SPEECH TRANSCRIPTION RESEARCH

Project: Listening Effort Player
Date: 15 Feb 2024
Author: Tim Murray-Browne, Preverbal Studio

OBJECTIVE

I undertook research to investigate the feasibility of introducing automatic speech transcription into the Listening Effort Player app, built for the Pico Neo3 VR headset.

The app currently records audio from participants, and the transcription feature would additionally add a text transcription of what the participant says into the existing logfiles.

ANALYSIS

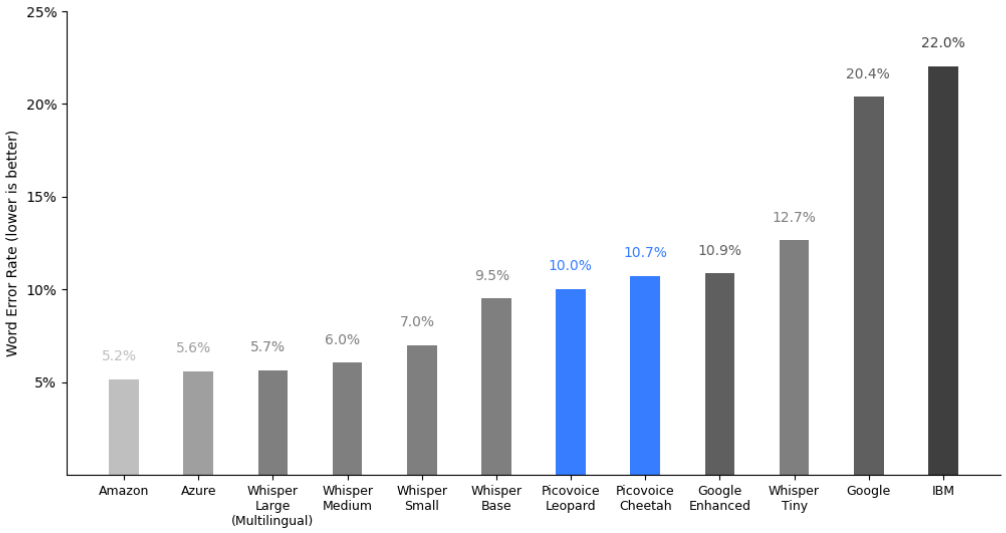
I considered three approaches:

1. Transcribing on a local PC after downloading logs from the device
2. Live on-device transcription on the Pico during user sessions
3. Cloud-based transcription service during user sessions

For both 1 and 2 I've prototyped using the *Whispercpp* model. *Whispercpp* is an open source CPU-based model based on Whisper which Open AI released. It runs on CPU which
means it can run on the Pico which does not have hardware for accelerated machine learning.

Whispercpp has multiple models. Bigger models require more CPU and give better results. From smallest to largest, they are: *tiny*, *base*, *small*, *medium*, *large*. Tiny through to small have English-only versions which give slightly better results. Further minor variations are also available (e.g. *tiny.en-q8_0*).

The following benchmark created by the rival firm Picovoice, suggests Whisper's *large* model is state-of-the-art level in terms of accuracy (measured as Word Error Rate).



Therefore I've stuck to Whispercpp for the following tests. I'm assuming the accuracy of cloud-based text-to-speech services will be comparable to Whispercpp with the *large* model.

Summary of findings

	Live on-device transcribe	Postprocess locally	Live cloud transcribe
Feasible	No	No	No
Attainable accuracy	Very low	Low	Low (estimated)
Cost	None	None	Low ongoing charges
Ongoing maintenance	Low	Low	Moderate
Privacy implications	None	None	Potential
Disruption to user experience	Potential	None	Low (internet connection required)

Prototype 1: Live on-device transcription

I prototyped live transcription performed on the Pico headset CPU while the app is being used. This is implemented on the **speech-recognition** branch of the repo.

In this prototype, the user waits after making their recording while the transcription is processed. In a final version, this processing could be offloaded to a background thread. However, it's not possible on the Pico to continue processing after the user has taken the headset off, so even with background processing, the user would need to remain in the app long enough for all recordings to finish processing.

The processing times per model on the Pico, as a multiple of the recording time, are as follows:

Model	On-device processing time per second of audio
tiny.en-q8_0	1.1s
tiny	2s
tiny.en	2s
small	23.4s
medium.en-q5_0	66s
large	(too slow to test)

With processing on a background thread, the *tiny* models are feasible, but all larger models would require the user to wait an unreasonable amount of time.

However, the accuracy of the *tiny* models were too poor to be usable. See below for more on this.

Prototype 2: Post-processing on a PC

In order to use larger Whisper models without disrupting the app, the investigator could perform the transcriptions at a later time from the audio recordings.

To prototype this, I wrote a Python script to transcribe all audio files in a log and write these to a CSV file. This script is available in the Utilities folder within the *whisper-postprocessor* branch of the *ListeningEffortPlayer* repo.

I tested the different models with 12 recordings of my own voice recorded by the app running on the Pico. I transcribed these shortly after recording them so my memory was able to assist.

When making the recordings, I tried to say exactly what I heard, irrespective of whether it seemed likely or not, to ensure the dataset also had negative results. ‘Ground truth’ in these tables is what I said, not what was in the video.

Ground truth	Whisper large-v2	Whisper medium.en	Whisper small	Whisper tiny	Whisper tiny.en
Thomas wants nine cheap beds	Thomas wants nine cheap bets.	Thomas wants nine cheap beds.	Thomas wants nine cheap vets.	Thomas once nine cheats bets.	Thomas wants 9 cheap bets.
Alan sold two green mugs	Alan sold two green mugs.	Alan sold two green mugs.	Alan solves two green mugs.	and then sold 2 green marks	and then salt two green mugs
Barry wants 12 old spoons	Barry wants 12 old spoons.	Barry wants 12 old spoons.	Barry wants 12 old spoons.	Barry once 12 old spoons	Barry wants 12-0 to experience.
Peter sees six pink chairs	Jesus is six pink chairs.	Jesus is six pink chairs.	Jesus is six pink chairs.	Titus is six pink chairs.	He just is six pink chairs.
Neil sold four green chairs	Neon salt for green chairs.	Neo Salt for green chairs.	Neil's Salt for Green Chance.	meal sold for green jazz	Peel salt full green chance.
Alan wants three dark rings	Alan wants three dark rings.	Alan wants three dark rings.	Alan wants three dark rings.	Alan wants three dark rings	Alan wants three dark rings.
Barry gives five red tests	Barry gives 5 red tests.	Barry gives five red tests.	Barry gives 5 red tests.	Dary gives the 5 red tests.	The array gives a 5-thread test.
Steven wins two big tins	Stephen wins two Big Tens.	Stephen wins two big tens.	Steven wins two big tins.	Steven wins two big tens.	Steven wins two big tenths.
Lucy wants eight meat beds	Lucy wants eight meat bags.	Lucy wants eight meat beds.	Lucy wants eight mute beds.	Lucy once ate meat pads	Lucy once ate meat first.

Lisa kept four green chairs	We accept four green chairs.	We accept four green chairs.	Reaccept 4 green chairs.	We make that 4kins chance.	We next up to four green chairs.
Rachel uses six cheap rings	Rachel uses six cheap rings.	Rachel uses six cheap rings.	Rachel uses six cheap rings.	Gradually use a 6-cheap rings.	Rachel gives us six cheap rings.
Anna gives two bald mugs	Anna gives two bald mugs.	Anna gives two bald mugs.	and it gives two bold mugs.	and I give 2 bold monks	And it gives two bolt marks.
Score	58.33%	66.67%	33.33%	8.33%	8.33%

The results indicate that even with the best Whisper model, the accuracy is too low to be feasible.

It's worth noting that the task at hand is particularly challenging for a speech-to-text model as there is no context to help distinguish between similar sounding words (as is the intended nature of the study).

Whisper does allow a textual context to be given that will prime its model before it starts. But it's unclear what kind of context would be useful here. If the model were primed with what we expect the participant to say, then it would bias the results as it would be less accurate when the participant gives a wrong answer.

The post-processing script may still be able to speed up the transcription process, but the investigator will still need someone to listen to each recording to correct any mistakes.

3 Cloud-based speech transcription service

In scenario 3, the App transmits the audio recordings during the user session to a cloud provider which then returns the speech. Pico themselves offer such a service, but it requires a complex registration with their Platform service. Also, there are rumours that Bytedance plans to wind down Pico soon. Bytedance themselves deny this, but it highlights the ongoing maintenance that comes from introducing a cloud-based dependency.

There are several reasons to prefer to avoid using cloud-based transcription on device:

- Ongoing maintenance in case the API changes or the service disappears.
- Ongoing subscription expenses. Most services charge by the minute, but are priced with an expectation of a significant amount of use.
- Users require an internet connection when using the app.
- Possible privacy implications as the participant's personal data is sent to a 3rd party provider.

Given these, and the indication that accuracy would be comparable to Whisper, I did not prototype any cloud-based solutions. It is possible that one provider emerges with a much higher accuracy in the future.

CONCLUSION

At this stage, it does not seem feasible to automate the transcription of participant recordings in the *ListeningEffortPlayer* app. This is because the state-of-the-art transcribers cannot handle the context-free sentences generated in the app with enough accuracy to eliminate the need for manual transcription.

I've developed two prototypes that may be taken further in the future as the technology develops. I don't expect the on-device transcription to ever be viable but the models available for post-processing may improve so as to be useful.