Audio Manipulation Report

Julian Evans, Nawl Maqbool, Owen Lindstrom
Macalester College
Comp 123
Elizabeth Ernst
May 7, 2022

User's Manual

This program might be useful for individuals who are interested in playing around with music, and people who would like to explore different aspects of music, for example, the separate vocals from the beat. In python the user must make sure when they unzip the file that all files attached are in the same location as the main py file. This consists of folders called "Songs", "Vocal", and "Beat". After our program's code is run, a window will be presented to the user, wherein they will be given a few song options, from which they can choose one to continue with. After the user chooses the song they want, a new window will pop up that will consist of options of what they want to do with the song. If for some reason the user wishes to select a different song before they select the option, they may click the "quit" button at the bottom of the screen, which will take them back to the list of songs. Once a song is selected the options the users may select include; "speed", speeding the song up and slowing it down, "reverse", going backwards, and then finally splitting up the "vocals" from the "beat". If the user selects the speed button a new window will appear, which presents a message prompting them to insert a number between 0 and 2. If the user, however, fails to do so, a message will appear saying; "You silly individual. Please enter a number between 0 and 2". When the user enters a number below 1 it will slow down the song. If the user selects a number above 1 it will speed up the song. If the user selects the reverse button instead, the operation will play right away. If the user selects the vocals or beats button, the operation will also play right away. For all songs, once an operation is selected, or a number is entered, the user must quit the program by stopping the program from running in python, in order to leave. If the user wishes to select a different song after they selected an operation, they must quit the program and run the program again. As long as the user does not select an operation, the user may move freely using the quit button throughout the program. The user interface for this project is very straightforward, and does not require the user to have much background knowledge on python or audiosegment. They will simply have to select their preferences.

Contents of program

To begin the project we download files and import packages in order for audio to work on the computer in python. We start off by downloading FFmpeg and FFprobe and using the terminal to find the direct link in which we can then type that into our environment variables. This allows all the packages we download to search python and cross through these files, allowing the packages to function properly. The packages we imported for this file included pydub, audiosegment, and tkinter. Once these packages were properly installed, we moved onto laying out our python file. We legally download songs from the internet into mp3 files in which we would later transfer into wav files. We transferred these song files from mp3 to wav because we had written code for audiosegment to operate with wav files. Once we had all of our wav files they were saved in a folder called “Songs” on our file “SoundManip”. We then started writing code for the speed and reverse functions. We used online resources to learn more about the audiosegment package which assisted us with writing these codes. Once that was completed we moved onto the spleeter function to split up the vocals from the beats. We were unable to write the code in the regular workspace, however, were able to do so in the terminal which spat out wav files for us to incorporate. We moved all the vocal wav files to a folder called “Vocal” and all the beat wav files to a folder called “Beat”. From here we were able to begin writing the user interface side of this program. For the user interface, we started a class in which our GUI would begin. Our first function ‘init’ took in one parameter, self. This allowed us to initialize the main window, user objects, and all its widgets. After defining the buttons and labels, we began defining each function for each operation. This included adding in our code from the change speed as well as reverse. This also included playing the vocals and beats for each song. We then created functions for each song. This allows us to add all the functions we have created thus far and implement them onto this function. The functions for each song created a new window and on this window it added a quit button, change speed button, reverse button, vocals button, and beats button and their individual functions. Creating these functions wrapped up our script smoothly and allowed everything to run without errors. Overall our script has about 3 main components within the class. The first one being creating a button for each song. Next the functions for each manipulation including; quit, speed, reverse, vocals and beats. Lastly was the functions for each song adding all the buttons and functions together.

When creating this program it was important we constantly ran our program to make sure of no huge errors. One thing that benefited us significantly was doing things individually before adding everything together. This allowed us to ensure that all of our code was correct before we implemented them together. We are confident this whole project runs smoothly except for when the user can enter a value between 0 and 2. The only error that might occur is if the user enters a string instead of numeric values then an error message will appear in the python console, however, the user's screen will appear normal.

Doing this project allowed us to actually implement the concepts we have been working with the entire semester. Even though we did not work with sound files in class we were able to use what we knew such as how to build a function, assignments, and GUIs to apply it to our project. The benefits of working on audio was that it showed us how much we were really able to accomplish. This course has given us the basic foundations to continue to create projects such as this one in the future. When planning out our project and the necessary steps we would need it didn't look too daunting, however, once we began writing the code we ran into problems all over the place. For starters just installing the packages and downloading files to run in python, was a very tedious process, and involved a lot of trial and error. Another surprising factor was how difficult it was to set up and get the GUI running. We did not realize how difficult it would be to incorporate a normal function with a GUI allowing them to co-operate. Nonetheless, after long hours and many days we were able to complete a working program.

Next time, we would do a lot more initial research and make a more detailed and organized plan because when starting the project we had a rough idea of what we needed, however, you never really know what you need until you dive into it. Now that we have created an entire project we know the types of things we need to look out for and plan ahead for in the future. If we were to change or add onto the project we would want to figure out how to write the spleeter code in the script area instead of the terminal. This was hands down the most time consuming part of the project, and if we were able figure out the code for it, it would have allowed us to add and incorporate more songs in less time. If we could figure that out we would want to download a whole bunch of songs in a separate dictionary file and give each song an assignment, so once the song is selected the program knows which song to play with its operations. This would be a much more efficient way to run our program.

References

Audiosegment module¶. audiosegment module - AudioSegment documentation. (n.d.). Retrieved May 4, 2022, from https://audiosegment.readthedocs.io/en/latest/audiosegment.html

Convert MP3 to WAV. Convert MP3 to WAV - Python Tutorial. (n.d.). Retrieved May 4, 2022, from https://pythonbasics.org/convert-mp3-to-wav/

Deshmukh, L. (2020, July 6). *Python music playback speed(1x,2x,3x) change without chipmunk effect.* Medium. Retrieved May 4, 2022, from https://medium.com/prog-ramming-solutions/python-music-playback-speed-1x-2x-3x-change-without-chipmunk-effect-890eb10826c1

How to use spleeter on macos 11.2020 - youtube. (n.d.). Retrieved May 3, 2022, from https://www.youtube.com/watch?v=cFj5heNlW98

How-to spleeter - youtube. (n.d.). Retrieved May 3, 2022, from https://www.youtube.com/watch?v=cC6kNcDoI2g

Installing ffmpeg and ffprobe on macos manually. macOS (manual ffmpeg installation) - Audio Orchestrator documentation. (n.d.). Retrieved May 4, 2022, from https://bbc.github.io/bbcat-orchestration-docs/installation-mac-manual/

itsMeitsMe 7111 gold badge11 silver badge77 bronze badges, Nombre ApellidoNombre Apellido 4111 bronze badge, tatarkintatarkin 3633 bronze badges, abhi krishnanabhi krishnan 1, & Lokesh DeshmukhLokesh Deshmukh 69855 silver badges1212 bronze badges. (1966, April 1). *How to change audio playback speed using Pydub?* Stack Overflow. Retrieved May 4, 2022, from https://stackoverflow.com/questions/51434897/how-to-change-audio-playback-speed-using-pydub

Jiaaro. (n.d.). *Jiaaro/pydub: Manipulate audio with a simple and Easy High Level Interface*. GitHub. Retrieved May 4, 2022, from https://github.com/jiaaro/pydub

ShuyinsamaShuyinsama 59111 gold badge44 silver badges66 bronze badges, slhckslhck 211k6363 gold badges572572 silver badges566566 bronze badges, & Meir ElishaMeir Elisha 111 bronze badge. (1960, July 1). *Install ffmpeg on OS X*. Super User. Retrieved May 4, 2022, from https://superuser.com/questions/624561/install-ffmpeg-on-os-x

Working with WAV files in python using pydub. GeeksforGeeks. (2022, January 18). Retrieved May 4, 2022, from https://www.geeksforgeeks.org/working-with-wav-files-in-python-using-pydub/

YouTube. (n.d.). YouTube. Retrieved May 3, 2022, from https://www.youtube.com/?gl=DE