Goal

RootPi made a VR TraumaSim in unity and they wanted a way for the user to control the program using their voice instead of having to select options with controllers. The ideas was that the program would constantly listen to the microphone and when the user speaks we send it to an ai that can use tool calling to run some code. They had previously done this with OpenAI's gpt-realtime which can do the job but can get expensive.

The idea we were going to try now was using a Voice Activation Detection to tell when the user is speaking, then record that and save it as an audio file, then transcribe it to text, then send that to an AI that has tool calling then we can send it to our main program that runs the code.

The tools I ended up using was:

- https://github.com/wiseman/py-webrtcvad (Voice Detection)
- https://github.com/openai/whisper or https://platform.openai.com/docs/guides/speech-to-text(Transcribing)
- https://platform.openai.com/docs/guides/function-calling (tool/function calling)
- https://fastapi.tiangolo.com/advanced/websockets/ (websockets)

webrtcvad **=>** Audio File **=>** OpenAI Whisper **=>** ChatGPT + tool calling **=>** websocket event to main program

There are lots of other tools for transcribing, for example from Google and Amazon. I did not try them out but I imagine they are of similar cost and quality to OpenAI. Also we need a service that can handle swedish which only big models can do, so some tools are not useful.

Voice Detection

For voice detection I used https://github.com/wiseman/py-webrtcvad
When I first tested it with the example code, it did not work well. It would constantly detect a voice and I would end up with lots of 1 second audio files with no audio. This was because it would trigger with just 1 frame(20ms) of detection. So I changed the code so that it would only trigger if several frames in a row was detected as speech and I set this to be around 15 frames in a row. After this I think it worked a lot better.

However it still gets confused if any kinda loud sound happens that is continuous. For example a person coughing, drumming on a table. It can ignore some background noise but not these things.

Cost Calculations

Source: https://platform.openai.com/docs/pricing

gpt-3.5-turbo

input - 3$/1M tokens
output - 6$/1M tokens

for my early test i had 100 input tokens and 20 output tokens.

In the future we will have way higher input tokens but I think output will remain low.

with this example let's say we did 100 queries, that would cost

100 * 100 = 10000 input tokens = 0.03 $

20 * 100 = 2000 output tokens = 0.012 $

so 100 queries would cost **0.042 $** (very cheap)

Whisper-1 API

cost = $0.006 / minute

This seems very cheap since we are doing short audio segments and there seems to be no extra cost for each transcription. Let's say someone is using our tool for 2 hours, that would be:

0.006 * 120 = **0.72 $**

That is less than a dollar and is assuming the person is talking nonstop for the 2 hours which they probably won't do

Local vs API

I ended up testing Transcription both locally and via API. In both cases I used OpenAI's Whisper. The downside of running locally is that you need a good machine, I was not

able to run the best models on mine. The testing I was doing was on the size called "small". When doing a bigger size you get better results but the speed goes down.

Size	Parameters	English-only model	Multilingual model	Required VRAM	Relative speed
tiny	39 M	`tiny.en`	`tiny`	~1 GB	~10x
base	74 M	`base.en`	`base`	~1 GB	~7x
small	244 M	`small.en`	`small`	~2 GB	~4x
medium	769 M	`medium.en`	`medium`	~5 GB	~2x
large	1550 M	N/A	`large`	~10 GB	1x
turbo	809 M	N/A	`turbo`	~6 GB	~8x

In the documentation of the API they mention that you can stream the audio instead of saving a audio file. This is interesting and promising but I have not had time to try it out.

Overall API is probably better but it was interesting to try locally.