Multilingual Pronunciation Assessment and Training Application

Project Report

Author: 黃毅成
Student ID: 2252634
Date: December 24, 2025

1 Introduction

In the domain of Second Language Acquisition (SLA), achieving native-like pronunciation is often one of the most significant challenges for learners. While vocabulary and grammar can be studied through textbooks, pronunciation requires physical coordination of the articulators and, crucially, immediate and accurate auditory feedback.

The core problem this project addresses is the scarcity of **accessible, granular, and multimodal feedback** for language learners practicing oral speech.

1.1 Limitations of Current Solutions

Traditional language learning methods rely heavily on human tutors. However, one-on-one coaching is expensive and not scalable. Conversely, many existing mobile applications (like Duolingo or standard voice assistants) use "black box" Automatic Speech Recognition (ASR). These systems are designed to be lenient—optimizing for understanding the meaning of what was said—rather than critiquing the accuracy of how it was said. They typically provide a binary "correct/incorrect" result without explaining which specific phoneme caused the error.

Learners often face a "feedback gap" where they know they sound "off" but cannot identify the specific articulatory error. For example, a learner might fail to distinguish between the vowels /i/ and /ɪ/, or miss the tonal nuances in Mandarin or Japanese.

1.2 Proposed Solution

This project aims to solve these issues by developing a desktop application that provides a hybrid feedback mechanism:

1. **Deterministic Precision**: Using a local C++ inference engine to calculate phoneme-level Goodness of Pronunciation (GOP) scores, allowing learners to pinpoint exact segmental errors (e.g., "You pronounced /r/ effectively, but the /ae/ was distorted").
2. **Holistic Coaching**: Leveraging Multimodal Large Language Models (LLMs) to analyze suprasegmental features (intonation, stress, rhythm) and provide natural language advice, simulating a human coach.

By supporting multiple languages (English, Chinese, French, German, Spanish, Russian), the system aims to democratize access to high-quality pronunciation training.

2 Market Survey of Existing Solutions

The field of Computer-Assisted Pronunciation Training (CAPT) has rapidly expanded with the rise of mobile computing and Artificial Intelligence. Currently, the market for language learning applications is saturated, yet few products offer a comprehensive solution for granular pronunciation correction across multiple languages.

- **ELSA Speak**:
  - *Overview*: Perhaps the most prominent competitor in the pronunciation space, focusing specifically on reducing accents.

- *Strengths*: It provides excellent phoneme-level feedback and articulatory visual aids.
- *Limitations*: It is primarily focused on English learners. Users learning other major languages (like Chinese, Russian, or German) cannot benefit from its advanced scoring engine. Additionally, its "coaching" is often template-based rather than dynamically generated.

- **Duolingo**:
  - *Overview*: A widely used language learning app that includes speaking exercises.
  - *Strengths*: Gamified learning experience and broad language support.
  - *Limitations*: The pronunciation feedback is binary and lacks detailed phonetic analysis. It does not provide specific guidance on how to improve pronunciation.

Gap Analysis: Most current solutions force a trade-off: they are either highly precise but monolingual (ELSA), or multi-lingual but lacking in rigorous feedback (Duolingo). Furthermore, very few utilize real-time Multimodal LLMs to provide "human-like" qualitative advice. This project aims to bridge this gap by combining multi-language support (English, Chinese, French, German, Spanish, Russian) with both precise scoring and AI coaching.

3 Methods

3.1 Wav2Vec 2.0

To achieve robust phoneme recognition across different languages, this project utilizes Wav2Vec 2.0[1], a state-of-the-art framework for self-supervised learning of speech representations.

- **Architecture**: The model consists of a multi-layer convolutional feature encoder that takes raw audio waveforms as input and outputs latent speech representations. These are fed into a Transformer network to build context representations.
- **Implementation**: In this application, the model is deployed via ONNX Runtime (C++) for high-performance inference. The system loads a quantized .onnx model and processes raw PCM audio data (Float32, 16kHz, Mono) directly in memory.
- **Output**: The model outputs a probability distribution (logits) over the vocabulary for each time step (frame), which serves as the foundation for the subsequent alignment algorithms.

3.2 Viterbi Algorithm

The primary function of the **Viterbi algorithm**[2] is to perform **Forced Alignment**. It synchronizes the variable-length sequence of acoustic frames (generated by the Wav2Vec 2.0 model) with the fixed sequence of target phonemes. This alignment is the critical prerequisite for identifying the exact start and end boundaries of each phoneme, which allows the system to isolate specific audio segments for Goodness of Pronunciation (GOP) scoring.

The algorithm uses dynamic programming to find the most likely sequence of states (the optimal path) by recursively maximizing the cumulative probability at each time step. The core recurrence relation is defined as:

$$V_{t,j} = \max_i \left[V_{t-1,i} \cdot a_{i,j} \right] \cdot P(o_t \mid j)$$

Where:

- $V_{t,j}$: The probability of the most likely path ending in state (phoneme) j at time frame t.
- $V_{t-1,i}$: The probability of the optimal path at the previous time step ending in state i.
- $a_{i,j}$: The transition probability of moving from phoneme i to phoneme j.
- $P(o_t \mid j)$: The emission probability, derived from the acoustic model (Wav2Vec 2.0), representing the likelihood that the audio frame o_t corresponds to phoneme j.

The execution of the algorithm proceeds in three distinct phases:

1. **Initialization**: The algorithm computes the starting probabilities for the first phoneme states at the first time frame ($t = 0$).
2. **Forward Recursion**: For each subsequent time frame, the algorithm calculates $V_{t,j}$ for all possible states. It stores a "backpointer" to the previous state i that maximized the probability, effectively building a trellis of possible paths.
3. **Backtracking**: Once the final time frame is reached, the algorithm selects the state with the highest probability and traces the backpointers in reverse order (from end to start). This reconstructs the optimal alignment path, yielding the precise time boundaries for every phoneme in the input text.

3.3 Goodness of Pronunciation (GOP) Scoring

Once the temporal boundaries of each phoneme are established via forced alignment, the system evaluates the acoustic quality using the **Goodness of Pronunciation (GOP) metric**[3]. GOP is defined as the duration-normalized log-posterior probability of the target phoneme given the acoustic observations. It provides a confidence score indicating how closely the user's realization of a phoneme matches the acoustic model's training data.The core formula for calculating the GOP score for a phoneme p is defined as:

$$\text{GOP}(p) = \frac{1}{d} \sum_{t=t_{\text{start}}}^{t_{\text{end}}} \log P(p \mid o_t)$$

Where:

- d: The duration of the phoneme in frames ($d = t_{\text{end}} - t_{\text{start}} + 1$).
- $t_{\text{start}}, t_{\text{end}}$: The start and end time frames identified by the Viterbi alignment.o_t: The acoustic feature vector at time frame t.
- $P(p \mid o_t)$: The posterior probability that the current frame belongs to phoneme p, as output by the acoustic model (Wav2Vec 2.0).

In this implementation, the resulting GOP score is a negative log-probability value. A score closer to 0 indicates a higher probability and thus "better" pronunciation. Empirical thresholds are applied to these scores to categorize pronunciation quality; for instance, scores above -1.0

are classified as “Excellent,” while scores above −2.5 are deemed “Good”. This metric serves as the quantitative basis for the feedback provided to the user.

3.4 Multimodal Large Language Models: GPT-4o Realtime

To overcome the limitations of traditional numerical scoring, this project leverages *GPT-4o-realtime-preview*, a cutting-edge multimodal model developed by OpenAI. Unlike standard Large Language Models (LLMs) that process only text, GPT-4o is natively multimodal, capable of processing audio, vision, and text inputs with ultra-low latency.

- **End-to-End Audio Processing**: Traditional voice assistants rely on a “cascade” pipeline: Automatic Speech Recognition (ASR) converts speech to text, the LLM processes the text, and Text-to-Speech (TTS) generates audio. This process often strips away paralinguistic features like tone, pitch, and emotion. In contrast, GPT-4o Realtime processes raw audio tokens directly. This allows the model to perceive suprasegmental nuances—such as hesitation, stress, and intonation patterns—that are critical for language coaching.
- **Real-Time Interaction**: The model is optimized for synchronous communication via persistent WebSocket connections, enabling immediate feedback loops. In the context of CAPT, this allows the system to function not just as a scorer, but as an interactive “virtual coach” that provides qualitative, human-like advice on pronunciation and prosody.

4 The Structure and Functionality of the Program

The application is built upon a hybrid architecture designed for both performance and interactivity. The User Interface (UI) and I/O operations are handled by **Electron** (TypeScript), while the computationally intensive tasks—acoustic modeling and alignment—are offloaded to a high-performance **C++ Native Addon** (N-API).

4.1 Core Workflow: Pronunciation Assessment

The primary function of the system is the deterministic scoring of user speech. This process follows a strict pipeline: Audio Input → Acoustic Inference → Forced Alignment → GOP Scoring.

4.1.1 Audio Input and Preprocessing

The workflow begins when the user records audio in the frontend. The data is transmitted to the main process as a raw Float32Array (PCM data) to avoid disk I/O overhead. Inside the C++ backend, the audio is preprocessed to match the model’s requirements: it is mixed to mono, resampled to 16,000Hz, and normalized.

```
// SpeechService.ts
// Efficient memory transfer of PCM data to C++ backend
public analyzeRaw(
  pcmData: Float32Array,
  sampleRate: number,
  channels: number,
```

```
  text: string): AnalysisResult {
    // ...
    // Calls the native C++ method directly
    const result = this.engine.analyze(pcmData, sampleRate, channels, text)
    return result
}
```

4.1.2 Acoustic Inference (Wav2Vec 2.0)

The preprocessed audio is fed into the Wav2Vec 2.0 model running via ONNX Runtime in C++. The model outputs a matrix of logits (log-probabilities) representing the likelihood of every token in the vocabulary for each time frame ($T \times V$ matrix).

```
// ModelRunner.cpp
// Run inference using ONNX Runtime
bool ModelRunner::runInference()
{
  try {
    // 1. Prepare Input Tensor (Zero-copy from audio vector)
    std::vector<int64_t> input_shape = { 1, (int64_t)audio.size() };
    Ort::Value input_tensor = Ort::Value::CreateTensor<float>(...);

    // 2. Run Inference using ONNX Runtime
    const char* input_names[] = { "input_values" };
    const char* output_names[] = { "logits" };
    auto output_tensors = session->Run(..., input_names, &input_tensor, 1,
output_names, 1);

    // 3. Extract dimensions and raw logits
    auto dims = output_tensors[0].GetTensorTypeAndShapeInfo().GetShape();
    this->time_steps = (int)dims[1];
    this->vocab_size = (int)dims[2];

    const float* raw_logits = output_tensors[0].GetTensorData<float>();
    output_log_probs.assign(raw_logits, raw_logits + (time_steps *
vocab_size));

    // 4. Compute Log-Softmax for Viterbi decoding
    computeLogSoftmax();

    return true;
  }
  catch (const Ort::Exception& e) {
    // Handle ONNX Runtime exceptions
    return false;
  }
}
```

4.1.3 Phonemization and Forced Alignment

Parallel to inference, the target text (e.g., “Hello”) is converted into a sequence of phonemes (International Phonetic Alphabet) using libespeak-ng.

To map the variable-length audio to this fixed phoneme sequence, the system employs the Viterbi Algorithm. It constructs a trellis based on the acoustic model’s output and backtracks to find the optimal path, determining the precise start and end frames for each phoneme.

```
// Align.cpp: Core Viterbi Forward Pass
for (int t = 1; t < T; ++t) {
    for (int s = 0; s < S; ++s) {
        // ... Calculates transition probabilities ...
        if (best_prev != -1) {
            dp[t * S + s] = max_score + current_prob;
            backtrack[t * S + s] = best_prev;
        }
    }
}
```

4.1.4 Goodness of Pronunciation (GOP) Scoring

After alignment, the system computes the GOP score. The implementation aggregates the log-probabilities for all frames assigned to a specific phoneme and normalizes them by the duration.

```
// Align.cpp: Calculating duration-normalized log-posterior probabilities
for (int t = 0; t < T; ++t) {
    if (path_states[t] == target_state_s) {
        if (start_t == -1) start_t = t;
        end_t = t;
        sum_log_prob += runner.getLogProb(t, target.token_id);
        count++;
    }
}
if (count > 0) {
    detail.score = sum_log_prob / count; // The GOP formula
}
detail.is_good = (detail.score > THRESHOLD_GOOD);
```

4.2 AI Coaching: Multimodal LLM Integration

While GOP provides objective accuracy, *GPT-4o-realtime-preview* is used for qualitative coaching. The LLMService manages a WebSocket connection to OpenAI, sending the user’s audio to be analyzed by a virtual coach.

The system maintains a persistent WebSocket connection to ensure low-latency interactions. Upon connection, the system defines the AI’s persona via a session.update event and subsequently pushes the audio data within a conversation.item.create event.

```
// LLMService.ts: Transmitting audio and prompt to GPT-4o
ws.on('open', () => {
  // 1. Configure the session instructions
  ws.send(JSON.stringify({
    type: 'session.update',
    session: {
      modalities: ['text'],
      instructions: "You are a professional oral coach. Analyze phonetics
and intonation..."
    }
  }));

  // 2. Send the processed audio data
  ws.send(JSON.stringify({
    type: 'conversation.item.create',
    item: {
      type: 'message',
      role: 'user',
      content: [
        { type: 'input_text', text: prompt },
        { type: 'input_audio', audio: base64Audio } // The transcoded audio
      ]
    }
  }));

  // 3. Trigger the model response
  ws.send(JSON.stringify({ type: 'response.create' }));
});
```

This integration allows the application to capture nuances such as improper word stress or flat intonation—features that are often missed by traditional ASR systems but are essential for mastering a second language.

4.3 Auxiliary Functionality: Azure TTS

To provide a “Gold Standard” for imitation, the application utilizes Microsoft Azure TTS. The service supports multiple languages (English, Chinese, French, etc.) and uses SSML to ensure high-fidelity neural voice synthesis.

```
// TTSService.ts: Constructing SSML for high-quality neural speech
private buildSSML(text: string, langCode: string): string {
    const config = this.getVoiceConfig(langCode);
    return `
      <speak version='1.0' xml:lang='${config.lang}'>
        <voice xml:lang='${config.lang}' name='${config.voice}'>
          ${escapedText}
        </voice>
      </speak>
    `.trim();
}
```

5 Performance Evaluation

Since a labeled "Golden Standard" dataset for non-native speech was not available for this project, the evaluation focuses on a qualitative assessment of the scoring behavior, alongside quantitative measurements of system latency and resource efficiency.

5.1 Qualitative Assessment of Scoring Accuracy

The system's Goodness of Pronunciation (GOP) metric was tested against live user recordings across different languages.

- **General Alignment Accuracy**: The Viterbi algorithm successfully synchronized audio with text in the majority of test cases, correctly identifying the start and end frames of phonemes even in continuous speech.
- **Sensitivity Observation**: The system exhibits high sensitivity to articulation errors. However, a phenomenon of "False Rejection" was observed during testing: specifically, certain phonemes sometimes receive abnormally low scores (near the threshold of -10.0) even when the user's pronunciation appears intelligible to a human listener.
  - **Analysis**: This behavior is likely due to the "Acoustic Mismatch" between the user's recording environment (microphone quality, background noise) and the clean studio data used to train the original Wav2Vec 2.0 model.
  - **Mitigation**: To prevent discouraging users, the system implements a "filtering" logic where extremely low scores (below -9.0) are excluded from the word-level average calculation.

5.2 Evaluation of Multimodal AI Coaching

The system utilizes *GPT-4o-realtime-preview* to provide holistic feedback. While the model demonstrates strong capabilities in understanding natural language and detecting prosodic features (intonation, rhythm), two significant limitations were observed during testing:

- **Limited Precision on Subtle Phonetics**: Unlike the local Wav2Vec 2.0 engine, which analyzes raw acoustic frames against a fixed phoneme set, the multimodal LLM processes audio tokens in a more abstract manner. Consequently, it often struggles to accurately identify subtle articulatory errors (e.g., distinguishing between very similar vowels like /i/ and /ɪ/), occasionally offering vague or generic advice rather than precise corrective instructions.
- **"Pedagogical Hallucination" (Hyper-Correction)**: A notable issue observed is the model's tendency to generate false positives when analyzing near-perfect speech. Driven by the system prompt which explicitly instructs it to "point out non-standard pronunciation", the model occasionally "hallucinates" errors in native-level speech, fabricating critiques to fulfill its role as a coach. This suggests that while the model is excellent for conversation and fluency practice, it cannot yet fully replace deterministic algorithms (like GOP) for high-stakes accuracy assessment.

5.3 System Latency (Response Time)

A key requirement for oral practice is immediate feedback. We measured the "End-to-End Latency" from the moment the user stops recording to the display of the score.

- **Local Inference Pipeline**: Thanks to the C++ Native Addon , the overhead is minimal. The **average processing time** is approximately 830 ms for a standard 5-second sentence.This confirms that the local ONNX Runtime implementation provides a near-real-time experience for segmental scoring.
- **Cloud Coaching Pipeline**: The **average response time** is approximately 7000 ms. The API call overhead and network latency contribute significantly to this delay.

5.4 Resource Utilization

The application was tested on Windows 11 with Intel i7-12700H CPU and 32GB RAM.

- **Memory Footprint**: The application consumes approximately 1300 MB of RAM. The majority of this is allocated to the Electron renderer and the loaded ONNX model weights.
- **Stability**: The system remained stable during repeated inference calls, with the C++ destructor correctly managing memory cleanup to prevent leaks.

6 Discussion

6.1 Advantages of the Proposed System

1. **Hybrid Feedback Mechanism (Deterministic + Generative)**: Unlike traditional apps that offer only one type of feedback, this system combines the best of both worlds:
   - **Precision**: The local Wav2Vec 2.0 + GOP engine provides objective, phoneme-level scores, pinpointing exactly which sound was mispronounced.
   - **Pedagogy**: The GPT-4o integration provides human-like, holistic advice on intonation and rhythm, covering the "suprasegmental" features that GOP algorithms often miss.
2. **Multilingual Architecture**: By decoupling the acoustic model (Wav2Vec 2.0) from the phonemizer (eSpeak-NG), the system supports a wide range of languages (English, Chinese, Japanese, French, etc.) without needing to redesign the core scoring logic.
3. **Low Latency Implementation**: The use of a C++ Native Addon (N-API) allows for high-performance numerical computing that would be too slow in pure JavaScript. Features like direct memory mapping (Float32Array) eliminate the need for slow disk read/write operations during analysis.

6.2 Limitations and Areas for Improvement

1. **Score Instability (The "False Rejection" Issue)**: As observed in the evaluation, the GOP metric is a measure of "acoustic confidence," not necessarily "human intelligibility." If a user speaks with a valid accent that differs significantly from the model's training data, or if the microphone quality is poor, the model may output extremely low probabilities, resulting in unfair penalties.
2. **Dependency on Forced Alignment**: The accuracy of the score is entirely dependent on the Viterbi alignment. If the user skips a word, pauses too long, or stutters significantly, the alignment path may drift, causing the scoring of subsequent phonemes to be incorrect.

3. **Resource Intensity**: Running a Transformer-based model (Wav2Vec 2.0) locally requires significant RAM and CPU power compared to lightweight cloud-based solutions. This may limit the application's performance on older or lower-end hardware.
4. **Complexity of Environment Setup**: The hybrid architecture (Electron + Native C++ + Python dependencies for training) increases the complexity of distribution and installation, as the native modules must be compiled for the specific target operating system.

Bibliography

[1] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations.” [Online]. Available: https://arxiv.org/abs/2006.11477

[2] A. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967, doi: 10.1109/TIT.1967.1054010.

[3] S. Witt and S. Young, “Computer-assisted Pronunciation Teaching based on Automatic Speech Recognition,” p. , 1997.