transcribvr

Author: Justin Martin

Revision: 0.1, 3/11/23

6. References

1. Transcribvr Overview

1.1. Description

Transcribvr is an application that implements Whisper [1] to create a web and GUI based transcription service for pre-recorded audio files up to a multi-hour length in a variety of formats and returns a transcript and associated metadata. The repository for transcibvr is here.

1.2. Purpose

The purpose of transcribvr is to create a fast, easy to use transcription application for recorded audio files. It is intended to make typical transcription tasks like interviews, digitization of audio records, and note taking easier and more effective.

1.3. Scope

Transcibvr is intended to implement an existing neural network for text transcription in an accessible and broadly deployable

The following capabilities are considered to be out of scope:

- Retraining any neural networks / transcription models

1.4. Features

F#	Title	Feature
Input Files		
F1.1	Audio Formats	Transcribvr shall support multiple audio formats, including popular [2] lossless, lossy, and uncompressed examples
F1.2	File Length	Transcribvr shall support audio files beyond whisper's input limitations
F1.3	File Batches	Transcribvr shall support batch processing of audio files
Output Files		

F2.1	Output Format	Transcribvr shall output plain text files containing the transcription with a metadata prefix
Performance		
F3.1	Reasonable Performance	Transcribvr shall run as a lightweight application requiring limited compute resources
Deployment		
F4.1	OSX Application	Transcribvr shall be runnable as a standalone, local GUI application on SOX
F4.2	Web tool	Transcribvr shall be deployed as web application on oblivrion

1.4.1.1. Table: Requirements

1.5. Requirements

Req #	Requirement	Description
F1.1: Transcribvr shall support multiple audio formats, including popular [2] lossless, lossy, and uncompressed examples		
R1.1.1	MP3	Transcibvr shall support this lossy format.
R1.1.2	M4a	Transcibvr shall support this lossy format.
R1.1.3	ACC	Transcibvr shall support this lossy format.
R1.1.4	Ogg Vorbis	Transcibvr shall support this lossy format.
R1.1.5	FLAC	Transcibvr shall support this lossless format.
R1.1.6	ALAC	Transcibvr shall support this lossless format.
R1.1.7	WAV	Transcibvr shall support this uncompressed format.
R1.1.8	AIFF	Transcibvr shall support this uncompressed format.

R1.1.9	DSD	Transcibvr shall support this uncompressed format.
R1.1.10	PCM	Transcibvr shall support this uncompressed format.
F1.2: Transcribvr shall support audio files beyond whisper's input limitations		
R1.2.1	>30s files	Transcibvr shall accept input files >30s and partition them into 30s parseable files
R1.2.2	Continuity	Transcibvr shall implement a strategy to maintain continuity between 30s audio files and prevent poor transcription at boundaries
R1.2.3	Long file streaming	Transcribvr shall report audio file transcription by 30s sample to prevent long transcription times without an output
F1.3: Transcribvr shall support batch processing of audio files		
R1.3.1	Multiple file inputs	Transcribvr shall accept multiple audio input files in a single batch with varying audio formats
R1.3.2	Zip file inputs	Transcibvr shall accept a zip file containing multiple audio files and produce a transcription report
R1.3.3	Multi File output	Transcribvr shall output the results from multi-file inputs in a file with separate metadata prefixes per file
F2.1: Transcribvr shall output plain text files containing the transcription with a metadata prefix		
R2.1.1	Metadata	Transcribv shall output a prefix containing relevant metadata for each audio sample, including the name of the audiofile, audiofile format, length of the audiofile, total transcription time, and [TBD Whisper metadata]
R2.1.2	Output file (application)	Transcribvr shall save the text file on the user's desktop and should allow the user to select an alternate location for the desktop application
R2.1.2	Output file (web)	Transcribvr shall return the text file as a text file that can easily be saved by the user

F3.1: Transcribvr shall run as a lightweight application requiring limited compute resources		
R3.1.1	Model size	Transcribvr shall select and run a reasonable whisper model size to reduce computational load
R3.1.2	Cloud deployment	Transcibvr shall utilize GPU resources to reduce the run time of the model
F4.1: Transcribvr shall be runnable as a standalone, local GUI application on OSX		
R4.1.1	Command Line	Transcibvr shall operate as a command line tool, accepting multiple command line arguments for file names and an output option to specify the saved file location
R4.1.2	GUI	Transcibvr shall provide a basic GUI implementing “select file” and “transcribe” buttons
F4.2: Transcribvr shall be deployed as web application on oblivrion		
R4.2.1	Start page	Transcribvr shall have a webpage with a “select file” and “transcribe” button to enable audio file selection, upload, and submission
R4.2.2	Results page	Transcribvr should return a formatted page displaying the text with a download link for the user

1.5.1.1. Table: Requirements

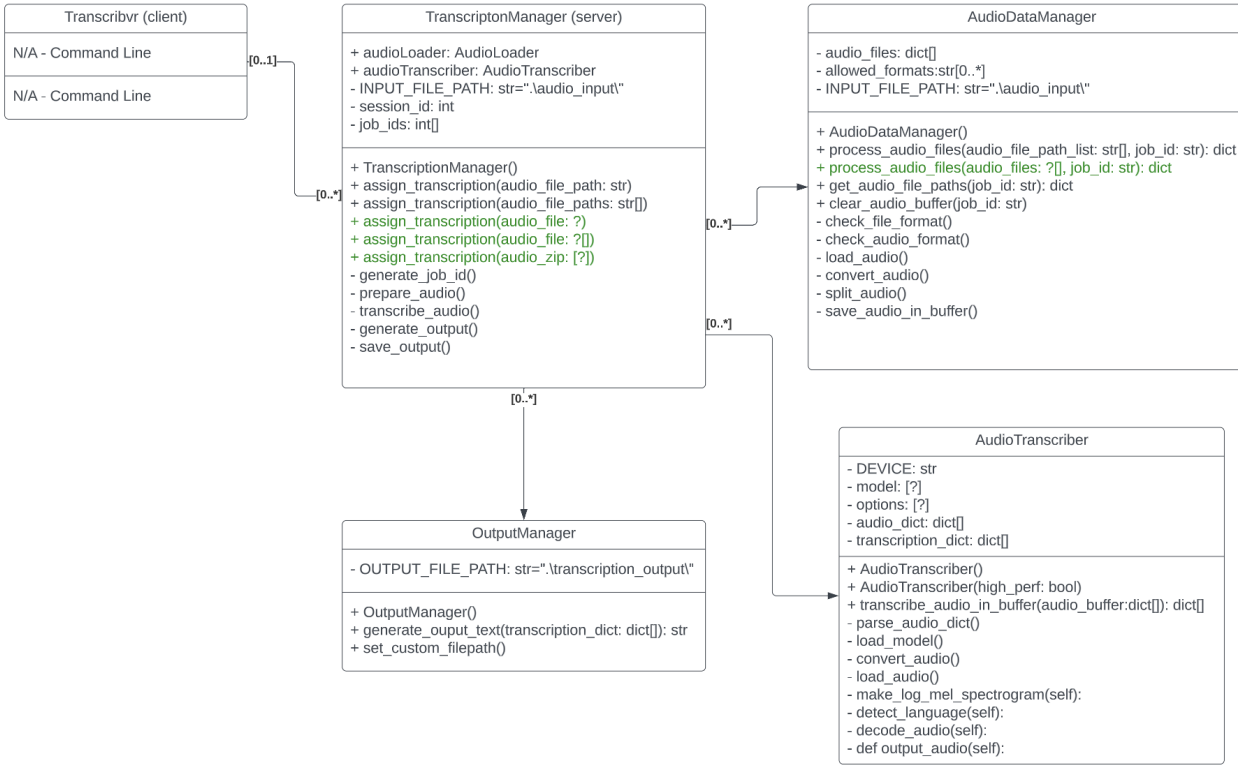
1.6. Further Work

F #	Features	Description
Input Files		
TBD	Input Streaming	Transcribvr shall run as a streaming application enabling live transcription
Output Files		

TBD	Output Report	Transcribvr shall output formatted transcription reports
Performance		
TBD	M2 Acceleration	Transcribvr shall utilize apple silicon resources to improve performance on OSX
TBD	Model quality selection	Transcribvr shall enable the user to select a higher performance transcription model
TBD	Multi Thread	Enable managing multiple simultaneous transcription tasks
Deployment		
TBD	Android	Transcribvr shall be runnable as an Android application
TBD	iOS	Transcribvr shall be deployed as an iOS application

1.6.1.1. Table: Requirements

2. System Architecture



2.1.1.1. Figure: Transcribvr Architecture Diagram

3. Software Subsystems

The Python application has four subsystems: TranscriptionManager acts as a server to manage the flow of data, while AudioDataManager handles audio data prep, AudioTranscribvr manages audio transcription, and OutputManager produces the final output of the job. Together, these subsystems provide an audio transcription service.

All subsystems shall conform to industry standards for software design documentation and adhere to best practices for maintainability, scalability, and reliability to ensure the long-term stability and success of the software application. The subsystems use clear comments, descriptive variable names, and organized code structure to make the code easy to understand and modify.

3.1. TranscriptionManager

The TranscriptionManager subsystem acts as the primary server and is responsible for receiving audio files, creating jobs, managing audio conversion, transcription, output

pre-processing, and exporting the results to the client. It coordinates the flow of data and interactions between the other subsystems.

3.2. AudioDataManager

The AudioDataManager subsystem manages audio data preparation, checking, and conversion. It receives a job ID and audio file paths or audio files as input and converts them to mp3s. The converted files are stored temporarily in a job file for processing. After transcription is complete, it clears the buffer to prevent unnecessary storage. The subsystem returns a dictionary/JSON object with the job ID, file names, and preliminary metadata.

3.3. AudioTranscriber

The AudioTranscribvr subsystem manages audio transcription for each job. It receives a dictionary with a job ID and file paths to the audio prepared by the AudioDataManager subsystem. It returns a dictionary/JSON object with the transcriptions and additional metadata added.

3.4. OutputManager

The OutputManager subsystem takes the audio transcription dictionary provided by the AudioTranscribvr subsystem to create the final output of the job. It returns the completed transcription to the TranscriptionManager subsystem for exporting to the client.

4. Data Architecture

4.1. Data References

The only data references used by transcribvr are the pre-trained models from [1]. These will be loaded once per audio transcribvr instance to reduce computational time and storage.

4.2. Persistent Data

The only persistent data for transcribvr will be logs, session IDs, job IDs, and performance information / metrics.

4.3. Transient Data

Audio files processed by transcribvr will need to be in an MP3 format. To enable this, audio files will be stored in a temporary on disk cache and converted to MP3s. These files will be removed after the job is completed.

Additionally, a dictionary will be built with metadata and transcription information for each job. In development, this may be saved for debugging and application improvements but in production, only a summary of each job will be logged.

4.4. External Interface Data

Data will be provided externally to transcribvr for the audio files that need to be processed. This can be provided in the following formats for the Command Line and GUI application:

- File path name for an audio file
- List of file path names for multiple audio files
- File path name for zipped folder of audio files
- List of file path names for zipped folder of audio files

For the web deployment of transcribvr, external audio data can be provided as actual audio files with a format to be specified in the future:

- Audio file
- Multiple audio files
- Zip file with audio file(s)
- Zip files with audio file(s)

Data returned to the client is in the form of an output transcription in the following formats:

- Text of metadata and transcription for job
- Text File containing metadata and transcription for job
- JSON data with results of transcription
- Formatted report with metadata and transcription

5. User Interface

5.1. GUI Application

5.1.1. Use Cases and Functions

U #	Use Case	Function
GUI Application		

U1	Load audio file(s)	Opens the file system navigator to select an audio file or files to be transcribed
U1	Submit a job	Starts the transcription process for the selected audio files
U1	Check job status	Displays the status of the current job, including whether it is in progress or complete
U1	Receive output	Displays the output to the user

Use cases:

- User opens the transcription application
- Application has two buttons (one for load audio, one for submit job) and an image displaying the status “waiting for job”
- User clicks a button to load audio
- File system navigator opens and user selects audio file, audio files, zip file, or zip files
- User closes file system navigator
- Status image displays “files selected”
- User clicks submit job
- Status image displays “in progress”
- After transcription finishes, status image displays “job complete, run another”?

5.1.2. UI Design and Flow

The transcription application has a simple user interface with two buttons, one for loading audio and one for submitting a job. The UI includes an image that displays the current job status, such as "waiting for job", "files selected", "in progress", or "job complete, run another".

5.1.3. UI Flow

1. User opens the transcription application
2. The UI displays an image with the status "waiting for job" and two buttons for loading audio and submitting a job
3. User clicks the "Load audio" button

4. The file system navigator opens, and the user selects an audio file, audio files, zip file, or zip files
5. User closes the file system navigator
6. The UI updates the status image to display "files selected"
7. User clicks the "Submit job" button
8. The UI updates the status image to display "in progress"
9. After the transcription is complete, the UI updates the status image to display "job complete, run another"
10. The user can repeat the process by clicking the "Load audio" button again.

6. References

[1] OpenAI Whispr Github

[2] Best Audio Format Typs