Blues for N0

An AI-bot playing the whisper game with itself.

Jonas Sjøvaag
University of Agder

I have built a bot that exists within a digital space. It was created to highlight errors in digital systems and to demonstrate the outcomes of such errors when they are interpreted anew, without conscious thought attached to them.

It currently utilizes speech-to-text (STT) and text-to-speech (TTS) technologies, one provided by ElevenLabs and the other by Google. Sometimes, I only use Google's STT because the default voice is completely free, which inadvertently points to another significant problem when dealing with technology: the best version comes at a cost.

In this bot, and in similar initiatives, the technical limitations described can be embraced as a method because simpler systems tend to reveal errors more readily. After all, errors are often what prompt humans to reflect. In a research setting, this is often desirable, and for artists, flaws and errors often serve as inspiration. I'm convinced that much of what we consider great art actually stems from attempts to rectify an idea that didn't initially work, or from abandoning the original idea in favor of a different approach.

For me, the bot started as a fun project but quickly proved useful in other contexts. I have created a piece for this poster presentation, closely echoing the theme in Alvin Lucier's "I am sitting in a room", and had the bot perform a five-minute piece at the Artistic Research Forum in Oslo, in March 2024. Currently, it only deals with textual output, but I'm considering expanding its functionality in one way or another. I could have spectral information extracted from audio, as a separate function in the script, or something along those lines, to get data that in turn can be used to trigger generative audio from SuperCollider, for instance, or be applied in other contexts.

The bot lives and expands, but it does so solely because of my ideas. This is probably the most poignant insight to be drawn from this experience. It is also very relatable.

Performance 1: in Rome
Performance 2: blues for No

www.jonassjovaag.no

```python
import os
import re
import glob
import librosa
import numpy as np
import soundfile as sf
import sounddevice as sd
import speech_recognition as speech_recog

def find_newest_audio_file(source_folder, file_extension="*.mp3"):
    list_of_files = glob.glob(f'{source_folder}/{file_extension}')
    if not list_of_files:
        return None
    latest_file = max(list_of_files, key=os.path.getmtime)
    return latest_file

def normalize_audio(audio_data, target_dBFS=-5):
    rms_current = np.sqrt(np.mean(audio_data**2))
    dBFS_current = 20 * np.log10(rms_current)
    target_amplitude = 10 ** ((target_dBFS - dBFS_current) / 20) *
    audio_normalized = audio_data * (target_amplitude / rms_curren
    return audio_normalized

def process_audio(file_path, folder="whispers"):
    y, sr = librosa.load(file_path, sr=None)
    y_normalized = normalize_audio(y, target_dBFS=-5)

    # Playback the normalized audio
    sd.play(y_normalized, sr)
    sd.wait()  # Wait until the playback has finished

    # Ensure the whispers directory exists
    os.makedirs(folder, exist_ok=True)
    output_path = os.path.join(folder, f"whisper_{len(os.listdir(f
    sf.write(output_path, y_normalized, sr)

    return output_path

def find_next_seq_num_text_output(folder, pattern="output_*.txt"):
    os.makedirs(folder, exist_ok=True)  # Ensure the folder exists
    files = glob.glob(os.path.join(folder, pattern))
    highest_num = 0
    for f in files:
        match = re.search(r'output_(\d+).txt', f)
        if match:  # Check if the search found a match
            num = int(match.group(1))
            if num > highest_num:
                highest_num = num
    return highest_num + 1

def remove_words(text, num_words=4):
    """
    Remove num_words words from the end of the text.
    """
    words = text.split()
    if len(words) >= num_words:
        return ' '.join(words[:-num_words])
    else:
        return ''

def recognize_speech_from_audio(audio_file_path):
    recognizer = speech_recog.Recognizer()
    with speech_recog.AudioFile(audio_file_path) as source:
        audio_data = recognizer.record(source)
    try:
        text = recognizer.recognize_sphinx(audio_data)
        text_output_folder = "text_output"
        next_num = find_next_seq_num_text_output(text_output_fo
        output_filename = os.path.join(text_output_folder, f'ou

        # Directly use the recognized text without limiting its
        full_text = text  # Use the full recognized text

        # Check for existing text and remove words if needed
        if os.path.exists(output_filename):
            with open(output_filename, "r") as file:
                existing_text = file.read()
            full_text = remove_words(existing_text)

        with open(output_filename, "w") as file:
            file.write(full_text + "\n")
        print(f"Transcribed text saved as {output_filename}")
    except speech_recog.UnknownValueError:
        print("Sphinx could not understand audio")
    except speech_recog.RequestError as e:
        print(f"Could not request results from Sphinx; {e}")
```