# **Evaluation and Improvement Plan for Offline Bank Call Transcription and Translation Model**

**June 2025**

## **Abstract**

This document outlines an internal plan to evaluate and enhance our offline transcription, diarization, and translation models for bank call audios, covering temporary pretrained models (Whisper, SeamlessM4Tv2, AgglomerativeClustering) and proprietary models under development. The models process client-provided audio calls to produce transcriptions, speaker labels, and English translations, operating offline on CPU. The plan focuses on internal evaluation using self-generated ground truth, generating performance reports, and improving models through iterative fine-tuning and deployment. Leveraging client audio calls and internal annotation, the approach ensures domain-specific accuracy for banking inquiries (e.g., loans, balances) while maintaining offline compatibility. Official sources, including Hugging Face documentation, guide the development process.

## **1. Introduction**

Our models process bank call audio to deliver three outputs:

* **Transcription**: Raw language text with timestamps, capturing banking terminology (e.g., “utang” as “loan”).
* **Diarization**: Speaker labels identifying the caller (spk1) and bank agent (spk2), using channel-based or neural separation for stereo/mono audio (temporary models use AgglomerativeClustering).
* **Translation**: English text from raw language transcription, preserving banking context.

The temporary models (Whisper large-v3, SeamlessM4Tv2) serve as an interim solution, while proprietary models are developed for long-term customization. Both operate offline on CPU, supporting intent prediction for banking inquiries. This internal plan addresses:

* Setting up temporary models and designing proprietary models.
* Building data pipelines for client audio calls.
* Evaluating performance using internally generated ground truth.
* Generating downloadable performance reports.
* Iteratively fine-tuning and deploying models via a model improvement cycle.

Clients provide audio calls, while all ground truth (transcriptions, diarization, translations) and evaluations are handled internally by our data annotation team.

## **2. Internal Tasks**

Our team handles the following tasks to support evaluation and improvement:

### **2.1. Model Setup and Design**

* **Temporary Models**: Configure pretrained Whisper, SeamlessM4Tv2, and AgglomerativeClustering for offline operation.
* **Proprietary Models**: Define architectures for transcription (transformer-based ASR), diarization (agglomerative clustering), and translation (neural machine translation). Prototype on pilot data and benchmark against temporary models.

### **2.2. Data Pipeline Setup**

* Develop preprocessing pipeline for audio (e.g., normalization, noise reduction) and text (e.g., orthography standardization, code-switching handling).
* Automate data loading, splitting (training/validation/test), and annotation conversion.
* Implement quality assurance (QA) checks for annotations (e.g., timestamp alignment, translation accuracy).

### **2.3. Ground Truth Generation**

* Label client audio calls for transcriptions, diarization, and translations using semi-automated tools (e.g., Whisper for initial transcription, manual correction).
* Reserve validation and test sets (e.g., 10%/10% of calls) separate from training data.
* Ensure ground truth includes banking terms (e.g., “balanse,” “utang”) and handles code-switching.

### **2.4. Continuous Integration and Testing**

* Set up a continuous integration (CI) pipeline with automated tests for model components and data pipeline.
* Conduct regression testing during fine-tuning to detect performance drops.

### **2.5. Evaluation Pipeline**

* Build a Python-based pipeline to compare model outputs to ground truth, computing metrics like Word Error Rate (WER) and BLEU using offline-compatible libraries (e.g., jiwer, sacrebleu, nltk).

### **2.6. Performance Reporting**

* Generate downloadable reports (PDF, CSV) summarizing metrics, error analysis, and improvement recommendations.

### **2.7. Model Improvement Cycle**

* Fine-tune models iteratively, using validation sets for hyperparameter tuning and regularization to prevent overfitting.
* Optimize models for CPU deployment (e.g., quantization, pruning).
* Deploy improved models with updated code and documentation.

## **3. Evaluation Metrics**

The following metrics assess performance:

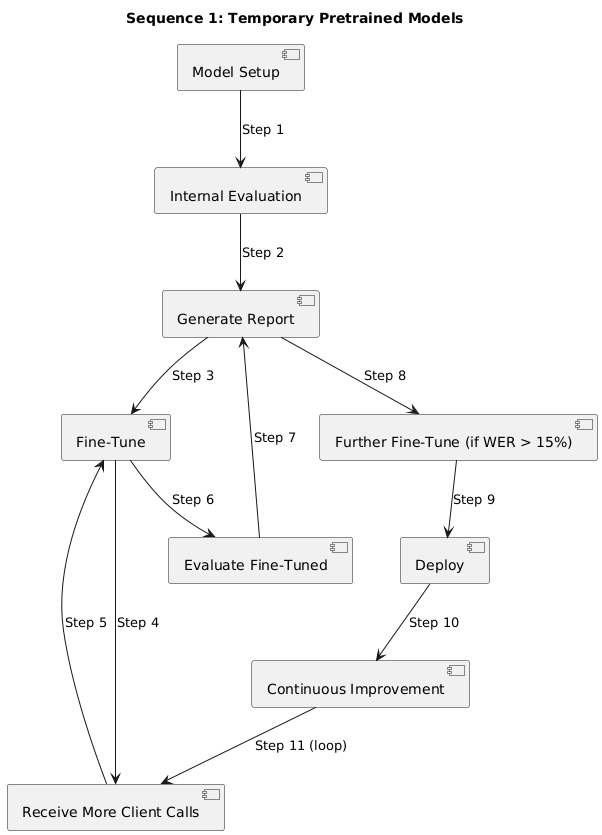
* **Transcription**:
  + **Word Error Rate (WER)**: Percentage of incorrect words (target < 20%) [1].
  + **Character Error Rate (CER)**: Character-level errors for Tagalog orthography.
  + **Banking Term Accuracy**: Accuracy for key terms (e.g., “balanse,” “loan”).
* **Diarization**:
  + **Channel Accuracy (Stereo)**: Correct channel assignments (target > 95%).
  + **Diarization Error Rate (DER) (Mono)**: Errors in speaker assignment and timing (target < 15%) [2].
* **Translation**:
  + **BLEU Score**: N-gram overlap with reference translations (target > 0.30) [3].
  + **METEOR Score**: Semantic accuracy for banking terms (target > 0.30) [4].
  + **chrF**: Character-based metric for short phrases.
* **Latency**: Processing times for transcription, diarization, and translation.

## **4. Evaluation and Improvement Plans**

Two sequences are outlined: Sequence 1 for temporary pretrained models and Sequence 2 for proprietary models. Both incorporate a model improvement cycle, highlighted in their respective flow diagrams.

### **4.1. Sequence 1: Temporary Pretrained Models**

#### **Flow Diagram**



#### **Step 1: Model Setup**

* Configure Whisper large-v3, SeamlessM4Tv2, and AgglomerativeClustering for offline operation (e.g., TRANSFORMERS\_OFFLINE=1).
* Integrate pyannote.audio for mono diarization if needed.

#### **Step 2: Internal Evaluation**

* Evaluate on internal dataset (e.g., 5 calls, 25 minutes) with self-generated ground truth.
* Compute metrics (WER, CER, BLEU, DER) and analyze errors (e.g., “balanse” misrecognition).

#### **Step 3: Generate Performance Report**

* Summarize metrics, errors, and fine-tuning recommendations.
* Note internal ground truth and temporary model usage.
* Formats: PDF (via reportlab), CSV.

#### **Step 4: Fine-Tune Model**

* Fine-tune Whisper/SeamlessM4Tv2 using internal dataset, tuning hyperparameters (e.g., learning rate) on validation set (e.g., 5 calls).
* Optimize AgglomerativeClustering parameters if applicable.
* Apply regularization (e.g., dropout) and quantization (e.g., 8-bit).

#### **Step 5: Receive Additional Client Calls**

* Obtain a large number of calls.

#### **Step 6: Fine-Tune Model (Model Improvement Cycle)**

* Label new calls for training, reserving validation/test sets (e.g., 10%/10%).
* Fine-tune Whisper/SeamlessM4Tv2 on training data, using validation set.
* Conduct regression testing (e.g., compare WER to Step 4).

#### **Step 7: Evaluate Fine-Tuned Model (Model Improvement Cycle)**

* Evaluate on a fixed test set (5 calls) with self-generated ground truth.
* Compute updated metrics and analyze errors.

#### **Step 8: Generate Performance Report (Model Improvement Cycle)**

* Summarize improvements (e.g., WER: 25% → 15%) and errors.
* Recommend further fine-tuning if WER > 15%.
* Formats: PDF, CSV.

#### **Step 9: Further Fine-Tune (Conditional, Model Improvement Cycle)**

* Fine-tune if WER > 15%, using validation set and regularization.
* Optimize model for CPU (e.g., pruning, ONNX).

#### **Step 10: Deploy Model**

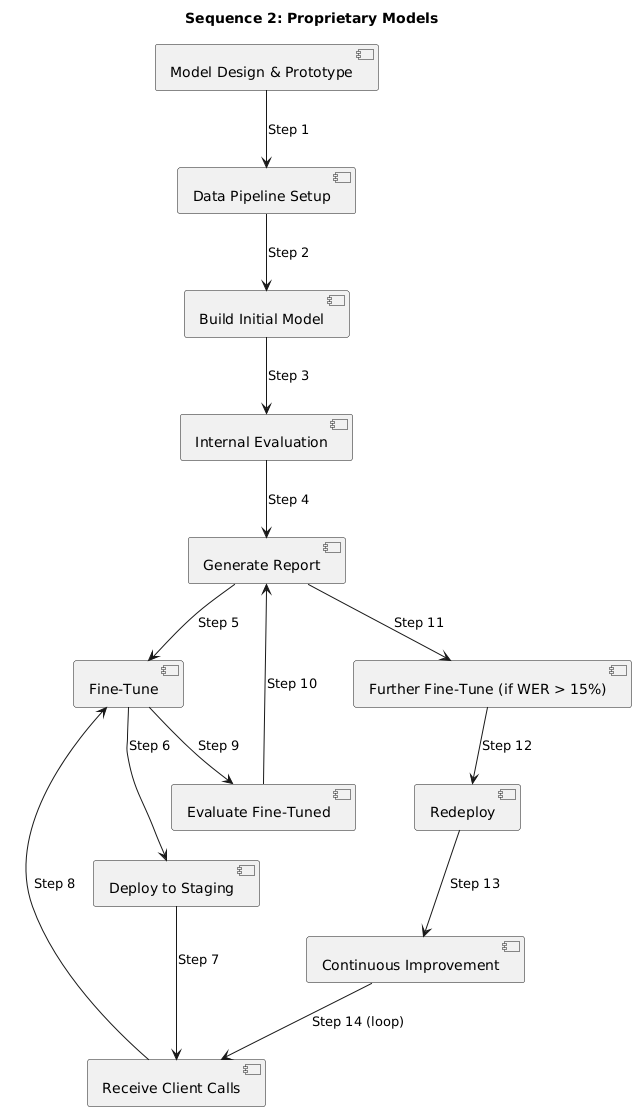
* Deploy to production with offline installation instructions.
* Document fine-tuning and evaluation results.

#### **Step 11: Continuous Improvement**

* Request more calls if WER > 15%.
* Re-run model improvement cycle (fine-tuning, evaluation, reporting).
* Transition to proprietary models as they mature.

### **4.2. Sequence 2: Proprietary Models**

#### **Flow Diagram**



#### **Step 1: Model Design and Prototyping**

* Define architectures for transcription (transformer-based ASR), diarization (neural speaker embedding), and translation (neural machine translation).
* Prototype on pilot dataset (e.g., 5 calls, 25 minutes) with annotations.
* Benchmark against Whisper/SeamlessM4Tv2 (e.g., WER < 25%).
* Document architecture and results.

#### **Step 2: Data Pipeline Setup**

* Develop preprocessing scripts (e.g., 16kHz normalization with librosa, text lowercase).
* Automate data splits (training/validation/test, e.g., 80%/10%/10%) and JSON conversion.
* Implement QA checks (e.g., flag timestamp overlaps).

#### **Step 3: Build Initial Model**

* Train on client calls (e.g., 50 calls, 250 minutes), using labeled training data.
* Reserve validation (5 calls) and test sets (5 calls) during annotation.
* Tune hyperparameters (e.g., learning rate) on validation set, monitoring WER.
* Apply domain adaptation (e.g., add 500 banking terms, weighted loss).

#### **Step 4: Internal Evaluation**

* Evaluate on fixed test set (5 calls) using self-generated ground truth.
* Compute metrics (WER, CER, BLEU, DER) and analyze errors.

#### **Step 5: Generate Performance Report**

* Summarize metrics, per-file errors, and fine-tuning recommendations (e.g., improve WER > 20%).
* Note internal ground truth usage.
* Formats: PDF, CSV.

#### **Step 6: Fine-Tune Model**

* Fine-tune based on report (e.g., target high WER).
* Use validation set for hyperparameter tuning and early stopping.
* Apply regularization and quantization.

#### **Step 7: Deploy to Staging**

* Deploy to internal staging environment.
* Validate on new calls (e.g., 5 calls).
* Collect informal client feedback (e.g., translation clarity).

#### **Step 8: Receive Additional Client Calls**

* Obtain large number of calls (e.g., 50–100, 250–500 minutes).

#### **Step 9: Fine-Tune Model (Model Improvement Cycle)**

* Label new calls for training, reserving validation/test sets.
* Fine-tune on training data, tuning hyperparameters on validation set.
* Conduct regression testing (e.g., compare WER to Step 6).

#### **Step 10: Evaluate Fine-Tuned Model (Model Improvement Cycle)**

* Evaluate on same fixed test set (5 calls).
* Compute updated metrics and analyze errors.

#### **Step 11: Generate Performance Report (Model Improvement Cycle)**

* Summarize improvements and errors.
* Recommend further fine-tuning if WER > 15%.
* Formats: PDF, CSV.

#### **Step 12: Further Fine-Tune (Conditional, Model Improvement Cycle)**

* Fine-tune if WER > 15%, using validation set and regularization.
* Optimize model (e.g., pruning, ONNX).

#### **Step 13: Redeploy Model**

* Deploy to production with offline instructions.
* Document architecture, hyperparameters, and results.

#### **Step 14: Continuous Improvement**

* Request more calls if WER > 15%.
* Re-run model improvement cycle.
* Conduct knowledge transfer workshops.

## **6. Conclusion**

This internal plan ensures robust evaluation and enhancement of temporary and proprietary models for bank call transcription and translation. Sequence 1 leverages pretrained models for immediate results, while Sequence 2 drives long-term customization with proprietary models. By using client audio calls, internal ground truth, and iterative model improvement cycles, we achieve high accuracy for banking inquiries while maintaining offline compatibility. Immediate next steps include setting up data pipelines and annotating new client calls to initiate both sequences.

## **References**

[1] V. Panayotov et al., “LibriSpeech: An ASR Corpus Based on Public Domain Audio Books,” INTERSPEECH, 2015.  
[2] H. Bredin, “pyannote.metrics: A Toolkit for Reproducible Evaluation, Diagnostics, and Error Analysis of Speaker Diarization Systems,” 2017.  
[3] K. Papineni et al., “BLEU: A Method for Automatic Evaluation of Machine Translation,” ACL, 2002.  
[4] S. Banerjee and A. Lavie, “METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments,” ACL Workshop, 2005.  
[5] Hugging Face, “Whisper Model Documentation,”<https://huggingface.co/docs/transformers/model_doc/whisper>, 2023.  
[6] Hugging Face, “SeamlessM4T Model Documentation,”<https://huggingface.co/docs/transformers/model_doc/seamless_m4t>, 2023.