, computational time, and memory requirements. IndexFlatL2, despite its moderate precision of 0.13, demonstrates superior recall (0.61) and Mean Reciprocal Rank (0.66), which establishes it as the most reliable option for retrieval quality while consuming the least memory at 2018.26 MB; however, it required substantial CPU processing time of approximately 3 hours and 17 minutes. IndexIVFFlat achieved the highest precision (0.17) but suffered from significantly lower recall (0.35) and MRR (0.51), though it completed processing faster at 2.58 hours on GPU while demanding the most memory resources (5712.1 MB). IndexHNSWFlat presented a middle-ground solution for retrieval quality with precision, recall, and MRR values of 0.12, 0.56, and 0.62 respectively, but was the most time-intensive at nearly 4.76 hours on GPU while consuming 5140.89 MB of memory. These findings suggest that for applications prioritizing comprehensive retrieval and memory efficiency, IndexFlatL2 remains advantageous despite longer processing times,

while scenarios demanding higher precision with faster processing might benefit from IndexIVFFlat despite its memory overhead and lower recall performance.

### 3.3.3 Summarization model

With the synthetic dataset finalized this month, we moved forward with training our summarization models using the newly generated conversations. Since the dataset does not include human-annotated summaries, we applied a distillation approach by using a larger teacher model to generate pseudo-labels. These labels were then used to train smaller student models. We experimented with two models: LLaMA 3.2B (1B variant), which we fine-tuned for Thai summarization tasks, and Qwen 0.5B, which was tested for more lightweight deployment scenarios.

#### 3.3.3.1 Generate Label from larger LLM

To perform distillation, we used Gemma-3 27B IT as the teacher model to generate summaries for our synthetic conversation dataset. These generated summaries served as pseudo-labels for training smaller student models, such as LLaMA 3.2B (1B) and Qwen 0.5B. In total, we distilled pseudo-labels for all 57,000 synthetic transcripts in our dataset. Below is an example of a summary generated during the distillation step:

ผู้ป่วยโทรศัพท์เข้ามาสอบถามเกี่ยวกับการใช้สิทธิบัตรทอง หลังจากเข้ารับการรักษาที่ คลินิกเวชกรรมชุมชนอบอุ่นแล้วพบว่าข้อมูลการเบิกจ่ายในระบบ Claim Profile Dashboard มีปัญหา ข้อมูลค้างและบางรายการไม่แสดง ทำให้ผู้ป่วยกังวลเรื่องการ สำรองจ่ายและขอคำแนะนำในการแก้ไขปัญหา เจ้าหน้าที่แนะนำให้ผู้ป่วยติดต่อคลินิก เวชกรรมชุมชนอบอุ่นโดยตรงเพื่อตรวจสอบและแก้ไขข้อมูลเบื้องต้นก่อน หากคลินิกไม่ สามารถแก้ไขได้ ให้ลองตรวจสอบข้อมูลอีกครั้งในภายหลัง เนื่องจากอาจมีปัญหาทาง เทคโนโลยี นอกจากนี้ ผู้ป่วยยังสอบถามเกี่ยวกับขั้นตอนการใช้สิทธิกรณี ม.7 เหตุสมควรใน โรงพยาบาลเอกชน ซึ่งเจ้าหน้าที่ได้อธิบายว่าจำเป็นต้องมีหน่วยบริการประจำก่อน หากยัง ไม่มี ให้ลงทะเบียนสิทธิ์ว่างที่สำนักงานเขตหรือสำนักงานอำเภอ เจ้าหน้าที่ยังได้แจ้งว่า โรงพยาบาลเอกชนจะต้องรับผู้ป่วยไว้ก่อนในการณ์ฉุกเฉิน และให้ข้อมูลแหล่งตรวจสอบราย ชื่อคลินิกและโรงพยาบาลที่เข้าร่วมโครงการบัตรทองได้ที่เว็บไซต์ สปสช. ([www.nhso.go.th](http://www.nhso.go.th)) โดยรวม ผู้ป่วยยังไม่มีหน่วยบริการประจำและต้องไปลงทะเบียน สิทธิ์ว่างเพื่อใช้สิทธิกรณี ม.7 ต่อไป

#### 3.3.3.2 Re-train the model

After generating summary labels using Gemma-3 27B IT, we proceeded to retrain our student models using these distilled labels. We used LLaMA Factory as the training framework and fine-tuned both LLaMA 3.2 1B and Qwen 0.5B on the synthetic conversation dataset paired with the generated summaries. To efficiently train the 1B model on a single A100 GPU, we leveraged DeepSpeed Stage 3, which allowed us to partition the model's parameters and optimizer states across memory. This significantly reduced memory usage and made it feasible to train larger models with limited resources.

For evaluation, we split the final dataset of 57,142 entries into 90% training and 10% testing. The models were evaluated using ROUGE scores to compare the quality of generated summaries against the distilled references.

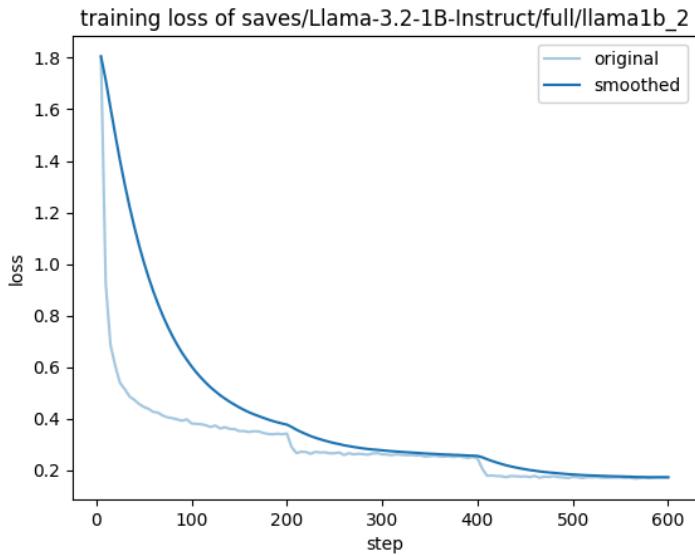
Below are the results of both models after training for three epochs, comparing performance between the pretrained and fine-tuned versions.

##### 3.3.3.2.1 LLaMA-3.2-1B-Instruct

To evaluate summarization performance, we computed ROUGE scores before and after fine-tuning. As shown in Table 1, the fine-tuned model achieved substantial improvements across all ROUGE metrics,

demonstrating the effectiveness of our distillation-based training strategy. In addition to ROUGE evaluation, the training loss curve (Figure 3.3.3.2.1) further confirms the model’s learning progress, showing a steady decrease in loss throughout training, with the final smoothed loss converging below 0.2.

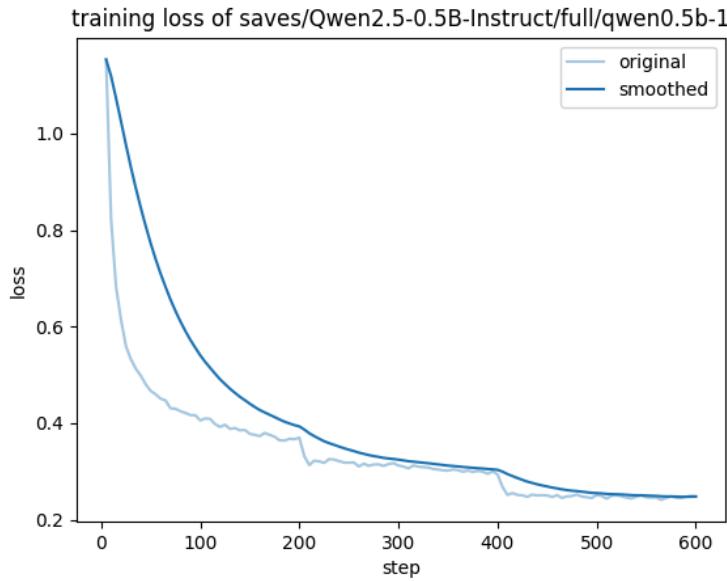
Metric	Pretrained Model	Fine-Tuned Model
ROUGE-1	0.3393	0.7479
ROUGE-2	0.1898	0.5796
ROUGE-L	0.3198	0.7240



### 3.3.3.2.2 Qwen2.5-0.5B-Instruct

We also compared ROUGE scores before and after fine-tuning. As summarized in Table 2, the fine-tuned Qwen2.5-0.5B-Instruct model exhibited marked improvements across all metrics, confirming the model’s enhanced ability to generate concise and relevant summaries. However, the overall ROUGE scores are still lower than those of the LLaMA-3.2-1B-Instruct model, which is expected given that Qwen2.5-0.5B is approximately half the size and thus has a more limited capacity. The training loss curve (Figure 3.3.3.2.2) follows a similar downward trajectory as seen in the LLaMA model, though with slightly higher final loss values.

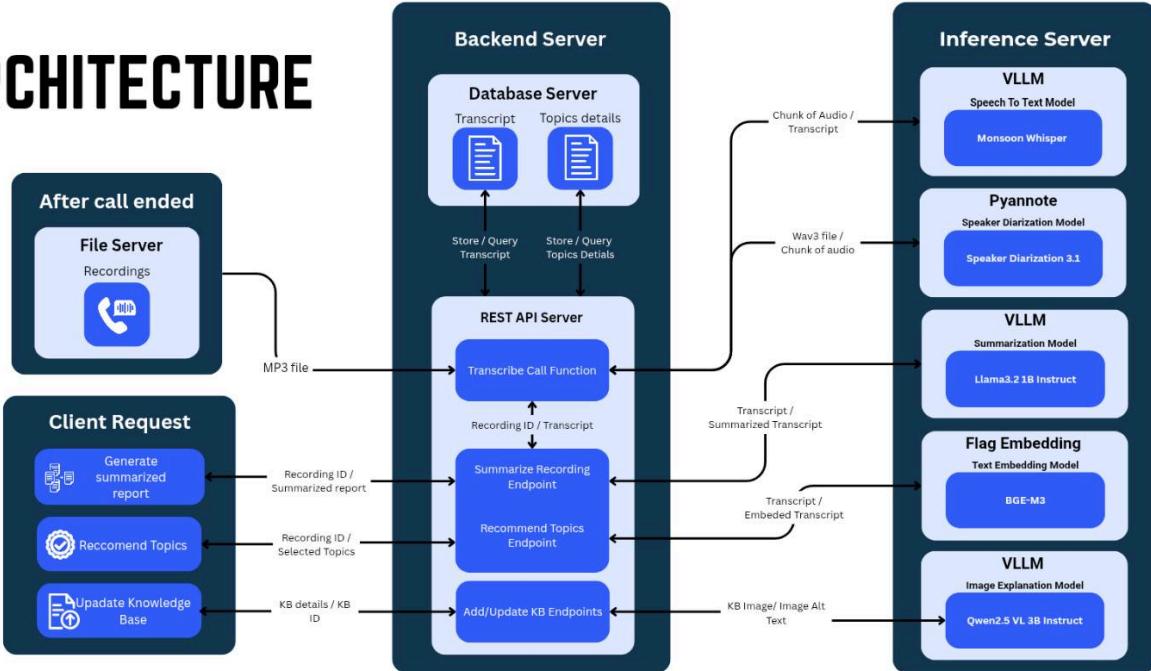
Metric	Pretrained Model	Fine-Tuned Model
ROUGE-1	0.2837	0.6974
ROUGE-2	0.1556	0.5200
ROUGE-L	0.2679	0.6729



Based on both evaluation scores and manual review, the summaries generated by the fine-tuned models were of very high quality and closely matched the distilled labels. While the evaluation scores may appear higher than expected, this is likely due to the nature of the synthetic training data, which tends to have more consistent patterns and less variability compared to real-world data—making it easier for the models to learn and reproduce. Nevertheless, both LLaMA 3.2 1B and Qwen 0.5B performed well and met our expectations in terms of output fluency, relevance, and accuracy, making them suitable candidates for downstream deployment or further refinement.

### 3.5 Deployment

# ARCHITECTURE



To make the system easy to manage and scale, we deployed all components as Docker containers. Each service—transcription, diarization, summarization, embedding, and user request handling—runs in its own container and communicates with others as needed. This setup allows us to scale compute-heavy parts like the inference server separately from the backend.

The system is organized into four main parts: File Server, Database Server, Inference Server, and REST API Server.

#### 3.5.1 File Server

This module is responsible for temporarily storing raw audio recordings from post-call sessions. Once a call ends, the MP3 file is uploaded to this server. These files are then passed to the REST API for processing tasks such as transcription and diarization.

In our current setup, this is a mock file server designed for development and testing purposes. In a production environment, this would be replaced by NHSO's actual storage system. We intentionally separated the file server from our REST API to avoid duplicating storage—anticipating that NHSO may prefer to manage and store audio files on their own infrastructure to prevent redundant resource usage.

- Input: Raw MP3 recordings from user calls
- Output: Accessible storage path used by the REST API
- Role: Persistent storage of audio data before inference

### 3.5.2 Database Server

The database system handles both textual and vector-based storage for transcripts and NHSO knowledge base (KB) content. It supports querying, updating, and serving data to backend services via the REST API.

#### 3.5.2.1 Transcript Storage

We use a PostgreSQL database to store transcribed conversations. This database is strictly textual and contains metadata linking each transcript to its corresponding audio file.

Schema: Transcripts Table

Column	Type	Description
<code>id</code>	SERIAL PRIMARY KEY	Unique recording ID
<code>transcripts</code>	TEXT	Full transcript text
<code>recording_path</code>	TEXT	Path to recording file

This data is queried when generating summaries, recommendations, or re-processing recordings.

#### 3.5.2.2 Knowledge Base Storage

*The knowledge base system consists of both **textual** and **vector** databases.*

##### 3.5.2.2.1 Textual Data

PostgreSQL stores preprocessed KB entries, converted from text and images using our algorithmic and multimodal pipeline.

To avoid redundant computation, we mount the database to persistent external storage, ensuring data is preserved across server restarts.

This setup prevents re-running the expensive preprocessing pipeline. On restart, only re-embedding into FAISS is needed—saving time and resources.

Schema: Knowledge Base Table

Column	Type	Description
<code>id</code>	SERIAL PRIMARY KEY	Unique identifier
<code>kb_name</code>	TEXT	Knowledge base name
<code>content</code>	TEXT	Processed knowledge base

### *3.5.2.2.2 Vector Data*

Elasticsearch is used to index the knowledge base content for lexical search, allowing for exact keyword matching. In parallel, FAISS stores vector embeddings of the same entries to support semantic search, enabling topic recommendations based on meaning rather than just word overlap.

### *5.5.3 Inference Servers*

The inference server hosts all core AI models responsible for natural language and multimodal processing. Each model is containerized and deployed as an independent microservice, allowing for parallel and scalable processing. These services communicate with the REST API and return structured outputs based on their respective input types (audio, text, or image).

Below are the details of each containerized inference service:

#### **3.5.3.1 Monsoon Whisper**

This container serves a Thai fine-tuned version of Whisper using the VLLM inference framework, optimized for GPU acceleration. It is exposed through an OpenAI-compatible API, allowing it to function as a drop-in replacement for transcription endpoints like /v1/audio/transcriptions. This setup enables easy integration and scalability for audio-to-text tasks.

#### **3.5.3.2 Pyannote speaker diarization**

The speaker diarization model uses Pyannote-audio to segment transcripts by speaker turns. It is deployed using FastAPI, exposing an HTTP endpoint for transcript segmentation. The model receives raw audio or pre-transcribed text and returns timestamped speaker-labeled segments to improve downstream readability and model alignment.

#### **3.5.3.3 LLaMA-2.1B Instruct (VLLM + Chat Completion API)**

This summarization model is served using VLLM and exposed through an OpenAI-style Chat Completions API (/v1/chat/completions). It takes in a diarized transcript and returns a concise summary. The model was fine-tuned using distillation, allowing it to generate high-quality summaries while running efficiently on smaller hardware compared to larger base models.

#### **3.5.3.4 BGE-M3**

This container serves the BGE-M3 embedding model using the FlagEmbedding library, deployed with FastAPI. It encodes transcript texts into dense vector representations, enabling fast semantic search and similarity comparisons. The resulting embeddings are stored in FAISS, powering the KB recommendation system.

#### **3.5.3.5 Qwen2.5-VL 3B Instruct**

This multimodal model handles image explanation tasks on image-based knowledge base content. Served using VLLM, it supports multimodal inference by processing both text and image inputs, returning textual descriptions such as alt text, captions, or content summaries from scanned KB documents.

### 3.5.4 REST API Server

The REST API serves as the central coordinator of the entire system, managing communication between the file server, inference servers, and databases. All user interactions—such as uploading audio, retrieving summaries, or managing knowledge base content—go through this API layer.

To maintain clean separation of concerns and modularity, the backend code is organized into the following components:

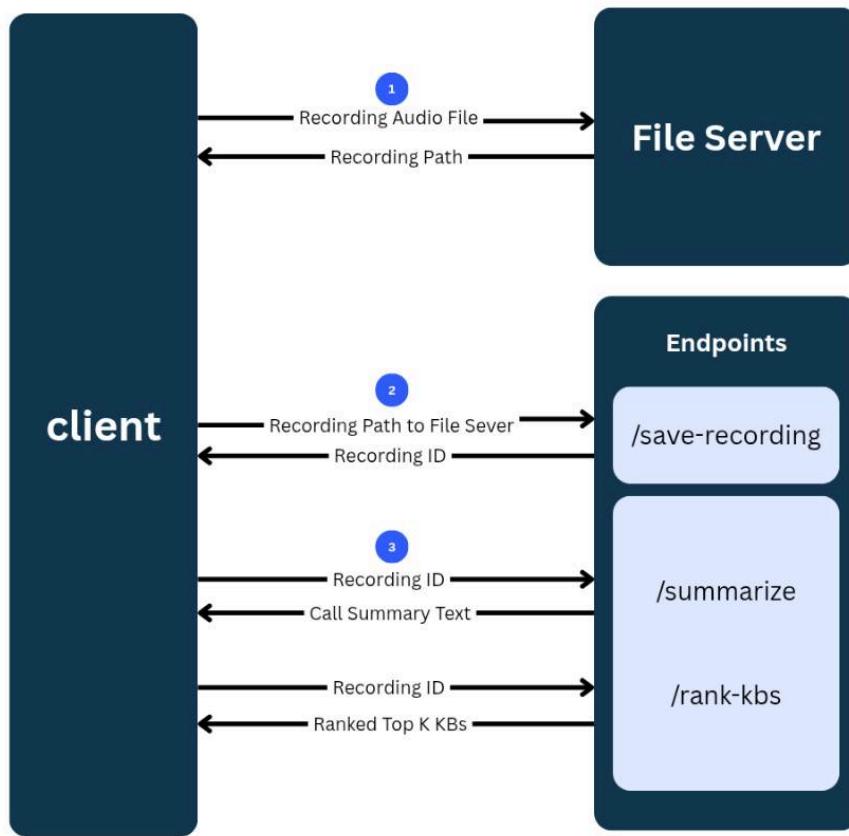
- `model.py`: Handles all logic related to database access, including PostgreSQL queries, Elasticsearch lexical search, and FAISS-based semantic retrieval.
- `inference.py`: Manages communication with inference servers. This includes sending requests to transcription, diarization, summarization, embedding, and image-to-text services.
- `routes.py`: Defines all API endpoints exposed to clients. This is the main entry point for user actions and external integrations.

#### *Client-Facing API Endpoints*

Endpoint	Type	Request	Response	Description
/save-recording	POST	{ "recording_path": "path/to/recording" }	{ "recording_id": 12 }	Save the audio file path into the recording table in the database. Create and return a new recording_id.
/summarize	GET	recording_id	{ "recording_id": 1, "summary": "..." }	Check if the recording already has a transcript. If yes, call summarize_text() on it. If not, call call_transcribe() and speaker_diarization(), store the transcript, then summarize.
/rank-kbs	GET	top_k/recording_id	[ { "kb_id": 1, "kb_name": "abc", "similarity": 0.89 }, ... ]	Same logic as /summarize: if transcript is missing, transcribe and diarize it. Then, embed using embed_text(), compare with

				existing embedded KBs using cosine similarity, and return top-k matches.
/add-kbs	POST	[ { "kb_name": "kb_name", "content": "..." } ... ]	{ "message": "success"}	For each KB entry, optionally use <code>image_to_text()</code> if needed, then <code>embed_text()</code> and insert into the <code>knowledge_base</code> table.
/remove-kbs	DELETE	{ "kb_ids": [1, 2, 3] }	{ "message": "success"}	Remove the specified KB entries from the <code>knowledge_base</code> table.
/re-process-kbs	POST	{ "image-to-text": true, "embed": true }	{ "process_count": 12 }	For all entries in the <code>knowledge_base</code> table, apply <code>image_to_text()</code> if enabled, and <code>embed_text()</code> if enabled. Useful for batch updating or recovering missing embeddings/text.

### 3.5.5 Client Request Flow



The client interacts with the system in the following steps:

#### 3.5.5.1 Upload Audio to File Server

The client begins by uploading an MP3 file to the external file server, which stores the raw audio and returns the storage path. This file will later be used for transcription and analysis.

#### 3.5.5.2 Register Recording

Next, the client sends the file path to the /save-recording endpoint. This registers the recording in the database and returns a recording\_id, which is used for all subsequent operations.

#### 3.5.5.3 Get Summary and Recommended KBS

With the recording\_id, the client can call /summarize to receive a summary of the conversation and /rank-kbs to get top-K relevant knowledge base entries. If no transcript exists, the system will transcribe and diarize the audio before processing.

## Chapter 4

### Results

#### 4.1 Data Synthesis

##### 4.1.1 Knowledge Base process Pipeline

For the task of converting image-based knowledge base content into clean, readable text, we prepared two multimodal models pipeline: Gemini 2.0 Flash and Qwen2.5-VL-3B-Instruct. While both were capable of handling the task, Gemini 2.0 Flash consistently outperformed Qwen in terms of accuracy and clarity. It was more effective at extracting text from dense or low-contrast images and generated cleaner, more coherent explanations. In contrast, Qwen sometimes produced incomplete outputs, missed fine text details, or included artifacts in the form of foreign (non-Thai) language phrases—even when postprocessed using LLaMA for Thai translation refinement.

From an infrastructure standpoint, Qwen requires a minimum of one GPU with at least 14 GB of VRAM to serve the model effectively, followed by additional compute for the LLaMA summarization step. This pipeline increases both latency and resource cost compared to Gemini, which offers a faster, more streamlined experience through its API service.

However, since Gemini 2.0 Flash is a paid API, and NHO's deployment preferences are currently unknown, we prepared processed KB outputs using both models. This approach ensures flexibility, allowing NHO to choose between a cost-effective open-source pipeline (Qwen + LLaMA) or a high-accuracy, API-based solution (Gemini 2.0 Flash) depending on their budget, infrastructure, and operational requirements.

##### 4.1.2 Final Dataset

The final synthetic dataset consists of 57,142 high-quality conversation transcripts, each generated based on 1 to 4 knowledge base (KB) entries. These were grouped across seven configurations (V1–V7), each corresponding to a different data generation strategy or KB combination logic. The breakdown of how many conversations were synthesized under each setting is shown below.

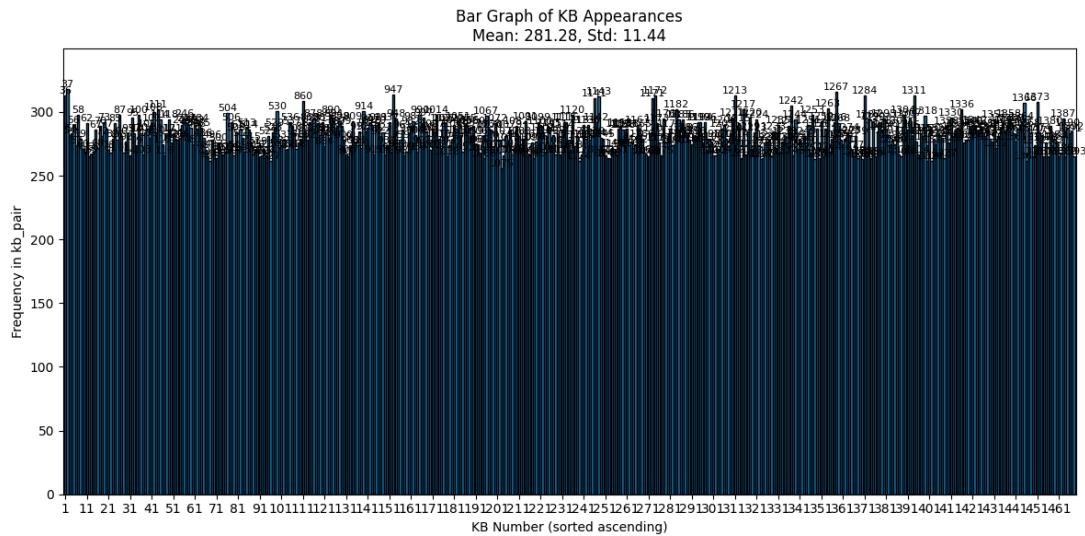
As seen in the table, the majority of conversations included 2 KBs (38.83%), followed by 3 KBs (28.69%), 1 KB (21.07%), and 4 KBs (11.39%). This variation in KB density was designed to both support model analysis and reflect realistic call center scenarios, ranging from simple fact-based inquiries to more complex conversations involving multiple topics.

Group number	V1	V2	V3	V4	V5	V6	V7	Total	Ratio
1kb	4423	0		0	0	0	7620	12043	21.07
2kb	0	700	8933	0	0	6260		22193	38.83
3kb	0	0	0	16395	0	0		16395	28.69
4kb	0	0	0	0	6511	0		6511	11.39
Total								57142	100

#### *4.1.3 Knowledge Base Usage Distribution*

In addition to varying the number of KBs per conversation, we also analyzed how frequently each individual KB appeared across the full dataset. The distribution is visualized in the bar graph above, where each bar represents one of the 468 KB entries used during synthesis.

Overall, the distribution is highly uniform, with each KB appearing an average of 281.28 times and a standard deviation of 11.44. This controlled variance ensures that no single KB dominates the training data, reducing the risk of model bias or overfitting to specific topics.



## 4.2 Model Performance

#### *4.2.1 Speech To Text*

#### 4.2.1.1 monsoon whisper fine-tuned

ผู้พด00: วิจัยล่าสุดของอาเวอร์ดเมลีเคลสกูลได้บอกไว้ว่ามี5พฤติกรรมของการนอน

ผู้ดูแล: ถือว่าเด็กที่ทำให้ผู้ชายอายุยังไม่ได้ถึง

ผู้ดูแลเด็ก: เพิ่มอายุในสิ่ต้องพอดำน้ำเพิ่มอย่างทழงได้ถึง 5 ปี และของผู้หลงเพิ่มถึง 2 จด 5 ปี

ផែទទៅ ៥ពាណិករម្យនៀះ ត្រូវការងារ

ផែលទី០០២: ការអនុវត្តន៍យោង

សំណើលេខ ០០២

សំណើលេខ ០១

## ផ្លូវលេខ ៩០០: ពេជ្ជការណ៍

## ធនធាន និង សេវាឌាម្បី

# ផ្លូវការណ៍របាយការណ៍នៃការងារ

ជុំដឹងទី 01 និងជុំដឹងទី 02 និង

**ផ្សេងៗទី១:** តើយើមខ្សោះនៅក្នុងប្រព័ន្ធដឹកជញ្ជូនដែលបានរាយការណ៍ឡើង

ផ្លូវទី០១ នៃខេត្តកណ្តាល និងខេត្តពោធិ៍ នៅថ្ងៃទី០៩ ខែមី ឆ្នាំ២០១៨

ผู้พูด01: ก็คงเหมือนกับหลบรายการหรือหลบรายการ  
ผู้พูด00: หลบรายการอันที่สื่อตีนมาแล้วรู้สึกสดชื่นและอันสุดท้ายอันที่5เนี่ยคือหลบอย่างต่อเนื่อง  
.....

**ผู้พูด01:** จริงจริงมันก็ไม่ได้ยากมากเนอะในการที่จะทำพฤติกรรมเหล่านี้แต่ว่าก็อาจจะไม่ง่ายอย่างเงียบเรื่องมันอาจจะมีอาชีพบางอาชีพที่ค่อนข้างยากส่วนอีกอัน1อั่งที่เจอก็ได้ค่อนข้างเยอะมากก็คือว่า

สมัยก่อนต้องบอกว่าการ

ผู้พูด01: ประಡະສ່ງຈາດໝາຍນກພິຮານສ່ງໂທຣເລຂໃຫ້ປະທີ່ວ່າຕ້ອງໄປອນຮອນໜຶ່ງສື່ອພິມພົກວ່າຈະໄດ້  
ຊ້າວສາຮອງແຕ່ລະວັນນະ

ຜົດ00: ນກພິຮາບ

ผู้พูด01: แต่ปัจจุบันเนี่ยแค่อย่างโทรศัพท์ไลน์ลงทะเบียนปะหรือว่าถ้าเราอยากจะบอกว่าເອົ້າຍາກຈະຮັງວ່າໄລຍ່ໄວ້ໃຫຍ່ທ່ານມີຄວາມສິ້ນສຸດແບບນີ້ແລ້ວ

ผู้พูด01: ชื่อพฤติกรรมเหล่านี้มันส่งผลให้คนเราเนี่ยกล้ายเป็นโรคสมาธิสั้นหรือว่าเอเด้อชดี

ผู้พูด01: มันก็มีคนเนี่ยยังเรียกว่าคนสมัยก่อนอ่ะไม่ค่อยเข้าใจหรอกว่าโรคสมาธิสั้นคืออะไร

ធ្វើដែលមិនត្រូវបានស្ថិតឡើងទៅក្នុងការរំពេលដែលបានរៀបចំឡើង

แค่แบบ4เปอร์เซ็นต์แต่บอกว่าพอผ่านมาถึงปัจจุบัน20ปีผ่านไปเนี่ยผู้ใหญ่เป็นโรคสมาร์ทสัน

ประมาณเดือน10เปอร์เซ็นต์

## ផ្លូវការណ៍ទី 00: ពេលវេលាដែលបានរាយការណ៍

## ផ្លូវការទំនាក់ទំនង: ការពិន័យនឹងសម្រាប់បច្ចេកទេស

ผู้พูด01: ถือว่าเพิ่มขึ้นเยอะมากเลยโดยเฉพาะในเด็กเนี่ยเขาก็นอกว่ามีคนทำงานนิวัชัยในเมริกาเด็กประมาณ210005100คนในระดับไฮสคูลหรือว่าระดับมัธยมปลายใช่ปั่นเห็นอกว่าแบงค์เด็กเป็น2กลุ่มเท่านั้นกัน

ผู้พูด01: กลุ่ม1เป็นกลุ่มที่เหมือนกับสมัยก่อนอ่านหนังสือธรรมดานะสามารถที่จะใช้แค่มา1ชั่วโมงต่อวันไม่เกินนะรอปางนี้

ผู้ทดสอบ01: กลุ่มที่สองนี้คือเครื่องท่าอะไร? ยกตัวอย่างมา 1 ตัว

ผู้ดูแล 01: ปรากฏว่าเด็กในกลุ่มที่เมื่อ่อนกับว่าใช้สมาร์ทโฟนวันละ 1 แค่ชั่วโมง 1 อั่ง

ผู้พูด01: พบว่าเป็นโรคสมมัครที่สั้นแค่ประมาณ4เปอร์เซ็นต์

ผู้พูด01: แต่ในกลุ่มที่ใช้วอลล์เดอเวอร์ที่อยู่วอลล์ที่อย่างจะใช้ pragmatics เป็นรากฐานสิ้นประมาณ10 เบอร์เซ็นต์

ຜູ້ພຸດ01: ເນັດຖາມເລືອກລ່າງໄດ້ວ່າຄ້າເຮົາວັນທີນີ້ຕິດໂສເຊີຍ

ผู้ดูแล01: คนโพสต์จะไร้กี๊หันไปมองดูคน datum อะไรก็ເວາແຕ່ໄປมองดูมันทำให้เราໄມ້ໄດ້ໂຟກສໄງ

ผู้พูด01: ม้าแต่คิดว่าเอกอนนี้จะทำอะไรคนนั้นจะตอบอะไรหรือเพื่อนจะว่าอะไรปอยางเงียบทำให้เราไม่ได้ฟังเลยก็เลยว่าการที่เราเป็นลิ่งเหล่านี้ทำให้เรามีภาวะเป็นสามาธิสัน្ឋได้ง่ายมาก

ผู้พูด 00: ไม้แต่คิด

ผู้ดูแล: ให้ผู้ก็เป็นเหมือนกัน

ผู้พูด01: แต่ก็มีคนตามว่าอ้วนแล้วถ้าสมมุติว่ามันทำไม่ได้ล่ะจะหมายความว่าให้มันห้อยหน่อyle กันอาจจะจำกดเนะในเฉพาะในเด็กเด็กเนี่ยอาจจะบอกว่าตึ่งเป็นค่อยเรียกว่ามีโปรแกรมชื่นม่วงว่า ลูกคุณอาจจะสามารถเล่นได้เวลาเท่าไหร่จะไรอย่างนี้มีฟอร์มสามารถอนโนเตอร์ได้

ผู้พูด01: หรืออ้างในเรื่องนี้ว่าเป็นต้องทำงานให้ปีกจะมีคนตามว่าເອົ້າຫມອດ້າມນີ້ແບບວ່າອາຫາຮ  
ເສເຣີມຕ້າວໃຫນນັງທີ່ພລວຍໜ່ວຍໄດ້ກົງມີກົດໂຄມເກ3 ທີ່ອີ່ມ້ອຍດ້ານນີ້ຫຍ່າຍປະມຸລສົມອງທີ່ຜູ້ໄລຢູ່  
ແລະເຖິງກົດິນໄດ້ນະໂຫຍດທີ່ຈ່າຍໄດ້ປະຍົກດືອນດອງວ່າໃນແນະໄກໝໍ້ຍຫວູ້ອົງໂກະທີ່ພລວຍໜ່ວຍໄດ້

ผู้พอดี หรือความ

ຜົນດອດ 00 · ທີ່ພອ

ผู้พูด00: แต่วันนี้ก็ต้องให้แนใจนะให้ปรึกษาแพทย์ก่อนแล้วกัน เพราะว่าเด็กกับผู้ใหญ่เนี่ยมีโถสหงัยแต่ละตัวที่ไม่เหมือนกัน

សៀវភៅ

ຜົມດອກ: ແລ້ວແພັນທີ່ກົດຄົງກາແນະໜ້ວງໆໃຫ້ແຕ່ລະຄົນນະເຄອລອນໄປໄຮົມໝາພົທຍໍກ່ອນ

ຜົມບົດ: ໂກງານທີ່ໄດ້ຮັດວຽກ ທີ່ມີຄວາມຮັດວຽກ ໃນລະດົບປະເທດ

ຜົດ00: ແມ່ນອັນຈຸກທີ່ນອຍຝ້າວ

សៀវភៅ ៩៨

From our test, the transcription and diarization result is very accurate. The reason why the same speakers can have multiple lines is because when they paused after a sentence, the system detects a break in the dialogue and starts a new line to maintain clarity and readability in the transcription.

#### 4.2.1.2 pyannote/speaker-diarization-3.1 result

Speakers	DER
Male and Female	1.72%
Male and Male	30.24%
Female and Female	46.73%
Total	10.63%

*Speaker 85 and 87 Both Script 1*  
 {"Actor's ID": "085", "Sex": "Male", "Age": 44}  
 {"Actor's ID": "087", "Sex": "Female", "Age": 30}

*Ground\_truth:*

```
SPEAKER concatenated_audio 1 0.00 3.29 <NA> <NA> SPEAKER_00 <NA> <NA>
SPEAKER concatenated_audio 1 3.79 2.85 <NA> <NA> SPEAKER_01 <NA> <NA>
SPEAKER concatenated_audio 1 7.14 3.73 <NA> <NA> SPEAKER_00 <NA> <NA>
SPEAKER concatenated_audio 1 11.37 2.76 <NA> <NA> SPEAKER_01 <NA> <NA>
SPEAKER concatenated_audio 1 14.63 2.99 <NA> <NA> SPEAKER_00 <NA> <NA>
SPEAKER concatenated_audio 1 18.12 2.41 <NA> <NA> SPEAKER_01 <NA> <NA>
SPEAKER concatenated_audio 1 21.03 5.15 <NA> <NA> SPEAKER_00 <NA> <NA>
SPEAKER concatenated_audio 1 26.68 3.33 <NA> <NA> SPEAKER_01 <NA> <NA>
SPEAKER concatenated_audio 1 30.51 4.11 <NA> <NA> SPEAKER_00 <NA> <NA>
SPEAKER concatenated_audio 1 35.13 2.96 <NA> <NA> SPEAKER_01 <NA> <NA>
```

*Hypothesis:*

```
SPEAKER concatenated_audio 1 0.03 3.32 <NA> <NA> SPEAKER_00 <NA> <NA>
SPEAKER concatenated_audio 1 3.76 2.92 <NA> <NA> SPEAKER_01 <NA> <NA>
SPEAKER concatenated_audio 1 7.15 3.73 <NA> <NA> SPEAKER_00 <NA> <NA>
SPEAKER concatenated_audio 1 11.37 2.72 <NA> <NA> SPEAKER_01 <NA> <NA>
SPEAKER concatenated_audio 1 14.61 2.95 <NA> <NA> SPEAKER_00 <NA> <NA>
SPEAKER concatenated_audio 1 18.09 2.40 <NA> <NA> SPEAKER_01 <NA> <NA>
SPEAKER concatenated_audio 1 21.01 5.21 <NA> <NA> SPEAKER_00 <NA> <NA>
SPEAKER concatenated_audio 1 26.64 3.36 <NA> <NA> SPEAKER_01 <NA> <NA>
SPEAKER concatenated_audio 1 30.52 4.10 <NA> <NA> SPEAKER_00 <NA> <NA>
SPEAKER concatenated_audio 1 35.08 3.00 <NA> <NA> SPEAKER_01 <NA> <NA>
```

*Diarization Error Rate (DER): 1.64%*

**Figure 1.** Example of Speaker Diarization with a Male and Female speaker

*Speaker 93 and 94 Both Script 2*  
 {"Actor's ID": "093", "Sex": "Male", "Age": 30}  
 {"Actor's ID": "094", "Sex": "Male", "Age": 45}

*Ground Truth:*

SPEAKER concatenated\_audio 1 0.00 3.77 <NA> <NA> SPEAKER\_00 <NA> <NA>  
SPEAKER concatenated\_audio 1 4.27 3.16 <NA> <NA> SPEAKER\_01 <NA> <NA>  
SPEAKER concatenated\_audio 1 7.93 3.67 <NA> <NA> SPEAKER\_00 <NA> <NA>  
SPEAKER concatenated\_audio 1 12.11 3.50 <NA> <NA> SPEAKER\_01 <NA> <NA>  
SPEAKER concatenated\_audio 1 16.11 3.85 <NA> <NA> SPEAKER\_00 <NA> <NA>  
SPEAKER concatenated\_audio 1 20.46 3.48 <NA> <NA> SPEAKER\_01 <NA> <NA>  
SPEAKER concatenated\_audio 1 24.44 5.63 <NA> <NA> SPEAKER\_00 <NA> <NA>  
SPEAKER concatenated\_audio 1 30.57 4.46 <NA> <NA> SPEAKER\_01 <NA> <NA>  
SPEAKER concatenated\_audio 1 35.54 4.80 <NA> <NA> SPEAKER\_00 <NA> <NA>  
SPEAKER concatenated\_audio 1 40.84 4.54 <NA> <NA> SPEAKER\_01 <NA> <NA>

*Hypothesis:*

SPEAKER concatenated\_audio 1 0.03 3.76 <NA> <NA> SPEAKER\_00 <NA> <NA>  
SPEAKER concatenated\_audio 1 4.28 3.16 <NA> <NA> SPEAKER\_01 <NA> <NA>  
SPEAKER concatenated\_audio 1 8.01 3.53 <NA> <NA> SPEAKER\_00 <NA> <NA>  
SPEAKER concatenated\_audio 1 12.11 3.53 <NA> <NA> SPEAKER\_01 <NA> <NA>  
SPEAKER concatenated\_audio 1 16.13 3.81 <NA> <NA> SPEAKER\_00 <NA> <NA>  
SPEAKER concatenated\_audio 1 20.50 3.39 <NA> <NA> SPEAKER\_01 <NA> <NA>  
SPEAKER concatenated\_audio 1 24.45 5.64 <NA> <NA> SPEAKER\_00 <NA> <NA>  
SPEAKER concatenated\_audio 1 30.56 14.82 <NA> <NA> SPEAKER\_01 <NA> <NA>

Diarization Error Rate (DER): 15.22%

**Figure 2.** Example of Speaker Diarization between two Male Speakers

*Speaker 98 and 99 Different Scripts*

{"Actor's ID": "098", "Sex": "Female", "Age": 23}  
{"Actor's ID": "099", "Sex": "Female", "Age": 22}

*Ground Truth:*

SPEAKER concatenated\_audio 1 0.00 3.39 <NA> <NA> SPEAKER\_00 <NA> <NA>  
SPEAKER concatenated\_audio 1 3.89 5.29 <NA> <NA> SPEAKER\_01 <NA> <NA>  
SPEAKER concatenated\_audio 1 9.68 3.39 <NA> <NA> SPEAKER\_00 <NA> <NA>  
SPEAKER concatenated\_audio 1 13.57 5.29 <NA> <NA> SPEAKER\_01 <NA> <NA>  
SPEAKER concatenated\_audio 1 19.36 3.39 <NA> <NA> SPEAKER\_00 <NA> <NA>  
SPEAKER concatenated\_audio 1 23.25 5.29 <NA> <NA> SPEAKER\_01 <NA> <NA>  
SPEAKER concatenated\_audio 1 29.04 3.39 <NA> <NA> SPEAKER\_00 <NA> <NA>  
SPEAKER concatenated\_audio 1 32.93 5.29 <NA> <NA> SPEAKER\_01 <NA> <NA>  
SPEAKER concatenated\_audio 1 38.72 3.39 <NA> <NA> SPEAKER\_00 <NA> <NA>  
SPEAKER concatenated\_audio 1 42.61 5.29 <NA> <NA> SPEAKER\_01 <NA> <NA>

*Hypothesis:*

SPEAKER concatenated\_audio 1 0.03 1.01 <NA> <NA> SPEAKER\_00 <NA> <NA>  
SPEAKER concatenated\_audio 1 1.04 0.13 <NA> <NA> SPEAKER\_01 <NA> <NA>  
SPEAKER concatenated\_audio 1 1.18 2.23 <NA> <NA> SPEAKER\_00 <NA> <NA>  
SPEAKER concatenated\_audio 1 4.05 5.16 <NA> <NA> SPEAKER\_01 <NA> <NA>  
SPEAKER concatenated\_audio 1 9.70 9.21 <NA> <NA> SPEAKER\_01 <NA> <NA>  
SPEAKER concatenated\_audio 1 19.37 9.20 <NA> <NA> SPEAKER\_01 <NA> <NA>  
SPEAKER concatenated\_audio 1 29.06 9.20 <NA> <NA> SPEAKER\_01 <NA> <NA>  
SPEAKER concatenated\_audio 1 38.74 9.23 <NA> <NA> SPEAKER\_01 <NA> <NA>

Diarization Error Rate (DER): 37.17%

**Figure 3.** Example of Speaker Diarization between two Female Speakers

#### 4.2.1.3 Speaker Diarization Fine Tuning results

RTTM File (Rich Transcription Time Marked): Marks speaker segments with start/end times and speaker labels

SPEAKER audio21 1 0.000 3.231 <NA> <NA> SPEAKER\_00 <NA> <NA>

SPEAKER audio21 1 3.231 4.007 <NA> <NA> SPEAKER\_01 <NA> <NA>

SPEAKER audio21 1 7.238 3.802 <NA> <NA> SPEAKER\_00 <NA> <NA>

SPEAKER audio21 1 11.040 4.331 <NA> <NA> SPEAKER\_01 <NA> <NA>

SPEAKER audio21 1 15.371 4.631 <NA> <NA> SPEAKER\_00 <NA> <NA>

SPEAKER audio21 1 20.002 4.720 <NA> <NA> SPEAKER\_01 <NA> <NA>

UEM File (User Evaluation Map): Defines valid speech time in the audio file

audio01 NA 0.0 24.721995464852608

#### Fine Tuning Results

Epoch	Diarization Error Rate	Confusion	False Alarm
0	0.1439	0.1404	0.0000
1	0.0889	0.0854	0.0000
2	0.0255	0.0190	0.0031
3	0.0118	0.0084	0.0000
4	0.0118	0.0080	0.0004

#### 4.2.2 KB Classification

The primary objective of our KB classification system is to return the top 10 most relevant knowledge base (KB) entries in response to a call transcript, immediately after the call ends. This task is modeled as a multi-label classification problem, where each transcript may reference 1 to 5 KBs (or more). Since most calls reference multiple KBs and may use varied language to describe similar policies, we optimize for both semantic recall and classification robustness across lexical variation.

Evaluation was conducted using our synthesized conversation dataset (v7), which consists of over 54,000 conversations, with 1–4 gold KBs per transcript ([kb\\_pair](#) field). Ground-truth labels were derived at the KB level, not the chunk level, and evaluation metrics focused on KB-level accuracy.

#### 4.2.2.1. Experimental Setup

We evaluated multiple end-to-end retrieval and classification strategies, using different combinations of:

- Retrieval modes: BM25-only, Dense-only, Hybrid (Dense + Sparse), and Tri-Fusion (Dense + Sparse + ColBERT)
- Embedding models: WangchanBERTa, PhayathaiBERT, SCT-Distil-Phayathai-BGE-M3, and BGE-M3
- Cross-encoder rerankers: finetuned models using BCE loss and hard negatives
- Chunk aggregation & reranking: max-pooled chunk scores aggregated by KB ID, followed by cross-encoder reranking of the top 30 KBs

Each pipeline was evaluated for:

- Recall@K
  - K=3, where ground truth KB = 1
  - K =5, where ground truth KB = 1, 2
  - K=10, where ground truth KB = 1-4
- MRR (Mean Reciprocal Rank)
- Inference latency
- Robustness across multi-KB conversations

#### 4.2.2.3. Retrieval Configurations and Model Results

The table below summarizes key retrieval and reranking configurations:

Method	Embedding Model	ColBERT	Reranker	Recall@10	MR R	Latency (ms)	Notes
BM25 Only	—	✗	✗	61–71%	~0.53	30–50	Fastest, but poor generalization
Dense Only (Siamese)	BGE-M3	✗	✗	76–83%	~0.75	50–150	Strong semantics; misses lexical keywords
Hybrid (Dense + Sparse)	BGE-M3	✗	✗	86–88%	~0.80	200–400	Balanced lexical + semantic match

Tri-Fusion (Dense + Sparse + ColBERT )	BGE-M3			86–88%	~0.80	220–420	Best pre-reranker recall
Hybrid + Reranker	WangchanBERT + WangchanBERT CE		(TorchScript)	87–89%	~0.82	220–450	Good for CPU-only sites
Hybrid + Reranker	BGE-M3 + BGE-Reranker-v2		(Triton)	<b>88–91%</b>	<b>0.85</b>	270–500	Best overall accuracy
Hybrid + Reranker	Distil-Phayathai-BGE + Distil-Phayathai-BGE-CE		(TorchScript)	87–89%	~0.81	200–430	Lightweight, ideal for CPU inference

These results show that **hybrid retrieval with BGE-M3 and reranking using bge-reranker-v2-m3** consistently outperformed other configurations in Recall@10 and MRR, particularly for complex multi-KB conversations.

#### 4.2.2.4. Key Observations

- **Semantic chunking** (hybrid recursive +  $\Delta$ -cosine) significantly improved context retention, especially for KBs exceeding 10,000 tokens.
- **ColBERT vectors**, when used in tri-fusion retrieval (dense + sparse + token-level match), provided subtle improvements in disambiguating near-synonyms or contextually related KBs (e.g., similar drug policies).
- **Cross-encoder reranking** contributed an additional +3 to +5 pp improvement over hybrid-only retrieval, confirming its importance for fine-grained ranking.
- **WangchanBERT-CE**, despite using a smaller context (512 tokens), performed well due to its strong Thai lexical base and careful chunk tuning.
- **BGE-M3 + reranker**, while GPU-intensive, showed the best tradeoff between long-range context, semantic understanding, and final top-10 accuracy.

#### 4.2.2.5 Final Pipeline Selection

Our final production configuration balances latency, cost, and classification accuracy:

Component	Choice	Justification
Chunking	Hybrid (Recursive + Semantic Cut)	High coherence, 4× recall vs. recursive

Embedding Model	BGE-M3	8192-token input, supports dense + sparse + ColBERT
Indexing	FAISS (dense), Elastic (sparse)	Fast ANN + BM25 retrieval
Retrieval	Tri-Fusion (dense + sparse + ColBERT)	Best coverage pre-rerank
Reranking	bge-reranker-v2-m3 (LoRA)	High accuracy, supports 4k context
Aggregation	Max-pool chunk score → KB score	Prevents large KBs from dominating
Fallbacks	Distil-CE or Wangchan-CE	TorchScript for CPU inference in clinics

#### 4.2.2.6 Conclusion

Through extensive experimentation, we achieved **Recall@10 up to 91%**, with MRR nearing 0.85 using our final tri-fusion + reranker configuration. The system generalizes well across multi-KB conversations and supports both GPU-heavy and CPU-light deployments.

This classification pipeline forms the backbone of our KB recommendation system, ensuring timely, accurate KB suggestions for post-call workflows at NHSO.

#### 4.2.3 Summarization

To accommodate different deployment preferences and cost constraints, we prepared two summarization models: LLaMA-3.2B-Instruct and Qwen2.5-0.5B-Instruct. While LLaMA-3.2B achieves higher ROUGE scores, likely due to its greater robustness to diverse input patterns, Qwen2.5-0.5B offers a lighter-weight alternative that may be more practical for NHSO's long-term deployment needs.

Although LLaMA performs better on paper, a manual inspection of the summaries shows that Qwen often produces language that is simpler and easier to read, with minimal differences in content accuracy for most common use cases. Given that Qwen is also around half the size, it requires fewer resources and may be a more efficient choice for real-world applications with limited compute capacity.

Metric	LLaMA-3.2B (Pretrained Baseline)	LLaMA-3.2B (Fine-Tuned)	Qwen2.5-0.5B (Fine-Tuned)
ROUGE-1	0.3393	0.7479	0.6974
ROUGE-2	0.1898	0.5796	0.5200
ROUGE-L	0.3198	0.7240	0.6729

Here is example of our fine-tuned model result:

4.3.	<b>Target 1</b>	<p>ผู้ป่วยโทรศัพท์เข้ามาสอบถามเกี่ยวกับปัญหาการเบิกค่ารักษาพยาบาลทางบัตรทองจากการทำฟันที่โรงพยาบาลตามสิทธิ โดยโรงพยาบาลแจ้งว่าการเบิกติดรหัส C438 เจ้าหน้าที่ได้สอบถามข้อมูลส่วนตัวของผู้ป่วยเพื่อตรวจสอบความถูกต้อง และแจ้งว่าสาเหตุอาจเกิดจากข้อมูลในระบบไม่ตรงกัน ซึ่งโรงพยาบาลจะต้องตรวจสอบและแก้ไขข้อมูลในระบบ e-Claim ให้ถูกต้องก่อนจึงจะสามารถเบิกได้ ผู้ป่วยไม่ต้องดำเนินการใดๆ เพิ่มเติม เพียงแจ้งปัญหาให้โรงพยาบาลทราบ/หากอกจากนี้ ผู้ป่วยยังสอบถามเกี่ยวกับสิทธิการรักษาฟันคุด รักษารากฟัน และทำฟันปลอมใหม่ เจ้าหน้าที่ได้แจ้งว่าสิทธิบัตรทองครอบคลุมการรักษาเหล่านี้ทั้งหมด แต่ต้องได้รับการอนุมัติจากทันตแพทย์ก่อนว่ามีความจำเป็นทางการแพทย์ และสำหรับการทำฟันปลอมใหม่ จะครอบคลุมเฉพาะฟันปลอมฐานพลาสติกเท่านั้น โดยต้องมีใบรับรองจากทันตแพทย์ยืนยันว่าฟันปลอมเดิมไม่สามารถซ่อมแซมได้หากผู้ป่วยรับทราบข้อมูลและจะไปสอบถามรายละเอียดเพิ่มเติมที่โรงพยาบาล เจ้าหน้าที่ได้แจ้งช่องทางการติดต่อเพิ่มเติมหากมีห้องสังสัยเพิ่มเติมคือสายด่วน สปสช. 1330 หรือเว็บไซต์ของ สปสช.</p>
	<b>Qwen2.5-0.5B-Instruct</b>	<p>ผู้ป่วยโทรศัพท์เข้ามาสอบถามเกี่ยวกับการเบิกค่ารักษาพยาบาลภายใต้สิทธิบัตรทอง กรณีที่โรงพยาบาลแจ้งว่าไม่เบิกติดรหัส C438 เจ้าหน้าที่ได้แนะนำให้โรงพยาบาลลดตรวจสอบข้อมูลในระบบ e-Claim ให้ตรงกับข้อมูล สปสช. ก่อนทำการเบิกใหม่ หากจากนี้ ผู้ป่วยยังสอบถามเกี่ยวกับการฝ่าฟันคุด การรักษารากฟัน และการทำฟันปลอมใหม่เนื่องจากฟันปลอมเดิมแตก เจ้าหน้าที่ได้แจ้งว่าการฝ่าฟันคุดและการรักษารากฟันเป็นสิทธิประโยชน์ที่ครอบคลุมภายใต้บัตรทอง แต่ต้องได้รับการอนุมัติจากทันตแพทย์ก่อน และการทำฟันปลอมใหม่จะต้องมีใบรับรองจากทันตแพทย์เพื่อยืนยันความจำเป็นในการทำใหม่ ผู้ป่วยรับทราบข้อมูลและจะไปสอบถามรายละเอียดเพิ่มเติมที่โรงพยาบาล เจ้าหน้าที่ได้แจ้งช่องทางการติดต่อเพิ่มเติมหากมีห้องสังสัยเพิ่มเติมคือสายด่วน สปสช. 1330 หรือเว็บไซต์ของ สปสช.</p>
	<b>llama3.2-1b-Instruct</b>	<p>ผู้ป่วยโทรศัพท์เข้ามาสอบถามเกี่ยวกับปัญหาการเบิกค่ารักษาพยาบาลทางบัตรทองตามสิทธิบัตรทอง โดยไม่เบิกติดรหัส C438 เจ้าหน้าที่ได้แจ้งว่าสาเหตุอาจเกิดจากข้อมูลไม่ตรงกันในระบบ และแนะนำให้โรงพยาบาลเช็คเรื่อง โดยให้โรงพยาบาลเป็นผู้ดำเนินการแก้ไขข้อมูลและส่งคำขอเบิกใหม่ ผู้ป่วยยังสอบถามเกี่ยวกับการฝ่าฟันคุด การรักษารากฟัน และการทำฟันปลอมใหม่ เจ้าหน้าที่ได้แจ้งว่าการฝ่าฟันคุด และรักษารากฟันสามารถใช้สิทธิบัตรทองได้ แต่ต้องได้รับการอนุมัติจากทันตแพทย์ก่อน ส่วนการทำฟันปลอมใหม่ก็สามารถใช้สิทธิได้เช่นกัน แต่ต้องมีใบรับรองจากทันตแพทย์ยืนยันว่าฟันปลอมเดิมไม่สามารถซ่อมแซมได้ ผู้ป่วยรับทราบและจะไปสอบถามรายละเอียดเพิ่มเติมที่โรงพยาบาล</p>
	<b>Target 2</b>	<p>คุณธีรภัทร โชคกุล พหารเกณฑ์ สังกัดกรมแพทย์ทหารเรือ ได้โทรศัพท์สอบถามเกี่ยวกับสิทธิการรักษาพยาบาล เนื่องจากได้รับบาดเจ็บจากการฝึก (แพลงฟร้าที่ขาและมือ) และกำลังพักรักษาตัวที่บ้านเกิดในจังหวัดสุราษฎร์ธานี เจ้าหน้าที่ได้ตรวจสอบสิทธิและแจ้งว่าคุณธีรภัทรมีสิทธิ SN001 ซึ่งสามารถเข้ารับการรักษาที่โรงพยาบาลของรัฐและจังหวัดสุราษฎร์ธานี ได้โดยไม่ต้องมีหนังสือส่งตัว และค่ารักษาพยาบาลจะถูกยกผ่านระบบ e-Claim โดยอัตโนมัติ โดยใช้เงินเดือนของตนที่ได้รับจากการฝึกหัดสุราษฎร์ธานี แต่เจ้าหน้าที่ได้ตรวจสอบสิทธิและแจ้งว่าคุณธีรภัทร มีสิทธิ SN001 ในการรักษาได้ที่โรงพยาบาลของรัฐและจังหวัดสุราษฎร์ธานี โดยไม่ต้องมีหนังสือส่งตัว เพียงแสดงบัตรประจำตัวและยื่นสิทธิหลักประกันสุขภาพแห่งชาติ ทางโรงพยาบาลจะดำเนินการเบิกค่าใช้จ่ายผ่านระบบ e-Claim โดยอัตโนมัติ และหากเป็นกรณีฉุกเฉินหรือรับการส่งต่อ ก็สามารถใช้สิทธิได้ทันที นอกเหนือไปนี้ เจ้าหน้าที่ยังแจ้งว่าหากมีการรักษาต่อเนื่อง เช่น ยาแก้ปวด สามารถรับได้ที่ร้านยาคุณภาพใกล้บ้านโดยแสดงบัตรประจำตัว ไม่จำเป็นต้องกลับไปที่หน่วยบริการตามสิทธิ คุณธีรภัทรรับทราบข้อมูลและไม่มีประเด็นใดที่ต้องส่งต่อเพิ่มเติม</p>
	<b>Qwen2.5-0.5B-Instruct</b>	<p>คุณธีรภัทร โชคกุล ซึ่งเป็นพหารเกณฑ์สังกัดกรมแพทย์ทหารเรือ โทรศัพท์สอบถามเกี่ยวกับสิทธิการรักษาพยาบาล เนื่องจากประสนลูบดีเหตุขณะฝึกและมีแพลงฟร้าที่ขา พร้อมมีไฟเขียวที่ได้รับการรักษาที่โรงพยาบาลสุราษฎร์ธานี และปัจจุบันพักรักษาตัวอยู่ที่บ้านเกิดในจังหวัดสุราษฎร์ธานี/ท่ามกลางเจ้าหน้าที่ได้ตรวจสอบสิทธิและแจ้งว่าคุณธีรภัทร สามารถใช้สิทธิ SN001 ใน การรักษาได้ที่โรงพยาบาลของรัฐและจังหวัดสุราษฎร์ธานี โดยไม่ต้องมีหนังสือส่งตัว เพียงแสดงบัตรประจำตัวและยื่นสิทธิหลักประกันสุขภาพแห่งชาติ ทางโรงพยาบาลจะดำเนินการเบิกค่าใช้จ่ายผ่านระบบ e-Claim โดยอัตโนมัติ และหากเป็นกรณีฉุกเฉินหรือรับการส่งต่อ ก็สามารถใช้สิทธิได้ทันที นอกเหนือไปนี้ เจ้าหน้าที่ยังแจ้งว่าหากมีการรักษาต่อเนื่อง เช่น ยาแก้ปวด สามารถรับได้ที่ร้านยาคุณภาพใกล้บ้านโดยแสดงบัตรประจำตัวและยื่นสิทธิ ไม่มีประเด็นที่ต้องส่งต่อ แต่หากคุณธีรภัทรมีห้องสังสัยเพิ่มเติม สามารถติดต่อศูนย์ภาษาไทยด่วน 1330 ได้ตลอด 24 ชั่วโมง</p>
	<b>llama3.2-1b-instruct</b>	<p>คุณธีรภัทร โชคกุล พหารเกณฑ์ ได้โทรศัพท์สอบถามเกี่ยวกับสิทธิการรักษาพยาบาลภายใต้ระบบหลักประกันสุขภาพแห่งชาติ (สปสช.) โดยแจ้งว่าได้รับบาดเจ็บจากการฝึกและต้องการทราบว่าสามารถเข้ารับการรักษาที่โรงพยาบาลสุราษฎร์ธานี ได้หรือไม่ เจ้าหน้าที่ได้ตรวจสอบสิทธิและยืนยันว่าคุณธีรภัทรมีสิทธิ SN001 ซึ่งเป็นสิทธิสำหรับพหารเกณฑ์สังกัดกรมแพทย์ทหารเรือ และสามารถเข้ารับการรักษาที่โรงพยาบาลของรัฐได้โดยไม่ต้องมีหนังสือส่งตัว โดยแสดงเพียงบัตรประจำตัวและยื่นสิทธิ สปสช. ได้ทันที เจ้าหน้าที่ได้แจ้งว่าการเบิกจ่ายจะดำเนินการผ่านระบบ e-Claim โดยอัตโนมัติ และสำหรับค่ารักษาพยาบาลในจังหวัดสุราษฎร์ธานีจะถูกเรียกเก็บจากกองทุนหลักประกันสุขภาพตามอัตรา Point system นอกจากนี้ หากคุณธีรภัทรต้องการเยี่ยมชมสถานที่ที่ต้องใช้ต่อเนื่อง สามารถรับได้ที่ร้านยาคุณภาพใกล้บ้านโดยแสดงบัตรประจำตัวได้เช่นกัน ปัจจุบันยังไม่มีประเด็นที่ต้องส่งต่อเพิ่มเติม</p>

## Deployment Performance

Since no call recordings were received from NHSO over the past year, we were unable to test the system on actual user data. To address this limitation, we created our own evaluation dataset by calling the NHSO contact center and recording five real conversations with their agents. One of these recordings, which we

used as a key test case, is 8.37 minutes long and covers a real inquiry about the availability and usage conditions of medications under the 30-baht scheme at certified pharmacies.

You can listen to the sample test conversation here: [NHSO Test Call Recording](#)

The conversation includes multiple turns involving eligibility checks, service procedures, and step-by-step instructions for using the NHSO website to locate participating pharmacies.

#### 4.3.1 Speaker Diarization and Speech-to-Text

For speaker diarization, we used the Pyannote inference framework along with a pretrained speaker-diarization model. The diarization process took approximately 7.18 seconds to process an 8.37-minute (502.2 seconds) audio file, which equates to around 70 milliseconds per second of audio. This runtime is acceptable, though Pyannote does not fully utilize GPU capacity, limiting its performance. Additionally, the diarization output included several incorrect segments—especially those under 0.5 seconds, which were mostly background noise or silence. These were filtered out in post-processing by discarding any segment shorter than 0.5 seconds. Aside from this, the diarization output was generally usable for downstream processing.

For speech-to-text, we deployed a 764M-parameter Thai Whisper model using the VLLM framework, which supports asynchronous inference. This allowed us to transcribe all valid diarized segments in parallel, resulting in a total processing time of 9.27 seconds, or approximately 54 seconds of audio processed per second. GPU utilization during inference was around 60%, suggesting potential headroom for performance tuning.

However, the transcription output still showed notable limitations in accuracy. The model struggled with specialized vocabulary, proper names, and uncommon phrasing, particularly when speakers used different pronunciations or unclear enunciation. Frequent errors included homophone confusion and mistakes between visually or phonetically similar words—such as confusing “0บริการ” (zero service) with “ศูนย์บริการ” (service center).

These issues largely stem from the fact that the model was not fine-tuned on real-world Thai contact center data, which limits its robustness. As a result, it tends to mishandle specialized terms, ambiguous phrasing, and domain-specific expressions that commonly appear in actual user interactions. This underscores the need for domain-specific fine-tuning to improve transcription accuracy in production use.

Below is the final transcript generated by the pipeline:

ผู้พูด01: 1330สวัสดีค่ะประจำวันรับสายยินดีให้บริการค่ะ

ผู้พูด00: สวัสดีครับพอดีผมมีคำถามเกี่ยวกับเรื่องร้านยา30บาทที่ครับอย่างทราบว่าถ้าผมจะไปซื้อยาพอดี  
ผมสามารถเช็คได้ในเครือข่ายไหนที่ร่วมรายการครับ

ผู้พูด01: ตอนนี้เลือกสายคุณกับสิทธิ์ใส่ตั้งหรือไม่คะใช่ค่ะราภานประเมินการรักษาของญาติท่านข้อมูลติด  
ว่าความถูกต้องของข้อมูลลักษณะบัญชีกับสิทธิ์ใส่ตั้งได้รับสิทธิ์ประโยชน์ที่หน่วยบริการประเมินคุณและ  
ประจำ0บริการสาธารณสุข504พัดเปรียบส่งต่อโรงพยาบาลตามศิลป์จังหวัดกรุงเทพฯว่าที่คุณกับสิทธิ์สอน  
ถ้าหมายถึงว่าการเข้ารับบริการที่ร้านขายยาคุณภาพของฉันถูกต้องหรือไม่ค่ะใช่ครับ สำหรับการให้บริการ  
ร้านขายยาคุณภาพจะคงผลกัดสิทธิ์จะเป็นการรับยาในกรณีมีภาวะเจ็บป่วยและซึ่งปกติถ้าหากว่ามีภาวะ  
เจ็บป่วยสามารถยื่นบัตรประชาชนแล้วก็ติดต่อเข้ารับบริการที่ร้านขายยาที่เข้าร่วมโครงการเพื่อใช้บริการได้เลย  
คือจะไม่มีค่าใช้จ่าย แต่หากว่าคุณภาคติดต้องการเป็นในลักษณะของการร้องขอหรือซื้อยาเองนั้นจะต้อง  
ชำระค่าใช้จ่ายเองแต่ส่วนยานะมีหรือไม่นั้นอาจจะต้องโทรสอบถามกับทางร้านขายยาโดยตรงค่ะ

ผู้พูด00: คือหมายถึงว่าต้องมีใบรับรองแพทย์อะไรประมาณนี้ใช่ไหมครับ

ผู้พูด01: หมายถึงว่าที่คุณกษิติสอนถ้าถามเพอร์มีต้องการซื้อยาประเภทใดค่ะ

ผู้พูด00: ก็ยาแก้ปวดทั่วไปของรับอะไรประมาณนี้

ผู้พูด01: ตอนนี้มีภาวะเจ็บป่วยอยู่หรือไม่ค่ะ

ผู้พูด00: อ้อก็ไม่มีครับพอดีอาจจะเป็นไข้nidหน่อย

**ผู้พูด01:** ถ้าเข่นนั้นอาจจะเป็นการแนะนำในเรื่องของการใช้สิทธิ์จะดีกว่าจะได้ไม่ต้องเสียอัตค่าใช้จ่ายในการเข้ารับบริการเพียงแต่ว่าให้ทางคุณภาศัยดูนะบัตรประชาชนแล้วมาติดต่อที่ร้านขายยาที่เขาร่วมโครงการพร้อมแจ้งอาการของกับทางเพศรัชกรประเมินให้ดังนั้นทางเพศรัชกรจะมีการจัดยาให้ตามอาการที่เป็นเลย

**ผู้พูด00:** อืมแล้วทางร้านยาที่เข้าร่วมรายการสามารถด้วยทางไหนบ้างอ่ะค่ะ

**ผู้พูด01:** กรณุณภานุสາมารถตรวจส่องรายชื่อผ่านหน้าเว็บไซต์ของทางสปสชได้เลยนะครับก็จะมีการแบ่งตามเขตพื้นที่ที่พักอาศัยแต่ก็สามารถตรวจสอบได้เช่นเดียวกันพร้อมทั้งจะมีหมายเลขอรหัสพทที่ติดต่อพร้อมแผนที่แสดงให้สร้างข้อมูลนะครับด้วยตัวเองเพิ่มเติมให้มั่ยค่ะ

**ผู้พูด00:** ได้ครับขออีกที่ได้ครับ

**ผู้พูด01:** เช่นนั้นคุณจะติสະตะวกเป็นการเข้าผ่านโทรศัพท์มือถือใช้ใหม่ค่ะ

**ผู้พูด00:** อ่าผ่านแล็บท้อปอ่ะครับ

**ผู้พูด01:** หายจะเท่านั้นให้พิมพ์หาในกูเกิลว่าสปสชว่าค่ะ

**ผู้พูด00:** สปสชรับผิดชอบ

**ผู้พูด01:** แล้วก็ให้เข้าสู่เว็บไซต์ได้เลยค่ะ

**ผู้พูด00:** เข้ามาล่ะครับตอนนี้

**ผู้พูด01:** จากนั้นนะครับคือให้เมื่อลามาด้านล่างนะจะมีเมนูให้เลือกทำรายการอยู่ค่ะ

**ผู้พูด00:** ก็เป็น

**ผู้พูด01:** จะเป็นเมนูค้นหาร้านยาค่ะซึ่งข้อมูลใหม่ค่ะ

**ผู้พูด00:** ที่เป็นหน่วยบริการหรือเปล่าครับ

**ผู้พูด01:** จะเป็นหน่วยข้อมูลหน่วยบริการก็ได้ค่ะเลือกอย่างโดยย่างได้เลยไม่ทราบว่าตอนนี้กดเข้ามาที่เมนูไหนค่ะ

**ผู้พูด00:** หน่วยบริการนะครับ

**ผู้พูด01:** ก็คือค้นหาเดิมบริการ**30นาทีรับสายตรงนั่นนะ**

**ผู้พูด00:** ใช่ค่ะ

**ผู้พูด01:** ปัจจุบันก็เคยอยู่ในพื้นที่ของจังหวัดกรุงเทพอยู่แล้วถูกต้องใหม่ค่ะ

**ผู้พูด00:** ใช่ครับผม

**ผู้พูด01:** ใช้เท่านั้นให้คุณกดสิรินะจะเลือกไปที่หน่วยนัดกรรมแพทย์ก็จะมีห้องคลิกที่นี่

**ผู้พูด00:** พอดีหาไม่เจอก็รับไม่ได้ไทยอยู่ตรงไหน

**ผู้พูด01:** จะเป็นตอนนี้ที่หน้าจอแสดงเป็นระบบค้นหาหน่วยบริการในระบบหลักประกันสุขภาพแห่งชาติหน้าจอจะเป็นพื้นหลังออกสีเทาหรือไม่ค่ะ

**ผู้พูด00:** พื้นหลังเป็นแมลงแผลที่จะครับแล้วก็มีกล่องข้อมูลให้ติด

**ผู้พูด01:** ข้อมูลให้ติดอ่าถ้าแค้นน้อยรบกวนกลับไปหน้าหลังอีกก่อนจะเข้าหน้าในโมดัลกล่าวได้มั้ยค่ะ

**ผู้พูด00:** ได้ครับอุ่นมาแล้วครับ

**ผู้พูด01:** จากนั้นนะครับคือให้ไปที่เมนูที่เขียนว่าหนูค้นหาหน่วยบริการ**30นาทีรักษาทุกที่ค่ะ**

**ผู้พูด00:** อุ่นมาเจ็บแล้วครับ

**ผู้พูด01:** จากนั้นเมื่อเข้ามาในหน้ารายการตั้งกล่าวแล้วให้คุณกล่าว**11**จะเลือกไปที่แบบสีเหลืองที่เขียนว่าหน่วยนัดกรรมแพทย์ที่นี่

**ผู้พูด00:** ครับผมคลิกแล้วค่ะ

**ผู้พูด01:** จากนั้นให้กดเลือกเขตที่พักอาศัยได้เลยจะแล้วก็สามารถเลือกแขวงได้หรือว่าจากกองจากเขตอย่างเดียวก็ได้

**ผู้พูด00:** เป็นเขตที่ตามบัตรประชาชนใช้ใหม่ครับ

**ผู้พูด01:** ที่พักอยู่สุดท้ายใช้ใหม่ครับตามพักอาศัยจริงได้เลยจะค่ะคนกลับในดิน

**ผู้พูด00:** ได้ครับໂອເຄຣັບພມ

**ผู้พูด01:** จากนั้นให้มาเลือกให้มาเลือกพริกในกล่องสีเหลี่ยมร้านยาคุณภาพ

**ผู้พูด00:** ร้านใหญ่คุณภาพนะค่ะ

**ผู้พูด01:** แล้วก็จากนั้นนะจะสามารถกดค้นหาจากที่อยู่ค่ะ

**ผู้พูด00:** เลขข้อมูลใส่ไว้

**ผู้พูด01:** ขึ้นข้อมูลถึงใหม่ค่ะ

**ผู้พูด00:** อ้อเข็นละค่ะ

**ผู้พูด01:** ซึ่งถ้าหากว่าในรายการที่แสดงก็คือจะมีเป็นหมายเลขโทรศัพท์ติดต่อพร้อมพิกัดให้สามารถตรวจ

สอบถามข้อมูลได้แนะนำหากว่าคุณการศึกษาต้องการใช้สิทธิในการเข้ารับบริการขอให้โทรเข้าไปนัดหมายแล้วก็สอบถามก่อนเนื่องจากว่าทางร้านขายยาบางร้านอาจจะมีเวลาของทางเพศรัชกรที่แตกต่างกันนะจะได้ไม่เดินทางแล้วเสียเวลาและ

ผู้พูด00: ได้ครับก็คือต้องบอกอาการอะไรพอกนี้ก่อนใช่ไหมครับปรึกษา ก่อน

ผู้พูด01: แต่สามารถปรึกษาเบื้องต้นก่อนได้ค่ะว่าหากอาการเหล่านี้หรือบ้านสอบถามว่าต้องการไปใช้สิทธิในการรับยาของสิทธิบัตรทองจะสามารถไปติดต่อค่ารับบริการเป็นวันใหม่ในบางนະคะนบังคับจะหลังจากนั้นถ้าหากว่าคุณเกิดเต็จไข้บริการไปแล้วแล้วอาการยังไม่ดีขึ้นสามารถยื่นบัตรประชาชนเข้ารับบริการที่สูงสาธารณสุข5043เขื่อนเพื่อให้แพทย์ประเมินอาการเพิ่มเติมให้นะคะกีก็จะมีการส่งต่อไปให้ตามขั้นตอนค่ะ

ผู้พูด00: อ่อได้ค่ะ

ผู้พูด01: ละนี่มีอาการมากก่อนแล้วค่ะ

ผู้พูด00: ก็เพิ่งมีลักษณะ

ผู้พูด01: เลี้ยงกันนະคะถ้าเช่นนั้นออกย้ายทุกช่วงเข้าลงโทรศัพท์สอบถามร้านขายอย่างดีแล้วก็ยื่นบัตรประชาชนในการเข้ารับบริการได้เลยนะคะ

ผู้พูด00: ได้ครับขอบคุณมากครับ

ผู้พูด01: ยินดีให้บริการค่ะจะป่วยเร่งด่วนด้านอื่นเพิ่มเติมนอกเหนือจากนี้ไหมค่ะ

ผู้พูด00: ไม่มีลักษณะ

ผู้พูด01: มีค่ารองค่าตามเลิกเติมເຄືອຂໍ້ມູນຕະຫຼາດ

ผู้พูด00: ໂວເຄລະຮັບຜົນ

ผู้พูด01: ดังนี้ข้อมูลลงสารเพิ่มเติมติดต่อ1330ได้กันตลอด204ชั่วโมงค่ะกว่าล่องทรายหุ้นเกษตรกิจสະຫວະประเมินการให้บริการตัวยหรือไม่ค่ะ

ผู้พูด00: สะດວກค่ะ

ผู้พูด01: ขอบพระคุณมากค่ะขอบคุณที่ใช้บริการ1330ດູແລວກ່າສຸຂພາບດ້ວຍນະຄະສວັສດີຕະ

ผู้พูด00: สວັສດີຕະ

#### 4.3.2 Summarization

We evaluated the performance of our summarization module using the VLLM inference framework. The response time was highly efficient: after averaging over three runs, Qwen2.5-0.5B generated a summary in approximately 1.46 seconds, while LLaMA-3.2B completed the task in 1.05 seconds. Both models operated at around 80% GPU utilization, indicating good inference efficiency with headroom for additional optimization.

However, despite the strong runtime performance, the quality of summaries on real audio data showed some weaknesses. This is primarily due to a mismatch between training and deployment conditions. The models were fine-tuned on clean, synthetic transcripts that contained no misspellings or noise. When applied to real-world data, the speech-to-text outputs often include transcription errors, informal phrasing, or inconsistent word boundaries, which were not seen during training.

As a result, the summaries sometimes became less accurate or misleading, occasionally highlighting incorrect keywords or missing important details. This demonstrates that real-world robustness remains limited without further domain adaptation, and future improvements should focus on fine-tuning the summarization models with noisy, real-life transcripts.

Below is a comparison of summaries generated by Qwen and LLaMA on the same real transcript:

LLaMA:

ผู้ป่วยโทรศัพท์เข้ามาสอบถามเกี่ยวกับการใช้สิทธิ 30 นาทีที่ร้านขายยาคุณภาพ โดยผู้ป่วยเข้าใจผิดว่าสามารถซื้อยาได้โดยไม่ต้องมีใบรับรองแพทย์ เจ้าหน้าที่ได้อธิบายว่าการใช้สิทธิ 30 นาทีที่ร้านขายยาคุณภาพนั้น จะต้องมีภาวะเจ็บป่วยและแสดงบัตรประชาชนเพื่อประเมินอาการโดยแพทย์ เท่านั้น ผู้ป่วยสอบถามถึงวิธีการค้นหาร้านยาที่เข้าร่วมโครงการ เจ้าหน้าที่ได้แนะนำขั้นตอนการค้นหาผ่านเว็บไซต์ สปสช. โดยให้เลือกค้นหาหน่วยบริการที่ต้องการ และแนะนำให้โทรศัพท์ติดต่อเพื่อสอบถามเวลาทำการและเตรียมตัวก่อนเข้ารับบริการเพื่อประเมินอาการเพิ่มเติมหากจำเป็น ผู้ป่วย

ได้รับข้อมูลและเข้าใจวิธีการใช้สิทธิเบื้องต้นแล้ว ยังไม่มีประเด็นที่ต้องส่งต่อในขณะนี้

Qwen:

ผู้ป่วยโทรศัพท์เข้ามาสอบถามเกี่ยวกับการใช้สิทธิร้านขายยา 30 บาท เนื่องจากมีอาการเจ็บป่วยเล็กน้อยและต้องการทราบว่าสามารถครัวบยาแก่ปวดทั่วไปได้หรือไม่ เจ้าหน้าที่ได้แนะนำให้ผู้ป่วยค้นหาร้านยาคุณภาพที่เข้าร่วมโครงการผ่านเว็บไซต์ สปสช. โดยให้ไปที่เมนูหน่วยบริการ หน่วยนัด格林แพทย์ เลือกเขตที่พักอาศัย และเลือกแขวง จากนั้นเลือกไปยังร้านยาคุณภาพและกดค้นหา จากนั้นเลือกไปยังเมนูค้นหาหน่วยบริการ ผู้ป่วยสามารถค้นหาได้สำเร็จและเข้าใจวิธีการค้นหาเบื้องต้นแล้ว เจ้าหน้าที่แนะนำให้ผู้ป่วยโทรศัพทนัดหมายกับร้านยาอีกครั้งก่อนไปใช้บริการเพื่อความสะดวกและแม่นยำในการประเมินอาการ หากอาการไม่ดีขึ้นหลังใช้บริการ เจ้าหน้าที่แนะนำให้ยื่นบัตรประชาชนที่โรงพยาบาลตามสิทธิเพื่อประเมินเพิ่มเติม ผู้ป่วยรับทราบและขอบคุณในคำแนะนำ ไม่มีประเด็นที่ต้องส่งต่อเพิ่มเติมในขณะนี้

#### 4.3.3 KB Classification

Due to the loss of our best configuration and all relevant files, as well as Apex not allowing Docker access, the pipeline deployed for the test run below is that of components that may have subpar performance, but do not require docker nor sudo for setup. Tokenization was done with PyThaiNLP's newmm, indexed with Elasticsearch and used a FAISS HNSW instead of Milvus, thus retrieval from dense embeddings only instead of dense + sparse. There was also no reranking step with bge-reranker-v2-m3 finetuned with ColBERT vecs as the best version of our pipeline. The top 10 KBs are ranked directly by aggregating the scores of the top 100 retrieved chunks via BM25 from the lexical index, and from FAISS with max pooling. Response time is still acceptable, at 1.719 seconds, with KBs of relevant themes retrieved. The call is about ร้านยา30บาท, under the policy “30 บาทรักษาทุกที่”. KB1387 “30บาทรักษาทุกที่ ในพื้นที่กรุงเทพฯ (ONE ID)” and KB1311 “นโยบาย 30บาทรักษาทุกที่ด้วยบัตรประชาชนใบเดียว(45 จังหวัด)” were retrieved, at rank 5 and 8 respectively.

Upon cross-inspection with between the call transcript and the call audio itself, we found that a number of keywords that could be used to identify relevant topics/policies and in effect the corresponding KB were misinterpreted in the transcript into completely different words. We suspect this could also be a bottleneck for the accuracy of KB retrieval in deployment.

However at the same time, some irrelevant KBs were also retrieved, such as KB1242 “การเบิก PP ผ่าน KTB และ App เป้าตัง” which was rank 1. The text in the top chunks retrieved were

```
"chunks": [
  {
    "rank": 1,
    "chunk_id": "KB1242#254",
    "score": 9.3488605,
    "text": "- วันศุกร์: 08:30 - 16:30"
  },
  {
    "rank": 2,
    "chunk_id": "KB1242#255",
    "score": 9.3488605,
    "text": "- วันเสาร์: 08:30 - 16:30"
  },
]
```

which had nothing to do with the conversation contents. We will diagnose this issue and see if this persists with later/better iterations of the retrieval pipeline.

```
{
  "kb_id": "KB1242",
  "title": "การเปิด PP ผ่าน KTB และ App เป้าตัง",
  "score": 9.3489,
},
{
  "kb_id": "KB1187",
  "title": "บริการแพทย์ทางไกล 42 กลุ่มโรคผ่าน 5 แอป ( Telemedicine )",
  "score": 8.913,
},
{
  "kb_id": "KB1172",
  "title": "ยกเลิกสัญญา รพ.เอกชน 9 แห่ง กทม.",
  "score": 8.913,
},
{
  "kb_id": "KB1148",
  "title": "บูรณาการสายด่วนรับแจ้งนำบัตรักษาสุขภาพติด",
  "score": 8.913,
},
{
  "kb_id": "KB1387",
  "title": "30นาทรักษากลุ่มโรคผ่าน 5 แอป ( ONE ID )",
  "score": 8.913
},
{
  "kb_id": "KB1305",
  "title": "บริการแพทย์ทางไกล 42 กลุ่มโรคผ่าน 5 แอป ( Telemedicine )",
  "score": 8.913,
},
{
  "kb_id": "KB1393",
  "title": "ดูห่วงใย",
  "score": 8.913,
},
{
  "kb_id": "KB1311",
  "title": "นโยบาย 30นาทรักษากลุ่มโรคผ่าน 5 แอป (45 จังหวัด)",
  "score": 8.913,
},
{
  "kb_id": "KB1314",
  "title": "ร่วมจ่ายหรือยกเว้นร่วมจ่าย ค่าบริการ 30นาท",
  "score": 8.9092,
}
}
```

#### Full JSON output of retrieval

#### 4.3.4 Performance Summary

To achieve a balance between performance and hardware cost, the system was deployed across 2 A100 GPUs, each responsible for different model groups:

GPU Allocation

GPU	Models Deployed	Model Sizes	GPU Usage Ratio
GPU 0	- Fine-tuned Whisper Moonson 746B - Pyannote-audio (Speaker Diarization)	746B / unspecified	80% Whisper / 20% Pyannote
GPU 1	- LLaMA 3.2 (1B) - Qwen2.5 (0.5B) - BGE-M3 (1.3B)	1B / 0.5B / 1.3B	80% LLMs / 20% Embeddings

Inference Latency (Sample: 8 minutes 37 seconds of audio)

Component	Latency (seconds)
Speaker Diarization	7.18
Speech-to-Text (Whisper)	9.27
Summarization	1.05 (LLaMA) / 1.46 (Qwen2.5)
KB Classification	1.72

This configuration represents an effective trade-off between throughput and hardware resource utilization, allowing real-time or near-real-time processing on multi-turn call recordings.

However, if cost reduction is prioritized by NHSO, a lower-resource alternative can be adopted:

- Replace LLaMA summarization with Qwen2.5 (0.5B)
- Replace BGE-M3 (1.3B) with SCT-Distil-Model-Phayathai-BERT-BGE-M3 (768B)

This reduces the combined memory footprint of GPU 1 by approximately 50%, potentially enabling inference on a single A100 GPU. However, this change will result in:

- Longer response times, especially during simultaneous requests
- Slightly reduced summarization and classification accuracy

## Chapter 5 Conclusions

## 5.1 Summary of Accomplishments

### 5.1.1 Speech To Text

#### 5.1.1.1 Evaluation and Selection of Speech-to-Text Models

We tested multiple Speech-to-Text models and determined that Monsoon Whisper, a fine-tuned version of Whisper, was the best choice due to its low word error rate and its availability at no cost.

#### 5.1.1.2 Implementation of Speaker Diarization with Pyannote

We implemented speaker diarization using PyAnnote's speaker segmentation pipeline to accurately identify and label distinct speakers within the audio. This process enabled us to segment the audio into speaker-specific segments, which were crucial for accurate transcription and analysis of multi-speaker conversations.

#### 5.1.1.3 Fine-tuning Monsoon Whisper

We fine-tuned the Monsoon Whisper model, a version of Whisper optimized for Thai speech, using the CMKL/Porjai-Thai-voice-dataset-central dataset. This fine-tuning led to a 4% improvement in transcription accuracy, significantly reducing the word error rate when tested on a small audio sample. This improvement demonstrated the effectiveness of fine-tuning in enhancing the model's precision and overall performance of the speech-to-text system.

#### 5.1.1.4 Fine-tuning Pyannote Speaker Diarization

To enhance the speaker diarization model's performance, we created approximately 10,000 synthetic conversation audio files, along with corresponding RTTM and UEM files, using data from THAI SER. These synthetic conversations were generated by combining audio clips from different actors across studio and Zoom recordings. This significantly improved the model's segmentation performance. However, we observed overfitting to the synthetic patterns, which led us to deploy the original PyAnnote model to ensure better generalization on unseen real-world data.

### 5.1.2. Data Synthesis

*Due to the absence of real NHSO call data, we synthesized over 57,000 realistic Thai-language call center transcripts using NHSO knowledge base documents. Across seven dataset versions, we progressively improved quality—from early large-model generation to a modular scenario-based pipeline using Gemma 3 and Gemini 2.0. This enabled us to support up to four KBs per conversation, increase variety, and produce more natural dialogues. We also integrated vision-language models for image-based KB extraction, enhancing multimodal coverage. The final dataset provides a strong foundation for downstream model training and evaluation.*

### 5.1.2 Topic Classification

In this project, we approached topic classification as a Knowledge Base (KB) retrieval problem, aiming to identify the most relevant KB entries for each call transcript with high recall and low latency. Unlike traditional single-label classifiers, our task required multi-label, open-domain classification—where each call could correspond to 1 to 5+ KBs, with highly variable length and content.

To solve this, we experimented with a wide range of retrieval strategies: BM25-only (sparse), dense-only (bi-encoder), and hybrid retrieval methods. We explored multiple Thai-language embedding models including WangchanBERT, PhayathaiBERT, SCT-Distil-BGE, and BGE-M3. Each model's limitations and advantages were carefully tested across chunking constraints, token coverage, and retrieval performance. We implemented several chunking strategies—including recursive,  $\Delta$ -cosine semantic cuts, and dynamic confidence-based splitting—eventually adopting a hybrid method for production to balance semantic coherence and runtime cost.

Our most effective pipeline combined dense, sparse, and ColBERT representations (tri-vector hybrid retrieval using BGE-M3), followed by cross-encoder reranking with LoRA fine-tuning. This approach achieved up to 91% Recall@10 and 0.85 MRR on our 57,000+ synthesized conversations. Importantly, we observed that chunk retrieval quantity and diversity ( $\geq 20\text{--}30$  KBs covered) significantly impacted recall, especially for long or multi-topic calls. We also addressed infrastructure constraints by deploying fallback models like WangchanBERT and SCT-Distil for CPU-only environments.

Overall, our KB classification framework achieves high semantic coverage and real-time feasibility, offering a scalable and accurate solution for post-call CRM field recommendation. The modular nature of our system also paves the way for future enhancements in retrieval, reranking, and real-world fine-tuning.

#### *5.1.3 Summarization*

We successfully developed and fine-tuned two summarization models—LLaMA 3.2-1B and Qwen2.5-0.5B—using a distillation-based approach on our 57,000-entry synthetic dataset. Despite the lack of real human-annotated summaries, we leveraged a large LLM (Gemma-3 27B IT) to generate high-quality pseudo-labels, enabling effective supervised training. Both models showed significant improvements in ROUGE scores after fine-tuning, with LLaMA achieving higher performance overall, while Qwen offered a lightweight alternative with competitive results. Our results demonstrate the feasibility of training robust Thai summarization models using clean synthetic data and model distillation, setting a strong foundation for future fine-tuning on real-world transcripts.

#### *5.1.4 Deployment*

We successfully designed and implemented a modular, scalable deployment pipeline for the entire system using Docker containers. All key services—file storage, transcription, speaker diarization, summarization, embedding, and REST API coordination—were containerized and integrated into a unified architecture. Despite lacking formal NHOI infrastructure specifications, we built a flexible demo system capable of handling realistic end-to-end workflows. Each inference model was deployed as an independent microservice, optimized for asynchronous operation and inter-container communication. The REST API layer was structured for clean modularity, exposing essential endpoints for audio uploads, summarization, and knowledge base management. This deployment framework lays a strong foundation for future adaptation to NHOI's production environment once their system requirements are known.

## **5.2 Issues and Obstacles**

### *5.2.1 Speech To Text*

We accomplish all of our goals but there were many obstacles along the way:

### **5.2.1.1 Difficulty in Transcribing Thai Audio**

An issue we encountered very early in our project was the models were having a difficult time transcribing Thai audio, they sometimes wrote the transcriptions in another language or missing words. This problem was very noticeable in audio samples with multiple speakers or background noise. We overcame this by researching about STT models especially made for Thai language like WhisperX and WhisperTH. These models provided much better accuracy for Thai speech.

### **5.2.1.2 Challenges with Overlapping Speech**

Another issue we encountered, which we have not yet fully resolved yet, was when two or more speakers talk over each other. The model sometimes doesn't transcribe all the words spoken or just transcribe one speaker that is speaking louder than the other. However, this issue improved significantly when we implemented speaker diarization, as it helped separate the speakers more clearly. We will continue improving this in the next semester.

### **5.2.1.3 Lack of Access to Relevant Datasets**

The last obstacle we encountered was the lack of access to NHSO's dataset or any suitable call recording datasets to fine-tune our Speech-to-Text model. To address this, we tried requesting access to FutureBeeAI's Call Center Speech Dataset for Healthcare, but we didn't receive a response. Without these datasets, we had to look for other available datasets online. However, it was still difficult to improve the model's performance and ensure it met the specific needs of our project.

## *5.2.2. Data Synthesis*

### **5.2.2.1 Lack of Real Transcript Examples**

One of the most significant obstacles was the absence of actual conversation transcripts from NHSO. The only data we received were brief summaries, not full dialogues. This made it extremely difficult to design realistic examples for our system prompts. Without authentic transcripts to reference, crafting conversation flows that mimic real call center interactions required heavy manual effort and assumptions.

### **5.2.2.2 Hardware Constraints for Long Context Generation**

On average, each knowledge base (KB) entry contains up to 20,000 tokens. Due to limited GPU memory, we were unable to include multiple KBs in a single input prompt to the large language models. This constraint limited our ability to synthesize multi-topic scenarios that would be more realistic. Additionally, longer input contexts significantly slowed inference, increasing the time required to generate each synthetic conversation.

### **5.2.2.3 Prompt Engineering Bottlenecks**

Producing high-quality, naturalistic conversations via prompting required extensive trial and error. Small adjustments to the system prompt often led to drastically different outputs, and in several cases, we had to regenerate entire datasets due to unsatisfactory results. Each iteration cycle could take over 8 hours, further delaying progress.

### **5.2.2.4 Delayed Start Due to Dependency on Real Data**

At the beginning of the first semester, we anticipated receiving real transcript data and therefore did not prioritize synthetic data generation. However, when it became clear in the second semester that no data would be provided, we had to pivot quickly to data synthesis. This late start, combined with the time required to experiment, design, and validate the

dataset, limited the opportunity for refinement. Since this synthetic dataset served as the foundation for multiple downstream components, it had to be completed under strict time constraints, which affected its overall quality.

### 5.2.3 TopicClassification

#### 5.2.2.1 Data Limitations

- **No Real-World Call Logs from NHO**

Due to a lack of access to actual contact center transcripts, all evaluations were conducted using synthesized data. While we generated >54,000 high-quality conversations with 1–4 ground-truth KBs, synthetic conversations may not fully capture the linguistic diversity, noise, and ambiguity present in real-world scenarios.

- **Chunk-Level Supervision Gap**

Our cross-encoder reranking model required [query, chunk] pairs for fine-tuning. However, the synthesized data provided only KB-level labels. We addressed this by filtering top chunks using Token-IoU and cosine similarity, but the supervision remained noisy, especially for longer KBs with diverse content.

- **Imperfect STT Transcripts**

Real-world testing revealed a major bottleneck in transcription quality. The Whisper model, while fast, introduced homophone errors, Thai transliteration inconsistencies, and segmentation issues — which in turn affected retrieval accuracy. This suggests an urgent need for domain-specific STT fine-tuning.

- **Lack of KB Complete KB Coverage in Transcript Training Data**

#### 5.2.2.2 Modelling and Retrieval Challenges

- **Semantic Drift vs Lexical Precision**

Dense models (e.g., BGE-M3) often captured paraphrases but missed exact lexical cues. Conversely, BM25 retrieved surface matches but lacked generalization. We resolved this through tri-fusion retrieval and reranking — though it increased system complexity and inference cost.

- **Long Document Chunking Trade-offs**

KB documents varied wildly in length (from <1k to >40k tokens). Chunking strategies like ClusterSemanticChunker gave excellent coherence but were computationally expensive. We had to adopt a hybrid chunker in production to balance recall and latency.

- **Overlapping KB Content**

Some KBs contained duplicated or closely related content (e.g., regional vs national policy variants). This made disambiguation harder for the reranker, and in some cases, led to ties or incorrect prioritization. More discriminative training data is needed here.

### 5.2.2.3 Infrastructure and Deployment Constraints

- **Lack of Docker/Sudo Support on Apex HPC**

Our deployment environment disallowed Docker and root access, limiting our ability to test production-ready setups (e.g., Triton with GPU-hosted rerankers or Milvus with GPU indexing). As a result, we had to re-implement parts of the pipeline using FAISS and standalone PyTorch models.

- **Resource Balancing on GPUs**

Simultaneous use of Whisper, PyAnnote, BGE-M3, and LLMs required careful GPU allocation (across  $2\times$  A100s). Without batching, inference latency would exceed our 2s target. The NHSO may lack GPUs altogether, necessitating fallback onto CPU

### 5.2.2.4 Evaluation Gaps:

- **Partial Ground Truth Coverage**

Even with Token-IoU and cosine filtering, it is possible that retrieved relevant chunks were unfairly penalized if they were not among the “top-3” labeled gold KBs. This likely underestimated our true recall and highlights the need for human-labeled multi-KB ground truth going forward.

- **Recall vs Stability Tradeoff**

Early retrieval configurations (e.g., fixed top-100 chunks) showed unstable results for long KBs. Increasing chunk quantity improved recall but risked noise accumulation. It required multiple rounds of ablation and hybrid evaluation logic to find the optimal balance.

## 5.2.4 Summarization

### 5.2.4.1 Training on Synthetic, Noise-Free Data

The summarization models were fine-tuned using the synthetic transcript dataset, which was extremely clean and free of noise. While this improved training stability, it created a gap between training and deployment conditions. In real-world usage, the input transcripts—produced by the speech-to-text system—often contain errors, informal phrasing, and transcription inconsistencies. Since the model was never exposed to these imperfections during training, its performance in practice is less accurate, often missing key details or generating summaries that do not reflect the full content.

### 5.2.4.2 Lack of Real Labeled Summaries from NHSO

Despite multiple follow-ups, we never received any real labeled summaries from NHSO. This forced us to generate summary labels using LLMs without ground-truth references, meaning the labels used for training may not fully represent the style, content, or purpose expected by NHSO. As a result, the model may have learned a style of summarization that does not align with real-world expectations, limiting its applicability without further fine-tuning on real data.

### *5.2.5. Deployment*

#### **5.2.5.1 Lack of Clarity on NHSO System Requirements**

We received no detailed documentation on NHSO’s existing infrastructure or deployment preferences. As a result, we had to design and implement multiple deployment configurations (e.g., GPU vs. non-GPU options, API interfaces, modular inference servers) without knowing which would be acceptable. This uncertainty significantly increased our workload and complexity during system design.

#### **5.2.5.2 Learning Curve for Containerized API Deployment**

None of the team members had prior experience deploying multi-container API systems using Docker, especially ones involving multiple GPU-based inference services. A considerable amount of time was spent learning best practices, debugging service communication, and managing container orchestration.

#### **5.2.5.3 Dependency on the APEX Supercomputer**

Due to limited local hardware (in particular, the lack of GPUs), most of the system had to be built and deployed on the APEX supercomputing infrastructure. This introduced additional friction, as configuration mismatches, environment inconsistencies, and occasional APEX system downtime led to unexpected bugs and delays that were difficult to debug locally.

#### **5.2.5.4 Codebase Integration Challenges**

Parts of the project—especially those developed in parallel by different team members—lacked consistent structure and modularity. This made integration across components more error-prone and time-consuming, requiring refactoring and additional coordination to unify disparate codebases.

#### **5.2.5.5 Delays in Other Sections Impacted Deployment**

Several dependencies were delivered later than expected, which pushed back the timeline for full system integration and testing. As deployment relies on stable outputs from earlier components, this cascading delay further compressed the implementation window.

### **5.3 Future Directions**

#### *5.3.1 Speech To Text*

We plan to improve our speaker diarization capabilities and refine the model’s performance in handling overlapping speech. Additionally, we aim to incorporate more healthcare and NHSO datasets to further fine-tune our models. Once the models are finalized, we will use them to transcribe NHSO recordings into transcripts, which can then be utilized by other departments for various applications. This will enhance the overall accessibility and usability of the data across the organization.

#### *5.3.2 Topic Classification*

While our current system meets most of the core objectives—particularly Recall@3  $\geq 95\%$  on synthesized conversations—several areas remain open for improvement, optimization, and real-world adaptation. Below, we outline actionable future directions to guide the next stages of development and deployment.

### 5.3.2.1. Real-World Data Collection and Evaluation

- **Ground-Truth Annotation from NHSO Calls**

To move beyond synthetic datasets, we aim to collaborate with NHO to collect real call recordings paired with verified KB tags. This will support high-fidelity evaluation, reduce domain mismatch, and provide critical insights into user language, discourse patterns, and real-world ambiguity.

- **User Feedback–Driven Evaluation**

In production, the system could solicit optional feedback from agents after each retrieved KB list. These implicit signals (clicks, selections, corrections) can inform future fine-tuning, model adaptation, and error analysis.

### 5.3.2.2. Reranker Expansion and Distillation

- **Cross-Encoder Label Refinement**

As human-labeled [transcript, chunk] pairs become available, future CE models can be trained with cleaner labels and larger positive/negative diversity. This may improve reranker generalization and coverage for borderline KBs.

- **Knowledge Distillation to CPU-Optimized Models**

While BGE-M3 cross-encoders deliver excellent accuracy, they are not always suitable for low-resource endpoints. Future work could distill their knowledge into smaller DistilBERT or TinyBERT variants, achieving comparable MRR at a fraction of the resource cost.

### 5.3.2.3. Multi-Modal and Visual KB Integration

- **Image-Aware KB Retrieval**

Some KBs (e.g., forms, diagrams, medical leaflets) contain crucial information in visual formats. Our planned MARVEL module will unify text and image encodings into a single retrieval framework. This enables more complete and equitable access to information, especially for visually dense documents.

- **OCR Fine-Tuning for Thai Health Documents**

We aim to train an OCR model tailored to the font, layout, and formatting of NHO-issued PDFs. This will increase extraction accuracy from form-heavy KBs and reduce post-processing burdens.

### 5.3.3 Summarization

To improve the accuracy and robustness of the summarization module, future work should focus on replacing synthetic training data with real NHO call recordings paired with human-written summaries from actual agents. This would allow the model to learn from real-world language patterns, common transcription imperfections, and practical summarization styles used in the field. Training on authentic data would significantly enhance the model’s ability to handle noisy inputs and produce summaries that are more accurate, context-aware, and aligned with NHO’s operational needs.

### 5.3.4 Deployment

The current deployment code was built under time constraints and still lacks consistency and adherence to standard coding practices. With more time, the codebase should be refactored for clarity, modularity, and maintainability, making it easier to scale and adapt in the future.

In addition, we have not yet conducted performance optimization for each component (e.g., parallel inference, memory management, request batching). Targeted improvements in these areas could significantly reduce response time and resource usage.

Finally, since we still do not have access to NHSO's actual system architecture or deployment requirements, the current system is structured as a demo-style implementation. Once NHSO provides more concrete information, the system can be adapted to their infrastructure, including integration with their data storage, authentication methods, or internal APIs.

## *5.4 Lessons Learned*

This project provided invaluable insights across multiple domains, from technical and engineering challenges to process management and team collaboration. The lessons we learned not only enhanced our technical expertise but also improved our ability to work effectively as a team under complex and dynamic conditions.

### *5.4.1 Speech-to-Text*

#### **5.4.1.1 The Importance of Thorough Testing**

Testing speech-to-text models with diverse audio inputs, such as noisy environments or multi-speaker dialogues, highlighted the importance of comprehensive evaluation. This thorough testing allowed us to identify strengths and weaknesses of models like Whisper and ensure they were suitable for real-world applications. By addressing potential issues early, we gained confidence in model selection and implementation.

#### **5.4.1.2 GPU Utilization**

Leveraging APEX's GPU capabilities enabled us to process large volumes of audio data efficiently. By optimizing model performance on GPUs, we reduced runtime significantly while maintaining transcription accuracy. This experience also strengthened our understanding of resource management for computationally intensive tasks.

### *5.4.2 Data Synthesis*

#### **5.4.2.1 Document Preprocessing:**

Learned to convert .docx files into clean, structured text by removing noise such as formatting artifacts, misordered image/table tags, and redundant punctuation. This preprocessing step was essential for ensuring consistency across synthesized dialogues.

#### **5.4.2.2 Vision-Language Inference with VLLM:**

Gained experience in using VLLM to deploy and serve vision-language models for image captioning. This included converting embedded images into meaningful Thai-language text descriptions by chaining VLLM output with downstream translation models.

#### **5.4.2.3 High-Throughput Synthesis with VLLM:**

Learned how to efficiently synthesize large datasets using VLLM with limited GPU resources by optimizing batching, controlling temperature for diversity, and managing memory under high-context-length scenarios.

#### **5.4.2.4 Leveraging Third-Party APIs (Gemini):**

Explored the practical use of third-party APIs such as Gemini 2.0 for vision-language inference. This included working within rate limits, managing token budgets, and evaluating output quality compared to open-source models.

#### **5.4.3 Classification**

##### **5.4.2.1 Real-World Data Mismatch is the Primary Bottleneck**

- While synthetic transcripts were helpful for pipeline prototyping and CE fine-tuning, the transition to real-world audio and transcription surfaced significant degradation—especially due to STT misinterpretations of Thai names, jargon, or overlapping speech.
- Manual inspection showed that many missed KB matches stemmed not from retrieval model failure, but from lossy or incorrect input representations due to transcription errors or unsegmented speaker turns. This reinforced the importance of robust preprocessing and potential STT fine-tuning.

##### **5.4.2.2 Recall is Not Enough Without Interpretability**

- Although we achieved >95% Recall@3 in controlled tests, real NHSO deployment demands explainability and auditability—agents need to know *why* a KB was suggested.
- This shaped our emphasis on logging, token-level attention maps, reranker score transparency, and candidate chunk previews in the API response. These will be essential for trust and future feedback-driven learning loops.

##### **5.4.2.3. Deployment Constraints Shape Everything**

- GPU availability, RAM ceilings, and even lack of Docker access in some environments forced us to build modular fallback versions of nearly every component:
  - CPU-only rerankers (TorchScript)
  - Smaller distilled embedding models
  - Flat index fallbacks for FAISS and Elastic
- The lesson: system flexibility matters just as much as model quality. Production-ready AI requires graceful degradation and multiple operational tiers.

##### **5.4.2.4. Iterative Experimentation Was Key to Progress**

- We rarely found the optimal configuration on the first attempt. Most of our final design—like chunk budgets, hybrid chunking, LoRA fine-tuning, reranking cutoffs—only emerged after multiple failed or suboptimal iterations.

- This process validated our choice to modularize components early (e.g., separating chunkers, embedding models, scorers, and rerankers), making it easier to swap parts, track performance, and scale experiments.

#### *5.4.4 Summarization*

##### **5.4.4.1 Fine-Tuning with LLaMAFactory**

Learned how to use LLaMAFactory to fine-tune LLaMA-based models on custom summarization tasks. This involved preparing instruction-style datasets, configuring training parameters, and evaluating output coherence on domain-specific texts.

##### **5.4.4.2 Model Distillation**

Gained experience in distilling smaller summarization models from larger teacher models. This process helped retain performance while reducing inference cost, making deployment feasible on limited hardware.

#### *5.4.5 Process and Team Dynamics*

##### **5.4.4.1 Time Management**

Underestimating the time required for preprocessing and model fine-tuning emphasized the importance of setting realistic timelines. Dividing the project into smaller milestones helped mitigate delays and maintain steady progress.

##### **5.4.4.2 Team Communication**

Clear documentation of experiments and results fostered transparency and collaboration. Regular check-ins and updates ensured alignment across team members, especially for technically complex tasks.

##### **5.4.4.3 Task Specialization**

Dividing responsibilities into preprocessing, model fine-tuning, clustering, and evaluation allowed each team member to focus on their strengths. This specialization enhanced efficiency and ensured high-quality outcomes.

##### **5.4.4.4 Adapting to Challenges**

Encountering unexpected challenges, such as missing data or technical limitations, taught us the importance of flexibility. By designing a robust system architecture and working on what was feasible, such as proof-of-concept models and system design, we maintained momentum and prepared for future phases.

#### *5.4.6 Deployment*

##### **5.4.6.1 Docker and Docker Compose for Productionisation:**

Learned to containerize individual services using Docker and orchestrate multi-service deployments with Docker Compose. This enabled modular, reproducible, and scalable infrastructure suitable for production environments, allowing efficient coordination between backend, inference servers, and databases.

#### **5.4.6.2 Inference with VLLM:**

Gained hands-on experience running large language models (e.g., Whisper, LLaMA) using VLLM, optimizing inference speed and memory efficiency for high-throughput applications.

#### **5.4.6.3 PostgreSQL Integration:**

Learned to set up and manage PostgreSQL as the backend database for storing transcripts, metadata, and processed outputs, supporting reliable querying and data persistence.

#### **5.4.6.4 FastAPI for Backend Development:**

Built RESTful APIs using FastAPI to serve frontend and inference components efficiently, leveraging asynchronous processing for better scalability.

#### **5.4.6.5 Structuring Inference APIs:**

Learned to design and implement clean API endpoints that encapsulate model inference logic, manage input/output pipelines, and coordinate between multiple model types (e.g., classification, summarization, STT) in a production-ready architecture.

### *Overall Lessons*

This project highlighted the need for rigorous testing, adaptability, and collaboration. We learned to navigate data limitations creatively, optimize resource use, and build scalable systems. The experience reinforced the value of meticulous planning and iterative refinement in achieving project objectives. These lessons will be instrumental in guiding future endeavors, particularly as we move toward integrating end-to-end functionality with the NHSO's CRM.

## **References**

### **Core Concepts and Architectures**

1. Transformer Model: Neural network architecture for sequence modeling. Documentation available at: <https://arxiv.org/abs/1706.03762> (Accessed: 5 May 2025).
2. RESTful APIs: Architectural style for designing networked applications. Documentation available at: <https://restfulapi.net/> (Accessed: 5 May 2025).
3. Relational Database: Database model based on relations (tables). Documentation available at: <https://www.ibm.com/topics/relational-databases> (Accessed: 5 May 2025).
4. Knowledge Base Retrieval: Techniques for extracting information from structured knowledge repositories. Documentation available at: [https://en.wikipedia.org/wiki/Knowledge\\_base](https://en.wikipedia.org/wiki/Knowledge_base) (Accessed: 5 May 2025).
5. Retrieval-Augmented Generation (RAG): Combining retrieval and generation in NLP. Documentation available at: <https://ai.facebook.com/blog/retrieval-augmented-generation/> (Accessed: 5 May 2025).
6. LoRA (Low-Rank Adaptation): Efficient fine-tuning method for large language models. Documentation available at: <https://arxiv.org/abs/2106.09685> (Accessed: 5 May 2025).
7. Supervised Learning: Machine learning paradigm using labeled data. Documentation available at: [https://scikit-learn.org/stable/supervised\\_learning.html](https://scikit-learn.org/stable/supervised_learning.html) (Accessed: 5 May 2025).
8. Multimodal Models: Models processing multiple data types (e.g., text, images, audio). Documentation available at: <https://arxiv.org/abs/2302.05891> (Accessed: 5 May 2025).
9. Vision-Language Models (VLMs): Models integrating visual and textual information. Documentation available at: <https://arxiv.org/abs/2102.03334> (Accessed: 5 May 2025).
10. Agentic LLMs: Language models with autonomous agent capabilities. Documentation available at: <https://arxiv.org/abs/2308.08155> (Accessed: 5 May 2025).

11. Asynchronous Processing: Non-blocking execution model for concurrent tasks. Documentation available at: <https://docs.python.org/3/library/asyncio.html> (Accessed: 5 May 2025).
12. Bi-Encoder Architecture: Dual encoder structure for efficient retrieval. Documentation available at: <https://arxiv.org/abs/1908.10084> (Accessed: 5 May 2025).
13. Cross-Encoder Architecture: Single encoder jointly processing input pairs for ranking. Documentation available at: <https://arxiv.org/abs/2005.14165> (Accessed: 5 May 2025).
14. Contrastive and Triplet Loss: Loss functions for representation learning. Documentation available at: <https://arxiv.org/abs/1412.6572> (Accessed: 5 May 2025).
15. Hard Negative Mining: Training strategy using challenging negative samples. Documentation available at: <https://arxiv.org/abs/1704.00646> (Accessed: 5 May 2025).

## Evaluation Metrics

16. Loss Function: Mathematical function for model optimization. Documentation available at: <https://developers.google.com/machine-learning/crash-course/descending-into-ml/video-lecture> (Accessed: 5 May 2025).
17. Precision: Metric for classification accuracy. Documentation available at: [https://en.wikipedia.org/wiki/Precision\\_and\\_recall](https://en.wikipedia.org/wiki/Precision_and_recall) (Accessed: 5 May 2025).
18. Recall: Metric for classification completeness. Documentation available at: [https://en.wikipedia.org/wiki/Precision\\_and\\_recall](https://en.wikipedia.org/wiki/Precision_and_recall) (Accessed: 5 May 2025).
19. F1 Score: Harmonic mean of precision and recall. Documentation available at: [https://en.wikipedia.org/wiki/F1\\_score](https://en.wikipedia.org/wiki/F1_score) (Accessed: 5 May 2025).
20. Accuracy: Overall correctness of predictions. Documentation available at: [https://en.wikipedia.org/wiki/Accuracy\\_and\\_precision](https://en.wikipedia.org/wiki/Accuracy_and_precision) (Accessed: 5 May 2025).
21. Diarization Error Rate (DER): Metric for speaker diarization accuracy. Documentation available at: <https://pyannote.github.io/pyannote-audio/reference/metrics/diarization/> (Accessed: 5 May 2025).
22. Mean Reciprocal Rank (MRR): Evaluation metric for ranked retrieval results. Documentation available at: [https://en.wikipedia.org/wiki/Mean\\_reciprocal\\_rank](https://en.wikipedia.org/wiki/Mean_reciprocal_rank) (Accessed: 5 May 2025).
23. Token Intersection-over-Union (Token-IoU): Overlap metric for token-level evaluation. Documentation available at: <https://aclanthology.org/2022.naacl-main.290/> (Accessed: 5 May 2025).
24. Latency (P95): 95th percentile latency measurement. Documentation available at: <https://cloud.google.com/blog/products/operations/understanding-distributions-measuring-latency> (Accessed: 5 May 2025).

## Additional Technologies

25. DeepCut: Thai word segmentation tool. Documentation available at: <https://github.com/rkcosmos/deepcut> (Accessed: 5 May 2025).
26. Node.js: JavaScript runtime for server-side applications. Documentation available at: <https://nodejs.org/en/docs/> (Accessed: 5 May 2025).
27. Express.js: Web framework for Node.js. Documentation available at: <https://expressjs.com/> (Accessed: 5 May 2025).
28. BGE-M3: Multilingual embedding model for retrieval. Documentation available at: <https://huggingface.co/BAAI/bge-m3> (Accessed: 5 May 2025).
29. VLLM: High-throughput inference engine for large language models. Documentation available at: <https://vllm.ai/> (Accessed: 5 May 2025).
30. FlagEmbedding: Library for efficient embedding generation. Documentation available at: <https://github.com/FlagOpen/FlagEmbedding> (Accessed: 5 May 2025).
31. OpenAI API Compatibility: Standardized API interface for LLMs. Documentation available at: <https://platform.openai.com/docs/api-reference> (Accessed: 5 May 2025).
32. Triton gRPC Inference: gRPC protocol for Nvidia Triton model serving. Documentation available at: <https://github.com/triton-inference-server/server/blob/main/docs/protocol/README.md> (Accessed: 5 May 2025).

## Speech-to-Text (STT) Models

33. Whisper: OpenAI's automatic speech recognition system. Documentation available at: <https://platform.openai.com/docs/guides/whisper> (Accessed: 5 May 2025).
34. pyannote-audio: Toolkit for speaker diarization and audio processing. Documentation available at: <https://github.com/pyannote/pyannote-audio> (Accessed: 5 May 2025).
35. pyannote-audio (Speaker Diarization) Bredin, H. et al. (2020) "pyannote.audio: Neural Building Blocks for Speaker Diarization." Paper available at: <https://arxiv.org/abs/1911.02232> (Accessed: 5 May 2025).
- 36.

## Natural Language Processing Models

37. BERT: Bidirectional Encoder Representations from Transformers for language understanding. Documentation available at: <https://github.com/google-research/bert> (Accessed: 5 May 2025).
38. ThaiBERT: Pretrained BERT models for Thai language. Documentation available at: <https://github.com/vistec-AI/thai2fit> (Accessed: 5 May 2025).
39. WangchanBERTa: Large-scale Thai language model based on RoBERTa. Documentation available at: <https://github.com/vistec-AI/wangchanberta> (Accessed: 5 May 2025).
40. Llama 3: Meta's large language model. Documentation available at: <https://ai.meta.com/resources/models-and-libraries/llama-downloads/> (Accessed: 5 May 2025).
41. Qwen2.5: Large language model developed by Alibaba Cloud. Documentation available at: <https://github.com/OwenLM/Qwen2> (Accessed: 5 May 2025).
42. Gemini 2.0: Google's multimodal generative AI model. Documentation available at: <https://deepmind.google/technologies/gemini/> (Accessed: 5 May 2025).
43. Gemma 3: Google's open-source lightweight language model. Documentation available at: <https://ai.google.dev/gemma> (Accessed: 5 May 2025).

## Retrieval and Ranking

44. BM25: Probabilistic information retrieval model. Documentation available at: [https://github.com/dorianbrown/rank\\_bm25](https://github.com/dorianbrown/rank_bm25) (Accessed: 5 May 2025).
45. FAISS: Facebook AI Similarity Search for efficient vector similarity search. Documentation available at: <https://faiss.ai/> (Accessed: 5 May 2025).
46. ColBERT: Contextualized Late Interaction over BERT for scalable retrieval. Documentation available at: <https://github.com/stanford-futuredata/ColBERT> (Accessed: 5 May 2025).
47. ColBERT Multi-Vector Retrieval: Extension of ColBERT for multi-vector search. Documentation available at: <https://github.com/stanford-futuredata/ColBERT> (Accessed: 5 May 2025).
48. HNSW Index: Hierarchical Navigable Small World graphs for approximate nearest neighbor search. Documentation available at: <https://github.com/nmslib/hnswlib> (Accessed: 5 May 2025).

## Machine Learning Frameworks and Tools

49. PyTorch: Open source machine learning framework. Documentation available at: <https://pytorch.org/> (Accessed: 5 May 2025).

50. Hugging Face Transformers: Library for state-of-the-art NLP models. Documentation available at: <https://huggingface.co/docs/transformers> (Accessed: 5 May 2025).
51. Accelerate: Hugging Face library for distributed training. Documentation available at: <https://huggingface.co/docs/accelerate> (Accessed: 5 May 2025).
52. LLaMA Factory: Toolkit for fine-tuning Llama models. Documentation available at: <https://github.com/hiyouga/LLaMA-Factory> (Accessed: 5 May 2025).

## Infrastructure and Deployment

53. Nvidia Triton Inference Server: Model serving platform for deploying AI models. Documentation available at: <https://github.com/triton-inference-server/server> (Accessed: 5 May 2025).
54. Kubernetes: Container orchestration platform. Documentation available at: <https://kubernetes.io/docs/home/> (Accessed: 5 May 2025).
55. Docker: Platform for developing, shipping, and running applications in containers. Documentation available at: <https://docs.docker.com/> (Accessed: 5 May 2025).
56. Docker Compose: Tool for defining and running multi-container Docker applications. Documentation available at: <https://docs.docker.com/compose/> (Accessed: 5 May 2025).
57. FastAPI: Web framework for building APIs with Python. Documentation available at: <https://fastapi.tiangolo.com/> (Accessed: 5 May 2025).

## Databases and Search

58. MySQL: Open-source relational database management system. Documentation available at: <https://dev.mysql.com/doc/> (Accessed: 5 May 2025).
59. PostgreSQL: Open-source object-relational database system. Documentation available at: <https://www.postgresql.org/docs/> (Accessed: 5 May 2025).
60. Elasticsearch: Distributed, RESTful search and analytics engine. Documentation available at: <https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html> (Accessed: 5 May 2025).

## Evaluation Metrics and Utilities

61. JiWER: Python package for measuring word error rate in speech recognition. Documentation available at: <https://github.com/jitsi/jiwer> (Accessed: 5 May 2025).
62. Rouge Score: Metric for automatic evaluation of summaries. Documentation available at: <https://pypi.org/project/rouge-score/> (Accessed: 5 May 2025).

## Text Processing

63. SentencePiece: Unsupervised text tokenizer and detokenizer. Documentation available at: <https://github.com/google/sentencepiece> (Accessed: 5 May 2025).
64. ICU Thai Analyzer: Thai language text segmentation tool. Documentation available at: <https://icu.unicode.org/> (Accessed: 5 May 2025).

