Face the Music

Amelia Simonoff

DD/MM/2022

A thesis presented for the degree of

Bachelor of Arts in Music

At the

THE UNIVERSITY OF CHICAGO | COLLEGE

1 Abstract

Current facial recognition technology has progressed to the point that it can be done in nearly real-time. In this project, I use the Python library DeepFace to read peoples' emotions in real-time, and then the Spotify Web API and Web Playback SDK to play music related to those emotions.

I set up this project on a Raspberry Pi, continually running a Python script that reads data from the camera and then pulls data from Spotify. I then set up the Raspberry Pi in the Media Arts, Data, and Design (MADD) Center at the University of Chicago.

2 Background

Music and emotions have been indelibly linked throughout human history. From the first chants to pop songs, music has evoked emotions in its listeners since time immemorial. Here, I was interested in the other direction: what if we create music based on an emotion, instead of the other way around?

To answer that question, then, I needed a way to read the emotions of users in near real-time, and then a way to either synthesize music of a certain emotion or pull from an already-existing database. I knew that facial recognition software could accomplish the first step, and so I chose to use a pre-trained model.

Facial recognition software has been available for the past DECADE (citation), and its implementations in the modern day are not without controversy. However, much research has been done on facial recognition, and current implementations are incredibly lightweight, especially using pre-trained models.

Unfortunately, real-time music synthesis is not yet available. While artificial-intelligence based musical synthesis has already passed the Turing Test (citation), the models are not quite versatile enough yet to produce a continuous stream of music with changing input parameters (i.e., the emotion of the user). As such, I decided to use a corpus of music that is already labeled, which is found in many streaming services, such as Apple Music or Spotify.

Spotify, the most-used music streaming service in the world (citation), has many playlists with emotional labels, such as "Happy Beats" or "Sad Piano", etc.. However, Spotify does not label their music based on emotion outright, and instead classifies music by a large variety of parameters (see Methods).

CONCLUSION

3 Significance Statement

Music is cool facial recognition is dope something about fronto-temporal dementia

4 Methods

This project is running on a Python backend with a Node.js environment communicating to a React application to output the music to a browser. The general overview of the project is outlined in Figure 1 (INSERT FLOWCHART).

I use a Python environment running OpenCV to pull video data from the webcam. Each frame is passed to DeepFace, which is a wrapper for multiple state-of-the-art emotion recognition models. I rewrote the code for real-time analysis of incoming video frames.

Each frame is classified by the probability that it is one of the following emotions: `angry, disgusted, scared, happy, sad, surprised`, and `neutral` (citation). Then, the most likely emotion is chosen as the `dominant emotion`, which is then sent for downstream processing.

Each video frame is also rendered, and a user's emotion, if available, is displayed on the frame. Pulling and rendering video frames takes significantly less time than analyzing an emotion in a frame, so both processes (known as workers for multiprocessing) run asynchronously and communicate using a Queue datastructure.

Parsed emotions are also passed to a third worker, which communicates with Spotify through the Spotify Web API. Upon initialization, the worker authenticates the user using the Spotify OAuth service, and then pulls the user's top songs and artists. The worker then makes a list of the genres of the user's top songs, and compares it with the available genres for recommendation, which is used as a mask.

After initialization, the worker loads the parameters for the recommendation request for each emotion. Every ten seconds, the worker queries Spotify for a list of recommendations given a random number of user top sonsg, artists, and genres as seeds, as well as the parameters given by the current emotion. The worker then plays a randomly chosen song from the returned list of suggestions, starting between one-sixth and one-third of the way through the song.[1]

The tunable parameters that I chose for each emotion for the recommendations query from Spotify are as follows. Each parameter has a maximum, minimum, and target value (e.g. `min_acousticness`, `max_danceability`, `target_valence`): `acousticness`, `danceability`, `energy`, `instrumentalness`, `liveness`, `loudness`, `popularity`, `speechiness`, `tempo`, and `valence`. See the supplemental material for the tuned parameters for each emotion.

Concurrently, a Node.js process supporting a React application is running to display the Spotify Web Playback SDK in a browser. The backend once again authorizes the user, and the browser functions as a device for Spotify playback, which is used as the default playback destination for the Spotify worker.

5 Code Availability

The code is available on Github under the GNU General Public License, version 3.

6 Observations

hehe look at all those ~~monkeys~~ people controlling this with their face hehe

7 Discussion

8 Conclusion

1. I chose these values because they are close to the start of the song but are past any lead-ins or introns, which might not accurately represent the emotion codified by the song.

References

Justin Patrick N. and Sloboda John A. Handbook of music and emotion: theory, research, applications. London: Oxford University Press, 2010.

Testing testing 123.

Sundberg Johan. Music, Mind, and Brain: The Neuropsychology of Music. Edited by Manfred Clynes. Chap. VII: Speech, Song, and Emotions. New South Wales State Conservatorium of Music, 1983.

Emotions passed from the voice. Vocal emotions irrespective of culture and language of origin.

9 Supplemental Material

9.1 Tuned Parameters

```
# emotion_labels = ['angry', 'disgusted', 'scared', \
#     'happy', 'sad', 'surprised', 'neutral']

angry_params = {
    'min_acousticness': 0,
    'min_danceability': 0.2,
    'min_energy': 0.4,
    'min_instrumentalness': 0,
    'min_liveness': 0,
    'min_loudness': -60,
    'min_popularity': 0,
#     'min_speechiness': 0,
    'min_tempo': 80,
    'min_valence': 0,

    'max_acousticness': 0.8,
    'max_danceability': 0.65,
    'max_energy': 1,
    'max_instrumentalness': 1,
    'max_liveness': 0.4,
    'max_loudness': 0,
    'max_popularity': 100,
    'max_speechiness': 1,
#     'max_tempo': 120,
    'max_valence': 0.6,

    'target_acousticness': 0.3,
    'target_danceability': 0.4,
    'target_energy': 0.65,
    'target_instrumentalness': 0.5,
    'target_liveness': 0.5,
#     'target_loudness': -60,
#     'target_popularity': 0.5,
    'target_speechiness': 0.2,
    'target_tempo': 140,
    'target_valence': 0.2
}

disgusted_params = {
    'min_acousticness': 0,
    'min_danceability': 0.2,
    'min_energy': 0.4,
    'min_instrumentalness': 0,
    'min_liveness': 0,
    'min_loudness': -60,
    'min_popularity': 0,
    'min_speechiness': 0,
    'min_tempo': 0,
    'min_valence': 0,

    'max_acousticness': 1,
    'max_danceability': 0.6,
    'max_energy': 0.7,
    'max_instrumentalness': 1,
    'max_liveness': 0.4,
    'max_loudness': 0,
#       'max_popularity': 100,
    'max_speechiness': 1,
    'max_tempo': 120,
    'max_valence': 0.5,

#       'target_acousticness': 0.5,
    'target_danceability': 0.4,
    'target_energy': 0.55,
    'target_instrumentalness': 0.5,
    'target_liveness': 0.5,
    'target_loudness': -60,
#    'target_popularity': 0.5,
    'target_speechiness': 0.5,
#    'target_tempo': 0.5,
    'target_valence': 0.2
}

scared_params = {
    'min_acousticness': 0,
    'min_danceability': 0,
    'min_energy': 0,
    'min_instrumentalness': 0,
    'min_liveness': 0,
    'min_loudness': -60,
    'min_popularity': 0,
    'min_speechiness': 0,
    'min_tempo': 0,
    'min_valence': 0,

    'max_acousticness': 1,
    'max_danceability': 0.4,
    'max_energy': 0.5,
    'max_instrumentalness': 1,
    'max_liveness': 0.4,
#    'max_loudness': 0,
#       'max_popularity': 100,
    'max_speechiness': 0.5,
    'max_tempo': 120,
    'max_valence': 0.6,

    'target_acousticness': 0.5,
    'target_danceability': 0.2,
    'target_energy': 0.3,
    'target_instrumentalness': 0.5,
    'target_liveness': 0.5,
    'target_loudness': -60,
#       'target_popularity': 0.5,
    'target_speechiness': 0.5,
#       'target_tempo': 0.5,
    'target_valence': 0.35
}

happy_params = {
    'min_acousticness': 0,
    'min_danceability': 0.5,
    'min_energy': 0.5,
    'min_instrumentalness': 0,
    'min_liveness': 0,
    'min_loudness': -60,
#       'min_popularity': 0,
    'min_speechiness': 0,
    'min_tempo': 60,
    'min_valence': 0.4,

    'max_acousticness': 1,
    'max_danceability': 1,
    'max_energy': 1,
    'max_instrumentalness': 1,
    'max_liveness': 0.4,
    'max_loudness': 0,
    'max_popularity': 100,
    'max_speechiness': 1,
    'max_tempo': 120,
    'max_valence': 1,

#     'target_acousticness': 0.5,
    'target_danceability': 0.8,
    'target_energy': 0.8,
#     'target_instrumentalness': 0.5,
    'target_liveness': 0.3,
#     'target_loudness': -60,
#     'target_popularity': 0.5,
#     'target_speechiness': 0.5,
#     'target_tempo': 0.5,
    'target_valence': 0.8
}
```

```
sad_params = {
    'min_acousticness': 0,
    'min_danceability': 0,
    'min_energy': 0,
    'min_instrumentalness': 0,
    'min_liveness': 0,
    'min_loudness': -60,
    'min_popularity': 0,
#     'min_speechiness': 0,
    'min_tempo': 0,
    'min_valence': 0,

    'max_acousticness': 1,
    'max_danceability': 0.4,
    'max_energy': 0.5,
    'max_instrumentalness': 1,
    'max_liveness': 0.4,
    'max_loudness': 0,
#     'max_popularity': 100,
    'max_speechiness': 1,
    'max_tempo': 120,
    'max_valence': 0.4,

    'target_acousticness': 0.6,
    'target_danceability': 0.2,
    'target_energy': 0.2,
    'target_instrumentalness': 0.8,
    'target_liveness': 0.3,
#       'target_loudness': -60,
#       'target_popularity': 0.5,
    'target_speechiness': 0.5,
    'target_tempo': 80,
    'target_valence': 0.2
}

surprised_params = {
    'min_acousticness': 0.3,
    'min_danceability': 0.4,
    'min_energy': 0.3,
#       'min_instrumentalness': 0,
    'min_liveness': 0,
    'min_loudness': -60,
    'min_popularity': 0,
    'min_speechiness': 0,
    'min_tempo': 0,
    'min_valence': 0.2,

    'max_acousticness': 1,
    'max_danceability': 0.6,
    'max_energy': 0.8,
    'max_instrumentalness': 1,
    'max_liveness': 0.4,
    'max_loudness': 0,
    'max_popularity': 100,
    'max_speechiness': 1,
#       'max_tempo': 120,
    'max_valence': 1,

    'target_acousticness': 0.5,
    'target_danceability': 0.5,
    'target_energy': 0.5,
    'target_instrumentalness': 0.5,
    'target_liveness': 0.5,
    'target_loudness': -60,
#       'target_popularity': 0.5,
    'target_speechiness': 0.5,
    'target_tempo': 108,
    'target_valence': 0.5
}

neutral_params = {
    'min_acousticness': 0,
    'min_danceability': 0,
    'min_energy': 0.3,
    'min_instrumentalness': 0,
    'min_liveness': 0,
    'min_loudness': -60,
    'min_popularity': 0,
    'min_speechiness': 0,
    'min_tempo': 0,
    'min_valence': 0.3,

    'max_acousticness': 0.8,
    'max_danceability': 0.5,
    'max_energy': 0.7,
    'max_instrumentalness': 1,
    'max_liveness': 0.2,
    'max_loudness': 0,
    'max_popularity': 100,
    'max_speechiness': 0.6,
    'max_tempo': 140,
    'max_valence': 0.7,

    'target_acousticness': 0.5,
    'target_danceability': 0.4,
    'target_energy': 0.5,
    'target_instrumentalness': 0.5,
    'target_liveness': 0.5,
    'target_loudness': -60,
#     'target_popularity': 0.5,
    'target_speechiness': 0.5,
#     'target_tempo': 0.5,
    'target_valence': 0.45
}
```