NON-SPEECH SOUND DETECTION IN SPEECH RECOGNITION

Investigating methods to enhance the detection of non-speech sounds during automated speech recognition processes

Kevin Zhao, Dasang Dolma

Introduction

This poster presents initial findings from an ongoing project, which aims to improve the accuracy of non-speech sound detection (NSSD) in automated speech recognition (ASR) processes for the refinement of speech and language disorder screening methodologies in children.

Context & Significance

This project places a broader definition on non-speech sounds for the purposes of early speech screening.

TYPES OF NON-SPEECH SOUNDS (NSS)

- Filler words (*uh*, *um*, *mmm*)
- Pauses or lengthy closures

EXISTING ASR CAPABILITIES

Our team explored the following application programming interfaces (API) that already perform ASR, including CMU Sphinx, Whisper, Vosk etc. These interfaces are trained to only output clean transcriptions, where all NSS and speech irregularities are excluded from the final transcript.

Clean Transcription Produced by Current ASR API's: `I don't know about that one I will get back to you on that next week`

Target Full Verbatim Transcription: `um I don't know about that one [pause] I will get back to you on that uh next week`

Differences between current API transcriptions and our goal.

Other studies have identified ways to produce full verbatim transcriptions through complex models, often involving neural network modeling and audio descriptors. However, we aim to develop a program that takes advantage of the existing capabilities of transcription software and manipulate their outputs in order to simplify the ASR process beyond what previous studies have achieved with similar accuracy.

Methods

API OVERVIEW

In-depth research conducted on the existing capabilities of multiple ASR API's indicate that most are acoustically trained to ignore NSS, necessitating either an adaptation of the software's acoustic model (requiring 1,000+ hours of data) or developing a complex program that simulates adaptation results for correct NSSD. Due to time and resource constraints, we chose the latter option, establishing a pipeline utilizing both programs [Figure A].

CMU Sphinx and Whisper are used in our program development for the following reasons:

- While CMU Sphinx exhibits lower transcription accuracy, its settings are easily editable to allow for the detection of pauses with slight program adjustments.
- Whisper supports prompt engineering, where, when fed a near-correct transcription as input, a transcription with high NSSD accuracy is outputted.

BASELINE PROGRAM DEVELOPMENT

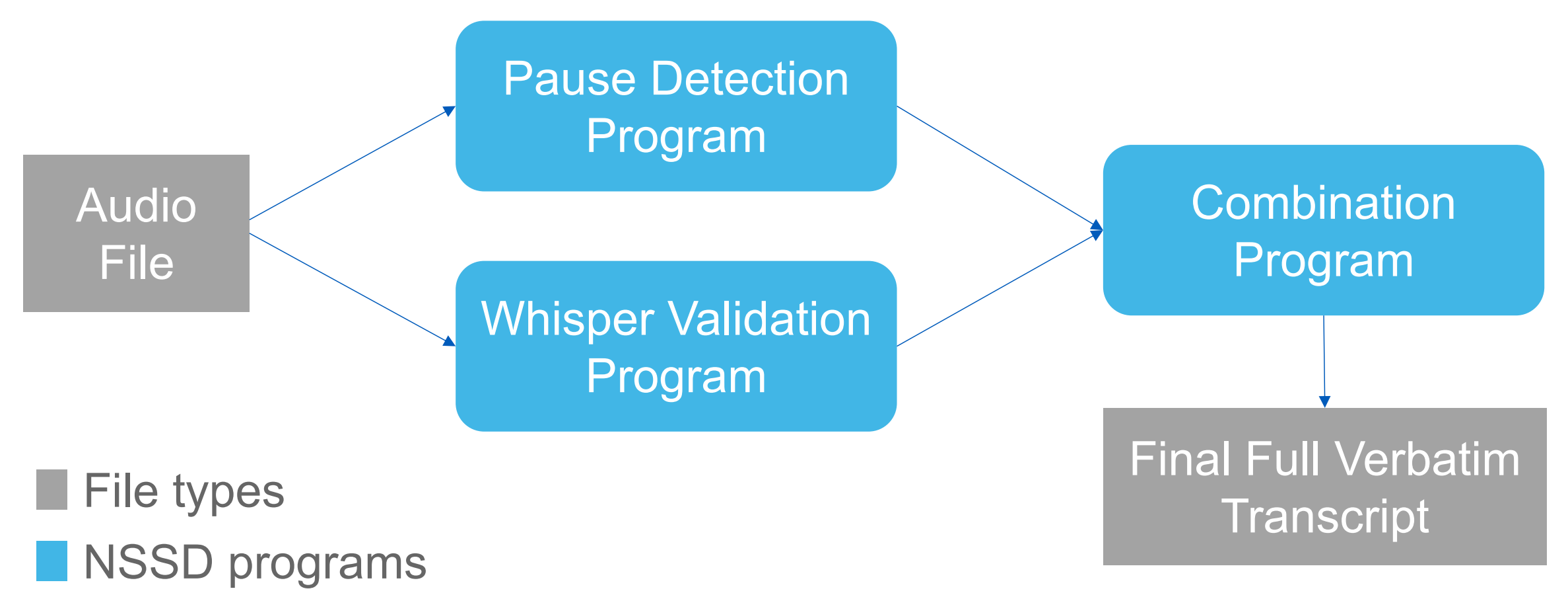
Initial construction of the NSSD program exclusively utilized CMU Sphinx to detect both pauses and NSS. However, the API's poor transcription accuracy, even with the addition of multiple filtration methods, prompted the integration of Whisper. Subsequently, the development timeline is divided into three stages: use of CMU Sphinx, adding filters, and combining CMU Sphinx & Whisper, detailed below:

A. **Pause detection program** – Uses CMU Sphinx to detect all possible pauses directly from a .WAV file, filtering the output with confidence scores to eliminate duplicate pauses and word/pause conflictions.

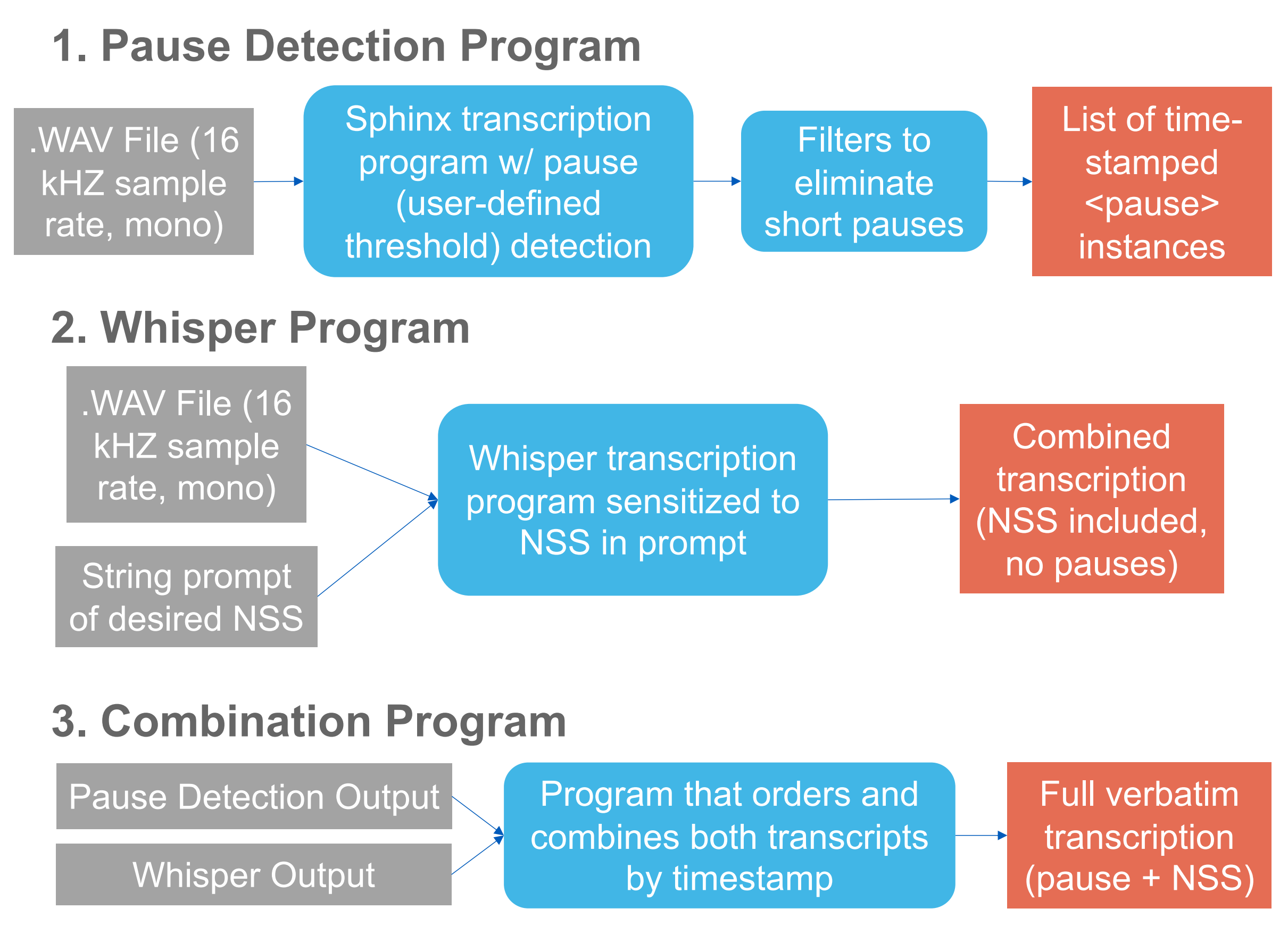
B. **Whisper program** - Prompts Whisper with NSS to detect (i.e. *"um uh mm"*) when transcribing original audio file. Prompt engineering sensitizes the API to the input phrase, mimicking an adaptation. As a result, the program outputs a full verbatim transcription (minus pauses).

In summary, the final approach couples two ASR tools and uses their unique features, to circumnavigate the complex adaptation process with high accuracy [Figure B].

Ⓐ General NSSD Pipeline Overview



Ⓑ Program Breakdown



1. Pause Detection Program
.WAV File (16 kHZ sample rate, mono)
Sphinx transcription program w/ pause (user-defined threshold) detection
Filters to eliminate short pauses
List of time-stamped <pause> instances
2. Whisper Program
.WAV File (16 kHZ sample rate, mono)
String prompt of desired NSS
Whisper transcription program sensitized to NSS in prompt
Combined transcription (NSS included, no pauses)
3. Combination Program
Pause Detection Output
Whisper Output
Program that orders and combines both transcripts by timestamp
Full verbatim transcription (pause + NSS)

Results

Testing was performed using 38+ recordings of randomly generated sentences and 0-12 instances of NSS, at different stages of the program development as we continuously modified our methods [Figure C]. The CMU Sphinx & Whisper program ultimately had the highest NSSD accuracy.

Note: Due to time constraints, testing exclusively involved 10-20 second clips recorded by a single speaker. As such further testing should be explored.

Ⓒ

Testing Results

Metrics (%)	Sphinx	Sphinx + Filters	Sphinx + Whisper
Correctly detected	3.11	9.55	85.71
Misdetected	8.00	6.18	5.65
Detected where no NSS	2.16	3.11	10.71
Overall accuracy rate	9.70	10.50	89.29

Conclusion

Preliminary results from this project suggest:

- **Current ASR Limitations:** Currently available ASR API's are incapable of detecting NSS due to manually filtered speech transcripts used when training its AI. Instead of the tedious process of adapting these acoustic models, various features integrated into such API's can be used to enhance full verbatim ASR.
- **NSS and Pause Detection:** Our program successfully detects pauses and NSS using a two-pronged approach utilizing existing API's with 90% accuracy.

FURTHER DIRECTIONS

- Further testing manipulating audio clip lengths, speaker demographics, and background noise
- Expansion of detection capabilities to include organic sounds (ie. sneezes, coughs, breath sounds)
- Integration of NSSD program with other team projects

References

1. Main pocketsphinx package — PocketSphinx 5.0.1 documentation. (n.d.). Pocketsphinx.readthedocs.io. Retrieved from https://pocketsphinx.readthedocs.io/en/latest/pocketsphinx.html
2. *OpenAI Platform*. (n.d.). Platform.openai.com. Retrieved from https://platform.openai.com/docs/guides/speech-to-text/prompting
3. Shmyrev, N. (n.d.). *Training an acoustic model for CMUSphinx*. CMUSphinx Open Source Speech Recognition. Retrieved from https://cmusphinx.github.io/wiki/tutorialam/