Singing Data Labeling Tool

Semester Ⅱ Project Plan

Team Members:

- Avinash Persaud - apersaud2018@my.fit.edu
- Nandith Narayan - nnarayan2018@my.fit.edu
- Carlos Cepeda - ccepeda2018@my.fit.edu

Faculty Advisor:

- Dr. William Shoaff - wds@fit.edu

Client:

- Caleb Matthew Long, Appalachian State University

Meeting Dates with Client:

- Continuous contact via Discord.

Goal:

Our goal is to create a standalone and user-friendly tool to assist users with labeling singing data for vocal synthesis and machine learning purposes.

Motivation:

The current tools for labeling singing data that exist are sub-par; multiple tools are often required to label singing data and said tools aren't user friendly.

Approach:

Our tool will have the following functionalities:

- Allow the user to create mono labels graphically. Mono labels are a special label format that contains information about the data; this includes starttime, endtime, and a phoneme identifier. This allows the user to tag and label singing data as well as align phonemes, all in the same tool.
- Exporting the user generated label data in the HTS full label format. With the click of a button, the user will be able to export all their work into an output file.
- Automatically detect which phonemes are present in the audio. This will help reduce the amount of manual work the user has to do.
- Automatically align phonemes with the audio. The tool will detect where each phoneme must start and end, and place the phoneme accordingly. This will further reduce the amount of manual work the user has to do.
- Automatically save the user's work every few minutes to ensure the user has a backup in case data is ever lost.

- Allow the user to export an existing project. This will allow the user to transfer a project from one computer to another if needed.
- Allow the user to activate common tasks with the help of keyboard shortcuts. To label singing data, the user will need to perform certain actions numerous times. To make this process easier, the user can use shortcuts to perform actions such as coping, pasting, duplicating, moving, and so on.

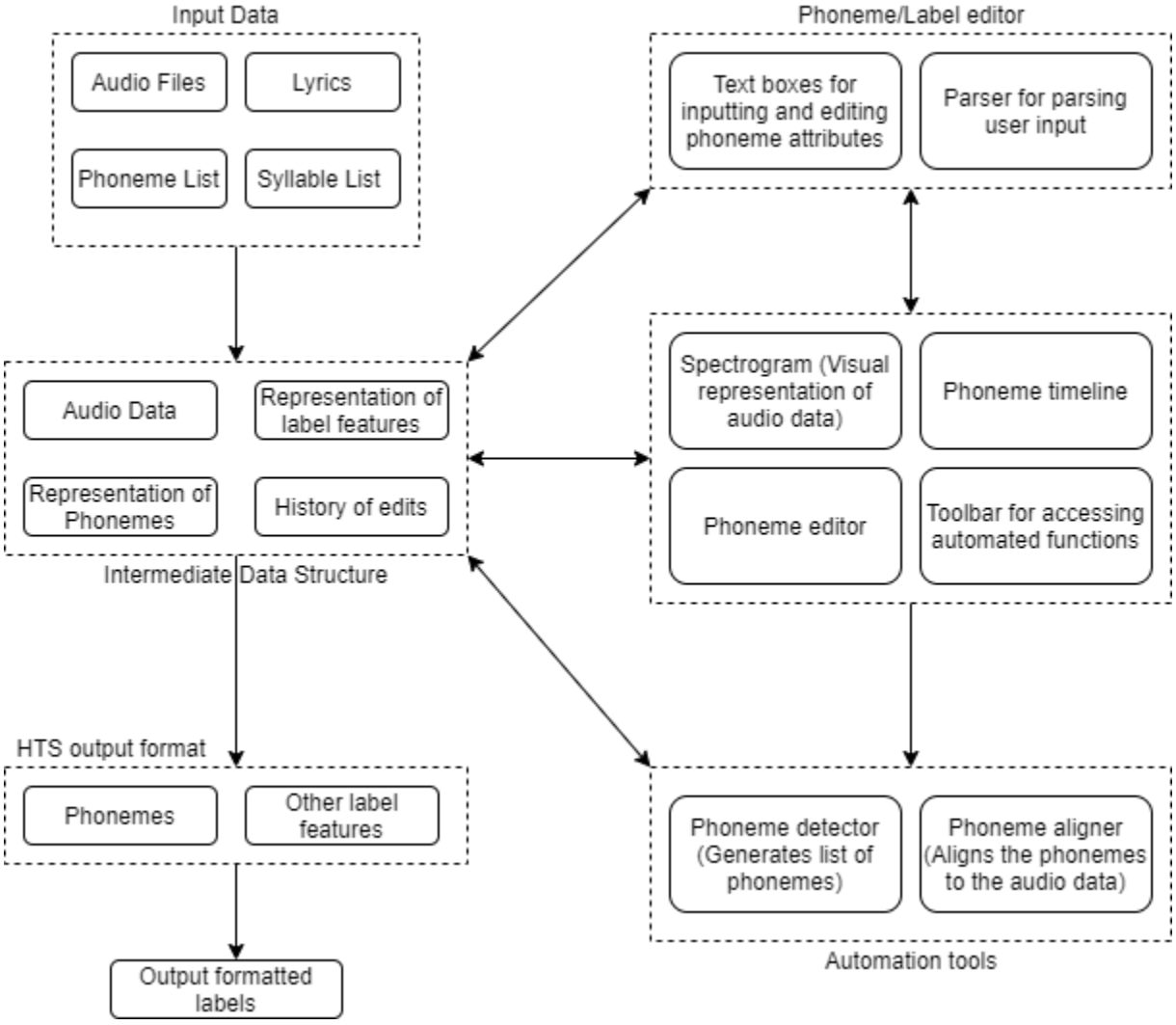
Novel Features:

- Allowing the user to copy and paste phonemes is a novel feature present in our tool. Other tools like praat don't let the user easily copy and paste phonemes or require plugins.
- Automatically detecting and aligning phonemes in the same tool is a novel feature. Other tools usually require multiple tools to achieve this.

Technical Challenges:

- Our target output format (HTS) is challenging to interpret by eye and contains a lot of specifications. Formatting our output data to this specification presents a challenge.
- Automating phoneme detection posed a challenge. We ended up splitting the task into phoneme identification and detecting where a phoneme starts and ends. To identify phonemes, we used a hierarchical classification model which converted the sound into MEL spectrograms and passed it through a series of convolutional neural network based classifiers.
- Increasing the speed of our spectrogram generation poses a challenge. Currently we use the Cooley-Tukey fast Fourier transform algorithm to compute the spectrogram. However, our implementation seems to be too slow for high sample rate audio.
- Smooth draggable UI elements pose another challenge due to the limitations of our GUI library.

System Architecture Diagram:



Evaluation:

We will measure the success of our project using the following metrics:

- Fluidity of the UI. We will survey our client about how easy to navigate our UI is. In addition we can count the number of clicks required to perform an action. For example, to export data, the user has to click once on the menu bar, then click export, then click the correct format.
- Speed. We will time how long every action takes, this includes generating the spectrogram, loading a song, switching song, and exporting data.
- Accuracy. We will compare the autogenerated phonemes with the output from other tools like praat.
- Reliability. We will automate stress testing by using a script to automatically add hundreds of songs at a time and perform actions like adding phonemes.

Project Summary:

Module/feature	Completion %	To do
Create Intermediate Data Structure	80%	Feature Tracks, Language definition
Create Spectrogram	95%	Find alternative FFT algorithm for increased speed
Create a graphical representation of the audio data	100%	
Phoneme classification	75%	Increase the number of categories in the hierarchical classification. Integrate model with the tool.
Phoneme detection	80%	Modify model to be more similar to a proven one
File IO	50%	Add the ability to save a project to a file. Add the ability to output to a label file.
Implement Zooming of the graphical elements	100%	
Implement keyboard shortcuts	10%	Add shortcuts for all phoneme operations (add, remove, duplicate, copy, paste, edit, etc…). Add shortcuts for import operations.Add shortcuts for export operations.

Milestone 4:

- Basic phoneme feature input
- Basic project file IO
- Exporting labels to an output file
- Build test song database
- Create label output
- Add better navigation

Milestone 5:

- Complete timeline view
- Integrate phoneme boundary network
- Start note view

- Dictionary system
- Create label output

Milestone 6:

- Implement, test, and demo which features/modules
    - File I/O
    - Phoneme labeling
    - Phoneme boundary detection
    - Output
    - Note Input
    - Lyric separation
- Test/demo of the entire system
- Evaluation results
- Create user/developer manual
- Create demo video

Milestone 4 Task Matrix:

Task	Avinash	Nandith	Carlos
Complete intermediate data structure	50%	50%	0%
Create timeline feature widget	20%	20%	60%
Create project file format and I/O	20%	60%	20%
Create label output	50%	50%	0%
Build test song database	100%	0%	0%
Add better navigation	0%	80%	20%

Milestone 4 Task Description:

- Complete intermediate data structure
    - This data structure needs to be able to store features, such as phonemes and syllables, in a manner to be easily displayed and manipulated. Information that needs to be stored includes feature position and content.
- Create timeline feature widget
    - This is a generic display widget that graphically displays the data in the intermediate data structure. This includes lines to indicate position and their contents. Those of which can be edited by dragging the lines and entering text.
- Create project file format and I/O
    - We want to allow the user to save and load their work as a project, as well as allow the user to export their project and move it to another computer if needed. To do so, we need to serialize the data present in the intermediate data structure, as well as any other relevant data such as user preferences. After serializing the data, we need to write the data to a series of files. We also plan to make the tool autosave every few minutes.
- Create label output
    - To generate output, we need to format all the label data according to the HTS singing label format specification.
- Build test song database
    - To test and train the neural networks, we need proper data. A small list of songs has been selected, now the lyrics and score need to be organized so that volunteers can record them. This data will then have their phonemes labeled.
- Add better navigation
    - There is currently no horizontal scroll bar. We will utilize the full audio view as a visual scroll bar.

Approval from Faculty Advisor

- "I have discussed with the team and approve this project plan. I will evaluate the progress and assign a grade for each of the three milestones."
- Signature: _____ Date: _____