Implementation of Music Chroma Extraction to Calculate Similarity Scores between Japanese Pop and Rock Artists

William Andrian Dharma T 13523006[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523006@std.stei.itb.ac.id, [2]williamadt123@gmail.com

***Abstract*—Music Information Retrieval is usually used to determine similar songs for a recommendation system. But another possible use of it is as a comparison metric between artists to evaluate how similar artists are to one another, that could also be of use to a recommendation system. This approach uses chroma extraction as the feature to compare upon and generate a similarity graph between artists, specifically Japanese pop and rock artists for this paper.**

***Keywords*—Audio Similarity, Chroma Extraction, Music Information Retrieval, Music Recommendation**

I. Introduction

Music Information Retrieval is the technique of extracting information from music as a feature vector comprised of one or more features extracted from a piece of music. The features extracted can come in many forms such as tempo, pitch, spectrogram, chroma, etc. This paper will mostly emphasize chroma extraction.

Chroma is a representation of audio which bins the whole spectrogram of an audio onto 12 bins, each representing the 12 semitones of an octave. Unlike a spectrogram which describes the whole frequency distribution of an audio, a chroma representation simplifies the spectrum without caring for its absolute frequency and bins it directly to its corresponding semitone. This simplification reduces the amount of noise in the extracted data for analysis and reduces the complexity for processing and comparisons. Multiple chroma representations exist such as Chroma-Pitch (CP), Chroma-Log-Pitch (CLP), Chroma Energy Normalized Statistics (CENS), etc. The one used in this paper is CENS.

The comparison will be done using cosine similarity. Cosine similarity is a metric to determine the similarity of two or more objects which ignores their magnitudes. Cosine similarity is chosen over Euclidean distance here because the similarity of audio shouldn't be affected by magnitudes which in this case would mean audio length.

Analysis done in this paper uses the programming language Python and the library Librosa as the main audio analysis tool.

II. Preliminaries

A. Chroma

The main feature used in this method is chroma. Chroma is extracted by binning the spectrogram of an audio of a certain window into the 12 semitones of an octave. The result is a 12-dimension vector with the intensity of each semitone as its values. The chroma feature matrix of an audio takes the shape of the number of frames times the 12-dimension chroma vector of a frame.

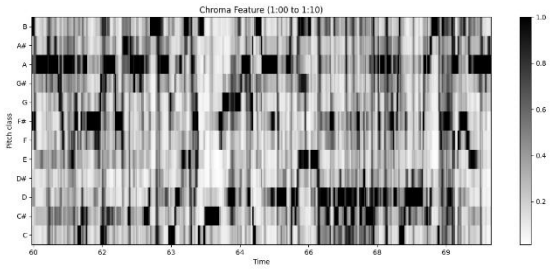

Fig. 1. Chromagram of the song "Haru" by Yorushika

```
Chroma matrix shape: (12, 11659)
One chroma column:
[0.21280679 0.04047984 0.31101635 0.09381019 0.02053199 0.00861399
 0.00468669 0.00805302 0.09769258 0.47278354 0.6230452  1.        ]
```

Fig. 2. Shape of an extracted chroma matrix

Chroma is the method used here because unlike Mel-Frequency Cepstral Coefficients or MFCCs [1], which captures the timbre and instrumentations of an audio, the chroma method is instrumentation invariant and tries to capture the melodic content instead [2]. That means that different versions or covers of a song would be recognized as its original song, even when the arrangement or instrumentation is different, which would be particularly useful on a query by humming feature.

B. Chroma Energy Normalized Statistics

The Chroma Energy Normalized Statistics (CENS) version of chroma is made by normalizing each chroma vector with respect to the $\ell 1$ -norm and expresses relative energy distribution. Next, a quantization of amplitude with

a logarithmic style approach is applied. Finally, the result is smoothened with a sliding window. CENS has been found to be a more robust version of chroma and is useful for audio matching applications [3].

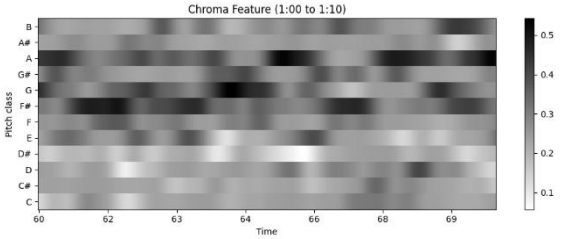

Fig. 3. CENS chromagram of the song "Haru" by Yorushika

C. Cosine Similarity

The cosine similarity between two vectors A and B is calculated with the formula below:

$$Cosine\ Similarity = \frac{\boldsymbol{A} \cdot \boldsymbol{B}}{\|\boldsymbol{A}\|\|\boldsymbol{B}\|} \tag{1}$$

The formula calculates the cosine of the angle between two vectors with its value being interpreted as -1 being an exact opposite, 0 as no similarity, and 1 as the exact same.

Since cosine similarity measures the angle between two vectors and not its absolute distance, it is well suited for comparisons where the actual magnitudes of the vectors do not necessarily correlate to the vector's similarity, which fits the case of audio similarity, where we might need to identify a short snippet of an audio and find the most similar match from a longer audio file [4].

D. Similarity Graph

A similarity graph is a type of node-link diagram where the edge weights denote the distance nodes are to one other. The more similar two nodes are, the closer they are in the graph. A Minimum Spanning Tree (MST) can be made using Prim's algorithm, Kruskal's algorithm, or other algorithms to display the closest neighbor to every node and could also show clusters in the data. Although we use cosine similarity which doesn't equate to distance, we can still create a similarity graph by transforming the scores.

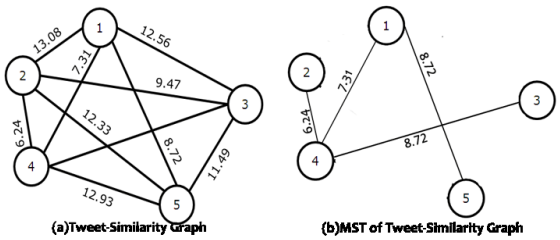

Fig. 4. Example of a similarity graph [4]

III. METHODOLOGY

A. Experiment Dataset

This paper specifies Japanese pop and rock artists as the comparison subject to localize a specific genre and see how similar or separate artists could be even in a relatively small scope. Expanding to multiple regions or genres would require additional decision making on whether an artist should be compared to everyone which would impact the performance metrics of the operation.

Before we move on to the dataset, to make things consistent, all the artists' names and songs will be in the Romanized form of the original Japanese name or in English in the case of katakana written loan words if the original name is not written with Latin characters.

Now for the dataset, 5 Japanese artists with their 5 most popular songs each are chosen as the comparison subjects. The most popular songs are taken because an artist's perceived style by the public would be taken from their most popular songs.

B. Tools Used

The programming language used in this experiment will be python, and we will be making use of several libraries which will help us extract chroma, compare and calculate similarity scores, and visualize the results of this experiment.

The main library used in this paper is Librosa. Librosa is a python library which specializes in audio information retrieval and audio processing. Librosa allows us to extract information from the audio track such as tempo, key, and pitch. We will make use of the chroma_cens function to extract the chroma from the audio. The chroma extracted from the audio can be visualized with a graph with the help of the matplotlib library and could be further processed to calculate the similarity score.

C. Data Preprocessing

Since this is a method to calculate the similarity scores between artists and not a music matching query, we need to make sure that all the songs are normalized by transposing every song to the key of C major. The transposition. This removes the problem of similar songs by style being calculated as dissimilar because of having different key signatures.

D. Similarity Calculation

We will calculate similarity scores between artists using the following approach:

1. Extract the chroma from every song which has been transposed and save it into a pickle file to reduce calculations.
2. Choose two artists to calculate the similarity score between them.
3. Take the first song of the first artist and calculate the cosine similarity values with every song from the second artist and store them.
4. Repeat step three for every song from the first artist.
5. Calculate the average similarity score of the previously stored values and that is our artist similarity score.
6. Repeat step 2-5 for every artist

The result would be 24 similarity scores between the artists which is the result of this experiment. Those similarity scores would then be used in creating a similarity graph to illustrate our findings.

Artist	Song 1	Song 2	Song 3	Song 4	Song 5
Back number	Suiheisen	Takane no Hanako san	Hanataba	Christmas Song	Happy End
Fujii Kaze	Kirari	Shinunoga E-Wa	Matsuri	Hana	Michi Teyu Ku
Higedandism	Pretender	Cry Baby	115 million Kilometer Film	I Love…	Subtitle
Yoasobi	Idol	Kaibutsu	Yoru ni kakeru	Gunjou	Tabun
Yorushika	Tada kimi ni hare	Dakara boku wa ongaku wo yameta	Hana ni bourei	Itte	Haru

Table 1. Artists and songs used in the comparison

IV. EXPERIMENT

A. Preprocessing

Firstly, every song listed above is transposed to C major using the Digital Audio Workspace (DAW) FL Studio's factory sampler plugin with the stretch mode, which transposes the audio without changing the length. The original key and final transposed key is check using the website tunebat.com .

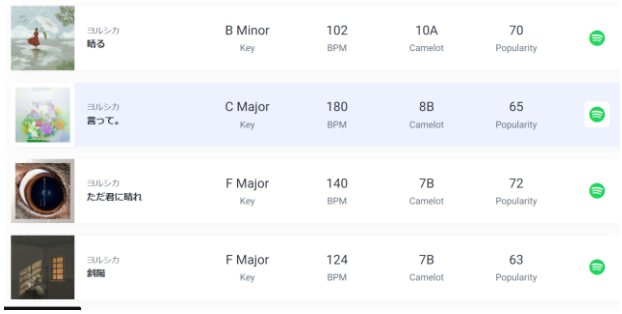

Fig. 5. Display of tunebat.com from a query

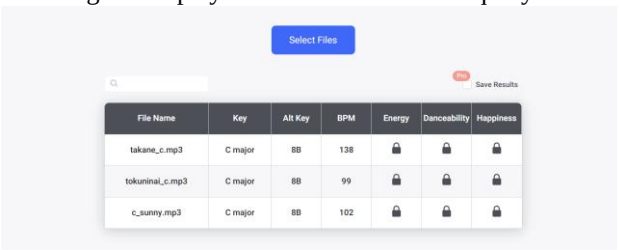

Fig. 6. Display of tunebat.com from a song key check

B. Chroma Feature Extraction and Caching

To save computational time in recalculating chroma from each song, every song's chroma is extracted and saved into a pickle file to be loaded in a similarity calculation. Chroma is extracted with the chroma_cens function which returns the CENS version of chroma. We then store the mean of all the frames of the chroma as the summary we store.

```
def cache_audio_features(audio_dir, cache):
    """Cache audio features for all files in the audio directory."""
    audio_files = [os.path.join(audio_dir, f) for f in os.listdir(audio_dir) if f.lower().endswith(('.wav', '.mp3'))]
    print(f"Caching features for {len(audio_files)} audio files...")
    for file in audio_files:
        y, sr = librosa.load(file, sr=None)

        chroma_cens = librosa.feature.chroma_cens(y=y, sr=sr)
        chroma_summary = np.mean(chroma_cens, axis=1)

        cache[file] = chroma_summary
    save_cache(cache)
```

Fig. 7. Chroma extraction and caching code

The chroma features are extracted to a dictionary which has the key name corresponding to the file name and is saved into a .pkl file. This file is where the chroma features will be loaded from.

C. Audio Similarity Scoring

Two audio files are selected and their chroma summaries are loaded from the cache file. Then, the similarity score is calculated by finding the cosine similarity between the two features.

```
def compute_audio_features(file_path, cache):
    """Compute Chroma CENS features for a given audio file, with caching."""
    if file_path in cache:
        print(f"Loading cached features for {file_path}")
        return cache[file_path]
    else:
        print("features not found in cache")

def compute_similarity(feature_1, feature_2):
    """Compute cosine similarity between two feature sets."""
    similarity = cosine_similarity(feature_1.reshape(1, -1), feature_2.reshape(1, -1))
    return similarity[0, 0]
```

Fig. 8. Audio similarity score code

D. Artist Similarity Scoring

The chosen two artists get their similarity score by comparing the similarity score between all their songs and getting the mean value. In essence, an array containing song names are created for each artist and the similarity score function is iterated between them.

```
feature_cache = load_cache()

yorushika = ['c_tadakimi.mp3','c_dakaraongaku.mp3','c_hananibourei.mp3','c_itte.mp3','c_sunny.mp3']
yoasobi = ['c_idol.mp3','c_kaibutsu.mp3','c_yorunikakeru.mp3','c_gunjou.mp3','c_tabun.mp3']

scores = []

for a in yorushika:
    for b in yoasobi:
        feature_a = compute_audio_features(os.path.join(AUDIO_PATH, a), feature_cache)
        feature_b = compute_audio_features(os.path.join(AUDIO_PATH, b), feature_cache)
        similarity_score = compute_similarity(feature_a, feature_b)
        scores.append(similarity_score)

print(f"Similarity score between yorushika and yoasobi: {np.mean(scores):.10f}")
```

Fig. 9. Code to calculate the similarity score between artists

```
❯ python similarity.py
Similarity score between yorushika and yoasobi: 0.9636424780
```

Fig. 10. Resulting similarity score

E. Similarity Graph

The similarity score still takes the form of a cosine similarity score and needs to be processed to be usable as a distance metric for the graph. First, the resulting scores are subtracted from the maximum possible value of cosine similarity which is one. This is because cosine similarity is inversely proportional to distance, therefore by subtracting it from one, we get a result which is directly proportional to distance and usable as a distance for the graph.

V. EXPERIMENT RESULTS AND DISCUSSION

Artist	Back number	Fujii Kaze	Higedandism	Yoasobi	Yorushika
Back number	-	0.9696530700	0.9767481089	0.9588846564	0.9723132849
Fujii Kaze	0.9696530700	-	0.9720816612	0.9723719954	0.9725326300
Higedandism	0.9767481089	0.9720816612	-	0.9640628099	0.9727508426
Yoasobi	0.9588846564	0.9723719954	0.9640628099	-	0.9636424780
Yorushika	0.9723132849	0.9725326300	0.9727508426	0.9636424780	-

Tabel 2. Original cosine similarity table between artists

Artist	Back number	Fujii Kaze	Higedandism	Yoasobi	Yorushika
Back number	-	0.0303469300	0.0232518911	0.0411154032	0.0276867151
Fujii Kaze	0.0303469300	-	0.0279183388	0.0276279449	0.0274673700
Higedandism	0.0232518911	0.0279183388	-	0.0359371901	0.0272491574
Yoasobi	0.0411154032	0.0276279449	0.0359371901	-	0.0363575220
Yorushika	0.0276867151	0.0274673700	0.0272491574	0.0363575220	-

Tabel 3. Subtracted cosine similarity table between artists

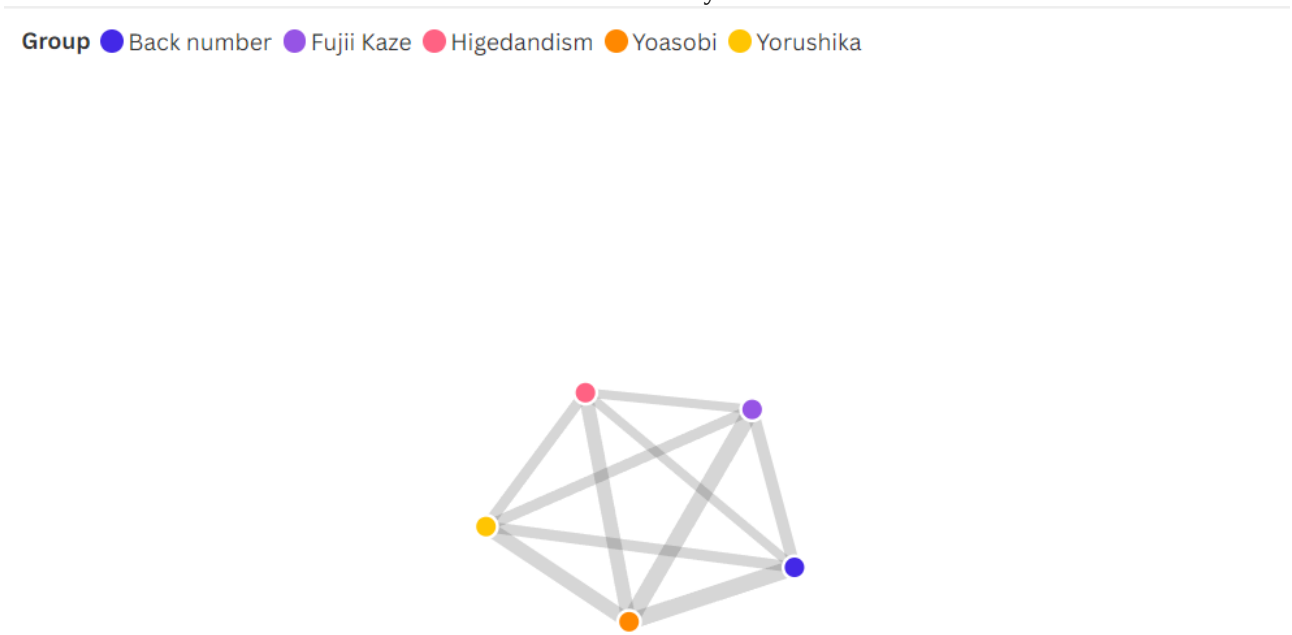

Fig. 11. Similarity graph between the artists

VI. APPENDIX

Appendixes, if needed, appear before the.

VII. ACKNOWLEDGMENT

I would like to give thanks to my family who supported me along my studies and my lecturer Dr. Ir. Rinaldi Munir, M.T.

REFERENCES

[1] Jean-Julien Aucouturier and Francois Pachet. “Music similarity measures: What’s the use?” in *Proc. 3rd International Symposium on Music Information Retrieval ISMIR*, Paris, 2002

[2] D. Ellis, “Classifying Music Audio with Timbral and Chroma Features,” in *Proc. ISMIR-07*, pp. 339-340, Vienna, Austria, October 2007.

[3] Meinard Müller, Frank Kurth, and Michael Clausen. Audio matching via chroma based statistical features. In Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR), pages 288–295, 2005.

[4] Oduntan, Odunayo & Adeyanju, Ibrahim & A.s, Falohun & Obe, Olumide. (2018). A Comparative Analysis of Euclidean Distance and Cosine Similarity Measure for Automated Essay-Type Grading. Journal of Engineering and Applied Sciences. 13. 4198-4204. 10.3923/jeasci.2018.4198.4204.
[5] Community Detection Based Tweet Summarization - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/aExample-of-Tweet-Similarity-Graph-for-the-sample-dataset-shown-in-Table-11_fig1_323546685 [accessed 31 Dec 2024]

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 27 Desember 2024

ttd

William Andrian Dharma T - 13523006