CPSC 490 Summary
Read the Room

Jordan Smith
Advisor: Scott Peterson

April 2020

1 Abstract

This project aims to create a home-assistant system that guides the user in music genre selection based on the mood of the room and how many people occupy within. Mood is determined by an analysis of the soundscape of the room. More specifically, by two different major components: speaker diarization (the number of different unique speakers), and speech emotion analysis through a wave-form feature trained regression. I've created a program using a multi-class probabilistic SVM classifier that takes in a waveform file and analyzes it for key components. I've used existing emotional sound data to train the SVM to classify the emotional content of the soundscape and create a schema for determining the mood of a room as a function of valence and arousal levels. In combination with knowing how many people are in the room and the emotional affect of the words that are being said, I'm then able to make a statement about the general mood of the room. From there I've created a logical schema to match a genre of music to the space, and effectively "Read the Room." To pull everything together I've used Amazon Web Service to create a queue that both the Amazon Alexa skill I've created and my computer can communicate with. Having made this connection I can trigger the process of recording, analyzing, and genre determining all by asking Alexa to "Read the Room."

2 Introduction

In my summary I'll detail each of the different components in the order I implemented them: Speaker Diarization, Speech Emotion Analysis, Genre Determination, and Amazon Alexa Integration.

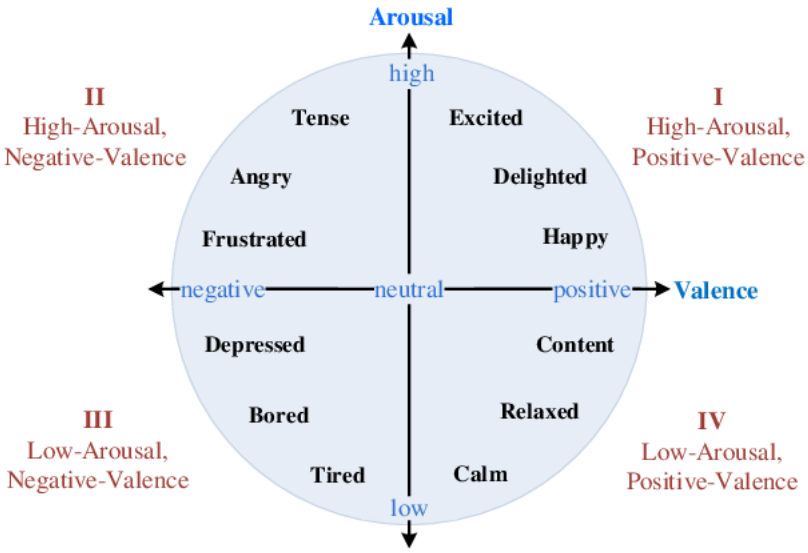
3 Speaker Diarization

I knew from the very beginning that determining how many people were gathered in a space together would be a key component in the success of my project.

To this end I took a lot of time researching the concept of speaker diarization, learning from different approaches and implementations to speaker diarization. I settled upon going in the direction of machine learning by clustering, specifically k-nearest neighbor clustering. To implement speaker diarization I used the open-source python package pyAudioAnalysis. I tested the speaker diarization function within audiosegementation.py and made some changes to fit my needs in terms of returning the number of people in a room. I found that the system worked best when the people where physically in the same room. Because of Covid-19 I experimented with recording zoom calls of varying size, but in the analysis of these recordings the algorithm was much less accurate. the maximum number of people I could credibly test was three; me, my mother, and my father. However I got consistent results in person.

4 Speech Emotion Analysis

When searching for python packages for support I found that the same library, pyAudioAnalysis, could help me with the speech emotion component of my project. The training algorithm relies on the python package sklearn to implement support-vector machine learning models. To train the regression pyAudioAnalysis contains a library of recordings used to give the regression the ability to classify an unknown recording as a function of valence and arousal values. Arousal (or intensity) is the level of autonomic activation that an event creates, and ranges from calm (or low) to excited (or high). Valence is the level of pleasantness that an event generates and both are defined along a continuum from [-1 to 1].(Bestelmeyer 2017)



My algorithm inputs a .wav file and returns a tuple containing a value between -1 and 1 for both valence and arousal respectively.

5 Genre Determination

To pick the right music genre for the room I used a series of logical gates taking the input of valence, arousal, and number of people to return which genre of music I determined was the best fit. The first logical gate is how many people there are, if there are very few people the algorithm is more likely to return a type of music one is more likely to listen to when alone, like classical or lofi chill beats. However, in my eyes if there are more than 6 people in a room they aren't very likely to want to hear something as isolating as classical music and they are more likely to listen to music that everyone can enjoy. This led me to return genres that lead people to dance more often when the number of people in the room was higher. After considering the number of people I move on to use the arousal and valence to help pinpoint what genre should be chosen. For High Arousal and High Valence, I return music people can sing and dance to. For High Arousal and Low Valence, I return rock music, which complements the high energy yet negative emotional content. For Low Arousal and High Valence I return chill beats and soul music to adhere to a calmer more content vibe. Finally, for Low Arousal and Low Valence I return a more depressed genre like blues or smooth jazz. I've explained the extremes, so for the values that are medium I follow the same general idea but use my own discretion for what I would play as the host. "Read the Room" effectively automates the steps I would take in choosing what music to play as the host of a gathering.

6 Amazon Alexa Integration

I researched multiple different ways to have Alexa run python, but I determined that my program was too complex to be run directly in the Amazon Cloud. I figured the best way to achieve my goal would be to have Alexa send a trigger to my computer and let all the heavy computation happen on my laptop.

6.1 AWS Queue

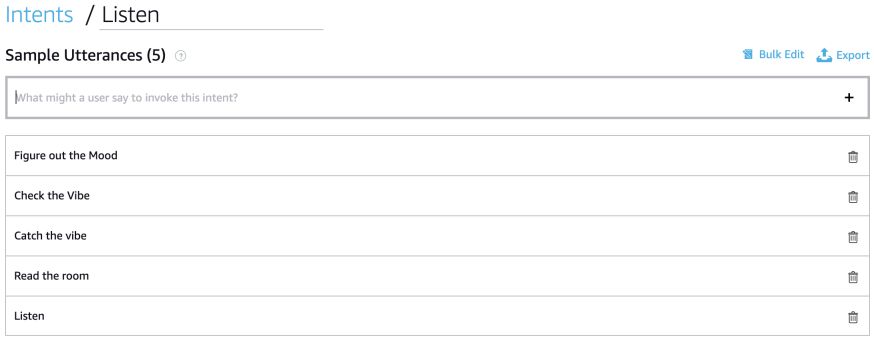
The easiest way to establish the connection between Alexa and my computer was through an Amazon Web Service queue. I created my own Amazon Web Service account and created a queue for Read the Room. Within AWS I followed their documentation in creating a queue for interaction with Alexa. The most technical part of this step was modifying a file called lambda_function.py. It handles listening to Alexa for what it should put onto the queue, while also listening to my computer for when to take something off the queue and prompt Alexa to speak again. I modified this function so that when Alexa says she is "listening" it puts the word "listening" on the queue. Meanwhile my primary read the room program is running checking for anything new on the queue; When my program sees "listening" on the queue it begins the process of recording the environment through my laptop microphone and analyzing that output.wav file to save the correct genre. When the user speaks to Alexa once more to ask

her what she read, Now the function puts the word "responding" on the queue which prompts my computer to put the aforementioned genre on top of the queue. The AWS function then modifies what Alexa says next to "I think you should play..[]" with the last word being the genre that was just pushed to the queue.

6.2 Alexa Skill

In order to have Alexa interact with the queue I had to create my own Alexa skill and specify the AWS queue as an endpoint. In order for Alexa to communicate with the queue she specifies an intent. Saying "Alexa Ask Read the Room to Listen" first invokes the Read the Room skill and then specifies the intent of listening which it communicates to lambda_function.py.

The Alexa Skills toolkit allows different phrases to signify intent, shown here.



There is a similar collection of phrases signaling the intent of responding, one of them being "Alexa ask Read the Room what's the vibe." Again, invoking the read the room skill and specifying intent to be communicated to the AWS queue.

A typical conversation would flow as follows:
User: Alexa ask Read the Room to Read the Room
Alexa: I'm Listening
Waits 30 seconds for recording and analysis to finish
User: Alexa ask Read the Room what's the vibe
Alexa: I think you should play some [*Insert Music Genre*]

At this point the user has the privilege of following Alexa's recommendation or not, while being able to specify what platform they prefer to play music on. I chose to have the user be in charge of playing the music as to not force a genre on the user that they don't necessarily want to hear even if that is what my program perceives is the right choice.

7 Closing Remarks

I decided to move away from the direction of speech-to-text natural language processing because I didn't think sentiment analysis would add much more on top of the speech emotion regression in terms of determining the mood of the room. If anything it would add an unnecessary layer of complexity.

If I could improve one thing about my project it would be the methodology behind choosing the right genre for the user. There are so many factors like the users baseline preference for a genre or if they've already been listening to a particular genre and are in the mood for something new. The program would never know these things without some preliminary preference selection mechanism. Moreover, I could add a reinforcement learning component to the genre selection, such that if a user was particularly satisfied or dissatisfied with the genre selection they could give the program the necessary feedback and tell it to remember that instance for the future.

7.1 Conclusion

All in all, I'm very happy with how my CPSC 490 project came to fruition. I achieved my goal of matching the right genre of music for a gathering of people and had Amazon Alexa successfully "Read the Room."

8 Reference

1. Bestelmeyer, Patricia E G et al. "Effects of emotional valence and arousal on the voice perception network." Social cognitive and affective neuroscience vol. 12,8 (2017): 1351-1358. doi:10.1093/scan/nsx059