Key Points

- Research suggests using Automatic Speech Recognition (ASR) systems with forced alignment can improve pronunciation correction, avoiding issues with your current G2P and grammar approach.
- It seems likely that open-source tools like CMUSphinx or commercial APIs like Speechace can help detect and correct pronunciation errors effectively.
- The evidence leans toward integrating feedback mechanisms, such as TTS, to guide users in real-time, preserving sentence meaning.

Direct Answer

Understanding Your Challenge

It sounds like your current speech pronunciation correction system, using G2P conversion followed by grammar correction, isn't working as expected, especially since it can produce sentences that lack meaning. This is a common challenge, and it's great that you're looking for a better way to help users improve their pronunciation while keeping their intended message clear.

A Better Approach

Instead of relying on G2P and grammar correction, consider using **Automatic Speech Recognition (ASR)** systems that focus directly on pronunciation errors. These systems compare your speech to a model of correct pronunciation using a process called **forced alignment**, which aligns the audio with expected phonemes. This method helps identify specific pronunciation mistakes without altering the sentence's meaning, addressing the issue you've encountered.

Tools to Try

- **Open-Source Option**: Try **CMUSphinx**, an open-source toolkit that supports pronunciation assessment. It allows you to align speech with expected phonemes and provides scores to highlight errors. You can set it up by installing it on your system and creating pronunciation dictionaries, then use it to analyze audio recordings.

creating pronunciation dictionaries, then use it to analyze audio recordings .

- **Commercial APIs**: If you prefer a ready-made solution, consider **Speechace** or **Pronounce**, which offer APIs for pronunciation assessment and feedback, making integration into your app easier .

How It Works

1. **Transcribe Speech**: Use ASR to convert spoken words into text.
2. **Align and Compare**: Align the audio with the expected phonetic sequence to detect errors, like substituting “th” with “s” in “think.”
3. **Provide Feedback**: Use Text-to-Speech (TTS) to play the correct pronunciation and highlight mistakes in the UI, allowing users to practice and improve.

Why This Helps

This approach directly targets pronunciation errors, preserves the original meaning, and provides clear, user-friendly feedback. It’s scalable and can be customized for your app, whether you build it from scratch with CMUSphinx or use APIs for quicker deployment.

If you’re interested in cutting-edge methods, research also suggests using Reinforcement Learning and Large Language Models for personalized feedback, though this may require more development effort

.

Let me know if you need help with specific steps, like setting up CMUSphinx or integrating an API!

Survey Note: Exploring Alternative Approaches for Speech Pronunciation Correction Systems

This note provides a comprehensive analysis of alternative methods for creating a speech pronunciation correction system, addressing the limitations of the user’s current approach using Grapheme-to-Phoneme (G2P) conversion followed by grammar correction. The current method has been identified as producing sentences that lack proper meaning, which

current method has been identified as producing sentences that lack proper meaning, which grammar correction alone cannot resolve. This survey explores research, tools, and implementation strategies to offer a robust, user-friendly solution, focusing on direct pronunciation error detection and feedback.

Background and Problem Analysis

The user's current system relies on G2P conversion, which transforms text into phonetic representations, followed by grammar correction to refine the output. However, this approach has significant drawbacks:

- **G2P Limitations**: G2P does not account for context or meaning, leading to incorrect transcriptions, especially for homophones (e.g., "read" vs. "red") or non-standard pronunciations. This can exacerbate pronunciation errors rather than correct them.
- **Grammar Correction Misalignment**: Grammar correction focuses on syntactic structure, not phonetics. If the initial transcription is wrong due to mispronunciation, grammar correction may produce semantically incorrect sentences, deviating from the user's intent.
- **Loss of Meaning**: The multi-step pipeline (speech-to-text → G2P → grammar correction) introduces cumulative errors, resulting in outputs that fail to preserve the original message.

Given these issues, the goal is to find an alternative that directly addresses pronunciation errors while maintaining semantic accuracy, suitable for integration into a user-facing app.

Alternative Approach: Pronunciation-Focused Speech Correction

Research suggests that a more effective method is to use **Automatic Speech Recognition (ASR)** systems with **forced alignment** for pronunciation error detection. This approach compares the user's spoken audio to a model of correct pronunciation at the phoneme level, providing targeted feedback without altering the sentence's meaning. Key components

providing targeted feedback without altering the sentence's meaning. Key components include:

- **Real-Time Speech Transcription**: Use ASR to convert spoken words into text, segmenting audio into phrases or words for analysis. Tools like Mozilla DeepSpeech, OpenAI Whisper, or CMUSphinx can handle diverse accents and noisy inputs.
- **Forced Alignment**: Align the transcribed text with the audio to identify pronunciation errors. This involves mapping the recognized phonemes to the expected phonetic sequence, flagging discrepancies (e.g., substituting /θ/ with /s/ in “think” pronounced as “sink”). Tools like the Montreal Forced Aligner (MFA) or CMUSphinx’s alignment features are commonly used.
- **Pronunciation Error Classification**: Classify errors (e.g., substitution, deletion, insertion) using a pronunciation lexicon (e.g., CMUdict) or pre-trained models. Calculate scores based on acoustic alignment and edit distance to quantify error severity.
- **Feedback Mechanism**: Provide auditory and visual feedback, such as highlighting mispronounced phonemes in the UI and using Text-to-Speech (TTS) systems (e.g., NVIDIA NeMo TTS, Google Cloud TTS) to play correct pronunciations. Allow users to retry and track progress over time, potentially with gamification (e.g., scores, badges).

This approach directly targets pronunciation errors, preserves the original meaning, and offers user-centric feedback, addressing the limitations of the G2P and grammar correction pipeline.

pipeline.

Tools and Technologies for Implementation

Given the user's need to build an app, both open-source and commercial solutions are viable. Below is a detailed breakdown:

Open-Source Tools

- **CMUSphinx**:
  - An open-source speech recognition toolkit with built-in support for pronunciation assessment through forced alignment. It uses Hidden Markov Models for ASR and provides acoustic scores for phoneme alignment.

  - **Implementation Steps**:
    - **Installation**: Install on Debian/Ubuntu with `sudo apt-get install automake python python-dev swig`, then build from source using SVN repositories for sphinxbase and pocketsphinx. For non-root, use `--prefix=$HOME/pocketsphinx --without-python`. On MacOSX, install autoconf, automake, and libtool first .
    - **Configuration**: Create pronunciation dictionaries (e.g., `phonemes.dict` and `words.dict`) using CMUBET (e.g., `aa AA`). Use JSGF files for alignment, such as `with-align.jsgf` for phoneme alignment.
    - **Running**: Use `pocketsphinx_continuous` with parameters like `-infile with.wav -jsgf with-align.jsgf -dict phonemes.dict -backtrace yes -fsgusefiller no -bestpath no > with-alignment.txt` for forced alignment. Analyze output for acoustic scores and durations to identify mispronunciations.
    - **Output Analysis**: Acoustic scores (log-probabilities, negative, larger closer to zero for confidence) and durations are key. Use `-bestpath no` for phoneme/word alignment, `-bestpath yes` for neighbors. Parse for missing scores and `(NULL)` entries to detect errors.
    - **Pronunciation Assessment**: Use logistic regression with variables like phoneme acoustic scores, durations, recognized vs. expected phonemes, and word scores.

Convert to standard scores after log transformation for normal distribution. Example data and scripts (e.g., wyn.tar.gz from GitHub Repository) provided for 80 utterances.

- **Correlations**: Kendall's tau rank correlations for methods show strong relationships, as seen in the table below:

Kendall's tau non-parametric rank correlation	A	N	P	W
A: acoustic score of entire phoneme alignment	1	0.44	0.74	0.85
N: number of expected phonemes among neighbors	0.44	1	0.57	0.47
P: mean standardized phoneme acoustic scores	0.74	0.57	1	0.71
W: acoustic score of word alignment	0.85	0.47	0.71	1

- **Troubleshooting**: For "Final result does not match the grammar" error, use `<ss>` instead of `<s>`, or adjust `-wbeam` (default 7e-29, try 1e-56) and `-beam` (default 1e-48, try 1e-57). Optimization tasks and long audio aligners are available .
- **References**: Key papers include Loukina et al. (2015, Interspeech Paper), Kibishi et al. (2014, Tut.ac.jp Paper), and Ronanki et al. (2012, ACL Anthology).

- **Mozilla DeepSpeech**:
  - Another open-source ASR toolkit that can be fine-tuned for pronunciation correction. Provides transcription and can be extended with phonetic alignment tools like MFA.

Commercial APIs

- **Speechace**:
  - A speech recognition API designed for pronunciation and fluency assessment, used by educational institutions. Provides immediate, pinpointed feedback on mistakes, suitable for language learning apps .
- **Pronounce**:
  - A free speech checker using AI for English pronunciation improvement, offering

grammar correction and CEFR guidance. Easy to integrate for user-friendly feedback .

Research-Based Advanced Methods

- **Reinforcement Learning (RL) and Large Language Models (LLMs)**:
  - Recent research, such as “Automated speech therapy through personalized pronunciation correction using reinforcement learning and large language models,” shows promise. It uses RL (e.g., Proximal Policy Optimization) for precise pronunciation evaluation and LLMs for detailed, user-specific feedback. Evaluated on datasets like CMU Sphinx Dictionary, TIMIT, LibriTTS, SpeechOcean762, and RAVDESS, achieving 97.9% phoneme-level accuracy, 87.7% word-level accuracy, 95.2% syllable count accuracy, and 89.4% perfect accuracy on CMU Sphinx .
  - This approach is modular, supporting adaptation to other languages and dialects, but may require significant development effort.

Implementation Roadmap

To build the system, follow these steps:

1. **Choose an ASR Toolkit**:
    - Start with CMUSphinx for its open-source nature and detailed pronunciation assessment capabilities. Alternatively, use Speechace or Pronounce APIs for quicker integration.
2. **Set Up Pronunciation Dictionaries**:
    - Use CMUdict for standard pronunciations. Create custom dictionaries for specific use cases, ensuring coverage of expected words.
3. **Record and Transcribe Speech**:
    - Use real-time audio processing libraries like PyAudio for capturing speech. Transcribe using the chosen ASR toolkit, ensuring audio is recorded at 16,000 Hz, 16 bits, mono for CMUSphinx compatibility.
4. **Perform Forced Alignment**:
    - Align the transcribed text with the audio using CMUSphinx or MFA. Analyze acoustic scores and durations to identify pronunciation errors, flagging discrepancies like missing phonemes or low-confidence alignments.

5 **Detect and Classify Errors**:

- Compare aligned phonemes with expected phonemes from the dictionary. Classify errors (e.g., substitution, deletion) and calculate scores using edit distance or logistic regression for assessment.

6 **Provide Feedback**:

- Highlight mispronounced words or phonemes in the UI. Use TTS systems like NVIDIA NeMo TTS to play correct pronunciations. Offer a “repeat and compare” feature for iterative practice.

7 **Iterate and Improve**:

- Track user progress by storing error types and improvement metrics. Integrate gamification (e.g., scores, badges) to enhance engagement, encouraging consistent practice.

Comparative Analysis of Tools

Below is a table comparing the key features of the recommended tools:

Tool	Type	Languages Supported	Ease of Integration	Cost	Key Features
CMUSphinx	Open-Source	English (expandable)	Medium (requires setup)	Free	Forced alignment, pronunciation scoring, customizable dictionaries
Mozilla DeepSpeech	Open-Source	English (expandable)	Medium (fine-tuning needed)	Free	Robust transcription, extensible for pronunciation correction
Speechace	Commercial API	Multiple (Education-focused)	High (API-based)	Paid (API key)	Immediate feedback, fluency assessment, used by institutions
Pronounce	Commercial API	English	High (API-based)	Free (limited),	AI feedback, grammar correction, CEFR

Paid (Premium)	guidance

Benefits and Challenges

- **Benefits**: This approach directly targets pronunciation errors, preserves meaning, and provides user-centric feedback. It's scalable, with open-source options like CMUSphinx allowing customization and commercial APIs offering quick deployment.
- **Challenges**: Implementing CMUSphinx requires technical setup and may need optimization for real-time performance. Commercial APIs may have costs or usage limits, while advanced methods like RL and LLMs require significant development effort and expertise.

Supporting Research and Resources

Research supports the effectiveness of ASR-based pronunciation correction, with studies like "Automatic Pronunciation Scoring And Mispronunciation Detection Using CMUSphinx" provide insights into phonetic assessment methods.

For practical implementation, the CMUSphinx wiki

· can be used for testing.

Conclusion

By shifting from G2P and grammar correction to ASR-based forced alignment, you can create a more effective speech pronunciation correction system. CMUSphinx is a strong open-source option, while Speechace and Pronounce offer commercial alternatives. For advanced personalization, consider exploring RL and LLMs, though this may require more effort. This approach ensures direct error detection, preserves meaning, and provides actionable feedback, enhancing user experience in your app.

Key Citations

- CMU US English Dictionary GitHub
- Automatic Pronunciation Scoring And Mispronunciation Detection Using CMUSphinx ACL Anthology
- CMU Lexicon Tool Website
- The CMU Pronouncing Dictionary Website
- Automatic Pronunciation Evaluation And Mispronunciation Detection Using CMUSphinx ResearchGate
- Pocketsphinx for Pronunication Evaluation CMUSphinx Wiki
- Tools for working with the CMU Pronunciation Dictionary GitHub
- Speechace Pronunciation and fluency assessment Website
- Professional English Speech Checker Pronounce Website
- Automated speech therapy through personalized pronunciation correction Research Paper
- NVIDIA NeMo TTS Documentation Website
- A non-native English corpus for pronunciation scoring task GitHub
- Long Audio Aligner CMUSphinx Blog
- Interspeech 2015 Paper on Pronunciation Assessment
- Tut.ac.jp Paper on Pronunciation Assessment

- CMUSphinx PocketSphinx Tutorial Website