Group Members

- Moiz Zulfiqar – 24543
- Ammar Lokhandwala - 25245

Song Recognizer Program in Python

Objective: The aim of this project is to develop a music recognition program that can identify songs from a database based on a short 30-second audio clip. The program is built using Python and uses Markov models to compare audio features of the clip to features extracted from songs in the database.

Methodology: The methodology followed for this project consisted of six main steps:

1. Build a database of songs: The first step involved building a database of songs. We extracted relevant audio features such as mel-frequency cepstral coefficients (MFCCs), spectral centroid, spectral contrast, spectral rolloff, and tempo from a set of 10 popular songs using the Librosa library in Python. Each song's audio features were used to create a Markov model, which was saved for future comparisons.
2. Preprocessing: We normalized the input that was a 30-second audio clip to ensure a consistent format for feature extraction. This was done by resampling the audio data and taking the mean if the audio data had more than one channel.
3. Feature extraction: We extracted the same audio features as the database songs from the 30-second audio clip using the Librosa library.
4. Model building: A Markov model was built for each song in the database using a method called Markov Chain Model. The Markov model was designed to represent the sequence of audio features in each song, capturing the temporal dependencies between consecutive features. Each state in the model corresponds to a distinct set of audio features and transitions between states represent the change of these features over time. We chose a Markov model of order 3, meaning the next state depends on the current state and the two states preceding it.
5. Song recognition: The extracted audio features from the inputted song were then compared to the Markov model for each song in the database. For this, we used the Viterbi algorithm, a dynamic programming algorithm that finds the most likely sequence of states in a Markov model given a sequence of observations. The song whose Markov model gave the highest likelihood was selected as the recognized song.
6. User interface: We developed a simple user interface for the program using Tkinter that allows users to input a 30-second audio clip and see the recognized song.

Results/Evaluation: The performance of the music recognition program was evaluated using a set of test audio clips. The program was able to accurately identify songs from the database based on short audio clips. The computational efficiency of the program was also assessed, and it was found to be reasonably efficient.

Limitations:

- The current implementation is limited to a small database of 10 songs, which may not be representative of real-world usage.

- Currently, the program requires users to provide a 30-second audio clip, which may not be practical in real-life scenarios where users may not have the patience to wait for 30 seconds
- The program's accuracy depends on the quality of the input audio clip and may not perform well for noisy or distorted clips.
- The algorithm's performance may be affected by the choice of audio features and the type of Markov model used.
- Currently, the program accurately finds the inputted song but when a song which is not in the database is given, it identifies it as a song in the database rather than indicating that the song is not found. This is a limitation that stems from the probabilistic nature of the Markov model and the threshold set for recognition. Increasing the threshold did result in more accurate recognition but it also lead to more instances of song not found, even when the song was in the database.
- Processing songs takes a significant amount of time, which may not be efficient for larger databases. This is due to the time taken to extract features and build the Markov models. However, it should be noted that the model building step is a one-time process and results can be saved for future use.

Future work:

1. Expanding the song database: Our initial implementation focused on a small database of 10 songs for proof of concept. As a next step, we plan to expand the database to include a wider variety of songs. This will improve the program's utility and make it more applicable to real-world usage.
2. Develop an algorithm that can accurately recognize songs based on shorter audio inputs, making the program more efficient and practical for real-life usage.
3. Improving computational efficiency: We intend to enhance the computational efficiency of the program by exploring other feature extraction methods or optimizing the existing one. In addition, we plan to use faster algorithms or data structures, and parallel processing to speed up the Markov model building process.
4. Enhancing the recognition algorithm: We are considering adjusting the threshold for better recognition or using a different algorithm that works with the current algorithm to recognize a song that is not in the database.
5. User interface enhancements: The current user interface is quite basic and can be improved by adding features such as a progress bar during feature extraction and model building, and options to add or remove songs from the database.
6. Incorporating machine learning algorithms: In the longer term, we may consider incorporating machine learning algorithms that can learn to recognize songs based on the extracted features. This could potentially improve the accuracy and efficiency of the song recognition program.

Conclusion: Despite its limitations, our song recognizer program shows promising results. It demonstrates the feasibility of using Markov models for song recognition and provides a foundation for further improvements. We believe that with future enhancements, this program can be a useful tool for song recognition.