Automatic Music Audio Summary Generation

Elias Aouad
CentraleSupélec
elias.aouad@student.ecp.fr

Abstract

Music summarization is a recent topic of interest in the field of entertainment. The idea is to be able to generate a chunk of audio that represents best the music that we want to summarize. Nowadays, most methods in music summarization use a featurization of the audio sample, using for instance MFCCs, and work on those features to find similarities between audio segments. In fact, the most used methods rely on computing a similarity matrix between all features of all frames, and then analysing this matrix to output a reasonable chuck of the said song as a good summary. While hose methods work well in general, Peeters et al. [3] decided to go another way : make the hypothesis that feature vectors are the realisation of hidden states, predict those states and try to induce patterns given those states. In this paper, we will reproduce the pipeline and try it on a song ("All you need is love" by the Beatles), and see what kind of summary it outputs.

1. Introduction

Mobile phone development and innovation driven by Tech companies have strongly impacted people's access to music and the relationship they entertain with it. From initially the need to be at home with a cassette or at a concert, the digitalization of access to music through apps and streaming platforms creates new use cases for music, increasing therefore the weekly number of hours spent listening to music. Indeed, the average US citizen spends on average 26.9 hours per week listening to music in 2019 according to Statista. This shows the importance of music to people and the strong connection they have with music sources.

A question related to how a human brain interprets music segments arises. What patterns are relevant to the perception of music? Music is in fact like every other sound from a physics perspective: a signal composed of certain frequencies our ears could perceive and send to the brain for interpretation and decoding of the logical chronological sequence it receives.

However, what differentiates music from any other random noise is the set of notes that are being played, as well as the harmony in the played music that is gentler to the ear. It's not just about the notes that are being played, or the instruments that are being used, but rather a combination of all these parameters that come together to create one unique symphony. Cords, tone, frequencies, those are all parameters that enter in the complex process of generating a music.

Music is therefore a complex phenomenon. Understanding its components and patterns might open the door to many applications. In our problem, we are trying to make use of those patterns to generate a coherent summary of a music. It would be presented as an excerpt of the music that should summarize well the content in the full-length audio. Multiple studies have already been performed on this topic, and they usually transform the audio into a sequence of states that would be easy to manipulate to extract patterns.

In this paper, we will mostly focus on one of these studies, namely the one performed by Peeters et al. [3], and will reproduce their methodology. It mostly performs clustering on audio features to extract states from the music and interpret th emusic as a succession of these states. Their method was applied to the song "Head over Feet" from artist Alanis Morisette. We will however perform our experiments on a different song, to see how well the method generalizes on other songs. As it happens, we went for "All you need is love" from the Beatles.

2. Related Work

2.1. "Sequences" approach

Most of those approaches are related to Foote's work on similarity matrices [1]. The signal's features used are the MFCC, and they use a similarity measure to compare MFCCs of different time steps. This similarity could be the Euclidean distance, cosine similarity, or Kullback-Leibler

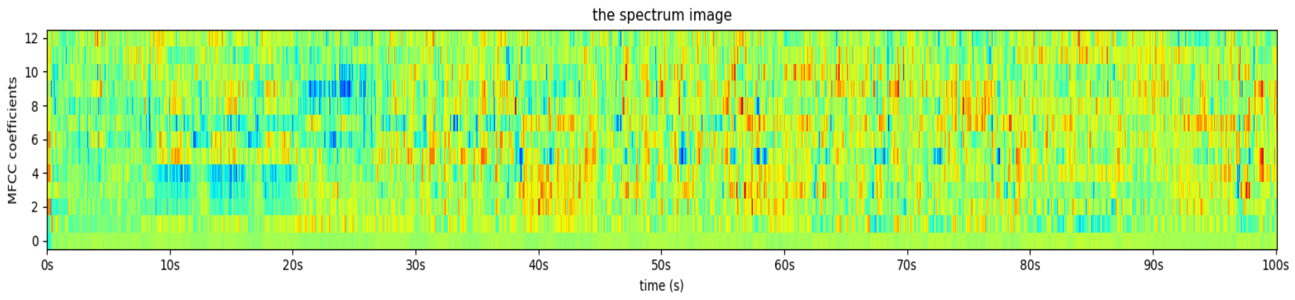

Figure 1: Spectrum image of ”All you need is love”

distance, ... In any cases, this approach produces a similarity matrix which allows a Visualization of the similarities in the music. If two segments in the song are similar, like for a tune that is repeated in the music, it appears directly on the matrix as the MFCCs of both segments end up with high similarity score. The focus is then to study this similarity matrix to extract repeated segments and output one of them as a summaru of the music.

2.2. "States" approach

In the ”states” approach, the focus is more on extracting hidden states which generate the music. It is done in several ways. One way is to divide the song into fixed length segments, which are then grouped according to a cross-entropy measure [2]. Grouped segments will then represented a particular state from which they were generated. Another way to do it and suggested in [2] is to make use of audio features to extract the hidden states. They used MFCCs and then train a hidden markov model on them. The model is then used to predict the sequence of states on the MFCCs and then construct a summary based on the outputted states (most represented state might capture a good summary of the song for instance).

3. Methodology

3.1. Choice of the song

First step is choosing the song we will use for our experimentation. We must choose a song that has somewhat repeated tones on which we could actually find interesting patterns. Not all songs can apply to this condition. For example, ”Bohemian Rhapsody” by Queen has no patterns in it, making it a difficult song to summarize. In our pipeline, we used the song ”All you need is love” by the Beatles. It has lots of repeated tones in it, and is easy to verify to see if the method works on it or not. For simplicity and faster computation power, we only consider the 100 first seconds of the song, but it still covers nearly half of the song.

3.2. Feature extraction & segmentation

For our study, we will use MFCCs as audio features for our song. Since we consider only the 100 first seconds of the song, this amounts to $9,999$ MFCC feature vectors (figure 1). Second, we want to find a way to segment the audio provided with those features. We will use the same idea as cited in the ”sequence” approach. We pass through the audio frame by frame, and compare each frame feature vector to the precedent one. If the cosine similarity score is higher than 0.8, then we are still in the same segment, otherwise, the new frame is in a new segment. This method is used to evaluate the proximity similarity between feature vectors, it does not however merge two vectors that are temporally far away from each other into a same segment. The idea is to have continuous segments, and he continuity should depend on time.

3.3. Merging segments

The precedent segmentation results in a big number of segments, most of which are very similar to each other. Also, the states given by the precedent segmentation are all distinct from one another, which is not what we want, since the interesting part is to understand which states are revisited along the time. In the end, we would want to have a reduced number of states, especially that the segments are seen as states, and we want to merge the segments that seem to belong to the same state.

To do so, we define a mean vector per segment, which is nothing more than the average vector of all feature vectors of a given segment. We then compare those mean vectors with the cosine similarity metric. Any two mean vectors who have a similarity score higher than 0.8 will see their segments merged into the same state.

3.4. K-means clustering

The result of the last step is to provide an estimation of the number of states and the affiliation of each feature vector to one of those states. Each state is provided with an average vector, which is unique per state.

Now, we know that this segmentation is not perfect. The goal is to learn a segmentation in an unsupervised way, preferably using K-means algorithm. However, K-means requires a prior information on the number of classes and a good initialisation for the centroids in order to output good results. For that, we can use the number of states outputted by our early work, and the average vector per state, again provided by our early work.

We can now run K-means, which uses euclidean distance to cluster data, and gather the predicted labels per frame. It provides our estimation of the current states of each frame.

3.5. HMM learning & prediction states

The ultimate goal of our study is to train a hidden Markov model (HMM) on our data. However, in the beginning of our study, we did not have any information on the labels of the feature vector nor did we have prior information on the distribution of the data. But now we are provided with the clustering of the K-means algorithm.

After our preliminary exploration with segmentation and K-means, we can assume that we have a good initialization for the training process of the HMM. Once we train the HMM on the provided data, we predict the label/state of each feature vector, and here is our final attribution of states for each frame.

3.6. Summary generation

Having trained the HMM, we now have an idea of what states generates which feature vectors (figure 3), and it looks something like this : $A \rightarrow A \rightarrow A \rightarrow B \rightarrow B \rightarrow C \rightarrow C \rightarrow A \rightarrow A$, etc... Now we just have to think of a way to generate a good summary of the music.

We came up with 2 different methods for generating a summary for the music :

- choose the state with the most repetition in the sequence
- choose the n-gram of state transition with the most repetition in the sequence

For the first type of summary, we take the longest streak of the state occurrence in the original signal and output it as a good summary of the song. For the second type of summary, we take the longest streak of the n-gram state transition occurrence in the original signal and output it as a good summary of the song.

4. Results

4.1. Similarity matrix

When we first computed the MFCC, we wanted to have a quick look at the similarities across the audio signal. Just like in the "sequences" approach, we are going to plot the similarity matrix of all MFCC features, represented in figure 2. Visualization should be easy on the similarity matrix, as similar segments will be represented by upper (lower) diagonals on the matrix.

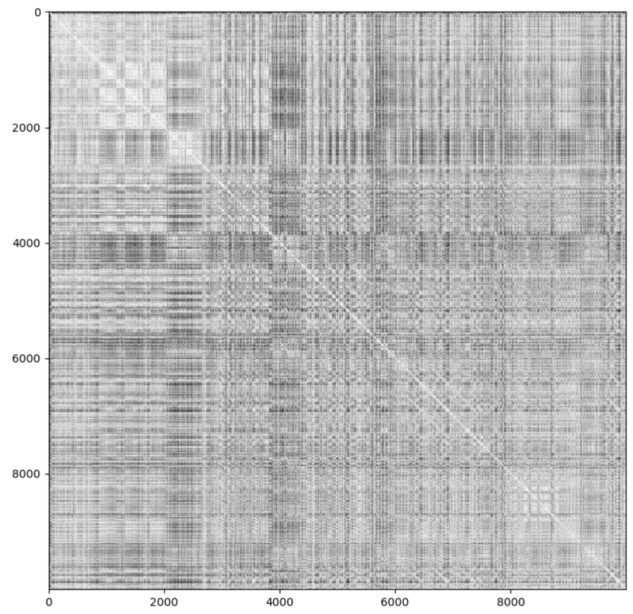

Figure 2: Similarity matrix of all features

In our plotted matrix, it does not look like there are lots of upper (lower) diagonals. However, if we focus a little, we would see a lower and an upper diagonal around points of axis 5000, and also around points of axis 7000. Since we took only 100 seconds of audio, it corresponds to the moments around $t = 50s$ and $t = 70s = 1min\ 10s$. At $t = 50s$, the Beatles are singing "Love, love, love" in the background, this is why the lower and upper diagonal are a bit shady. However, at time $t = 70s$, the Beatles are only singing "Love, love, love", with a repetition that is more clear in the lower and upper diagonal around that time.

4.2. Segmentation

In terms of segmentation, we will first compare the state transitions that are outputted by K-means and HMM, which we can visually observe on figure 3. As we can see, the K-means states are very varying. We witness lots of state-jumps, which are explained by the fact that those states are actually close to each other. The HMM clearly puts many K-means states into one state. In fact, we observe only 3 states for the HMM, in comparison with 9 states for the K-means. We should also note that for visibility reasons, we are only showing the 2.5 first seconds of the audio. The HMM mostly flattens the jumpy regions of the K-means, and manages to make a good generalisation of the state, which was not achieved by the K-means method.

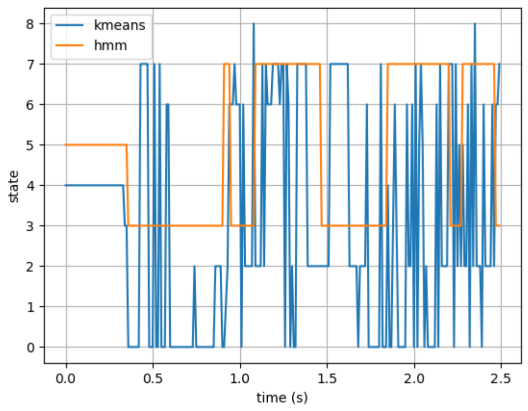

Figure 3: Comparing state transitions on the first 2.5 seconds of the music for two methods : K-means and HMM

We should also note that the HMM was initialized with the K-means labels and mean vectors for the prior information of the model. The K-means however was initialized with the merged segments of step 2 of the methodology. So this explains a little bit why the HMM is smoothing the results of the K-means, since it is provided with the K-means results to generalise.

On the other hand, we also plotted the state segmentation on the actual plot of the amplitude of the signal with respect to time, which appears on the appendix on figures (4, 5, 6, 7). Also, for visibility on the image, we only plotted the first 20 seconds of the audio array.

The first thing we observe is that all segmentations are completely different from one another. No similar properties are shared, unless for the K-means and HMM which share somewhat similar segments, especially if we look at the second half of the audio.

The first segmentation takes big chunks of audio together, even when visually they're not that similar. However, the second pass merges too much, as we can see here, it merge all the segments in the first 20 seconds together. This actually depends a lot on the type of features used for the study. Here, we chose 13-dimensional MFCCs. And since we put a threshold of 0.8 similarity, it merged too much information together. However, this was necessary to reduce the number of states in the audio.

Next, as we saw earlier, the K-means is too strict in its clustering process, allowing the formation of multiple jump-states in short periods of time. The HMM takes the information provided by the K-means and merges some chunks into one state to have a smoother representation of the states.

In the end, no representation looks like the other. Plotting these segments into different colors actually helps us visualize the signal the way each method sees it : for instance, in the first pass segmentation, we only compare each vector to the next one in order to see if they're in the same segment or not. The idea is that an audio segment must be continuous with respect to time, and on the plot we can see that the state-jumps occurs mostly whenever the signal brutally changes patterns. At the second pass segmentation, we may have merged too much the segments that are close to each other, however reducing a lot the number of states. Then, the K-means clusters the features provided with the second pass segmentation clusters. It actually does not care about continuity, it just sees distances withing clusters and reducing them is the only objective. This is why we observe lots of discontinuity in the clustering of K-means. And last, the HMM sees the sequence as a realisation of a probabilistic model, which provides a smoother way of dealing with the state transitions.

4.3. Music summary

In the end, what matters the most is the kind of summary that is outputted by the model. Recall that we used two types of summary generation, one involving the most repeated state, and another one involving the most repeated n-gram.

First, let's discuss about the most represented state. The idea is to count the number of repetition of each state, take the one that has the highest number of occurrence, and last output the longest streak of this state as a summary of the song. When we do this on the song ”All you need is love”, the result is the chunk of audio in which the Beatles sing ”Looo-”, which is the first part of the word ”love”. It is actually interesting to see that the summary is coherent with the title of the song. Also, it is satisfactory since the word ”love” seems to be repeated a lot in the song.

Second, we'll discuss about the most represented n-gram. The idea here is to play with transitions. We don't want the model to output only one syllable, but rather a pattern of transition states that might be more understandable to the ear, which would perhaps complete the summary ”Looo-” outputted by the most represented state. To do so, we transform our sequence so that once state cannot appear consecutively twice. For instance, if we originally had the sequence $A \rightarrow A \rightarrow A \rightarrow B \rightarrow B \rightarrow C \rightarrow C \rightarrow A \rightarrow A$, it would be transformed into $A \rightarrow B \rightarrow C \rightarrow A$. With that representation, we count the number of occurrence of n-grams in the new sequence, and extract the n-gram with the most repetition. Once we have it, we output the chunk of signal in which the n-gram occurred with the longest streak as a summary for the song. And for $n = 10$, the result is the chunk of audio in which the Beatles sing ”all you need is”, which is the first part of the title of the song, also repeated many times in the song.

5. Conclusion

In conclusion, we followed the methodology suggested by the original paper [3] and got interesting results : an interpretable segmentation with a HMM and a good music summary. However, we do need to understand what were the key points of this study, what could be changed and what should remain the same, which is why we drew a list of positive and negative points in the study.

Positive points

Using HMMs to model the sequence of states generating is a good idea for this project. It seems natural that HMMs allow a certain flexibility that other methods don’t, like for instance K-means.

The nice part of this study is that the results we got were very interpretable : the outputted summaries ar ein direct relation with the title, which shows that the method has some potential of summarization, even if the summary is not perfect.

Negative points

The use of unsupervised methods provides a small generalization of the pipeline to all music available nowadays. We have tried it on a given music, and it outputted results that seem okay. But what about other songs, that don’t share the same distribution of features along time. The visited method is too restricted to songs that share the same properties. This can especially be seen on the examples provided in the original paper : the number of states outputted by the model is very small, around 6 states maximum and with a threshold of similarity of 0.99. It tells us that the song they used in the paper was very well picked for the success of the pipeline.

Also, the use of K-means is unjustified in the paper. It seems unnatural to use K-means to study the patterns of a music. Hopefully, the authors used HMM in the end, but K-means could have been removed from the pipeline.

References

[1] J. Foote. Visualizing music and audio using self-similarity. In *ACM Multimedia*, pages 77–84, Orlando, Florida, USA, 1999. 1

[2] B. Logan and S. Chu. Music summarization using key phrases. In *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100)*, volume 2, pages II749–II752 vol.2, 2000. 2

[3] G. Peeters, A. La Burthe, and X. Rodet. Toward Automatic Music Audio Summary Generation from Signal Analysis. In *ISMIR*, pages 1–1, Paris, France, Oct. 2002. cote interne IRCAM: Peeters02c. 1, 5

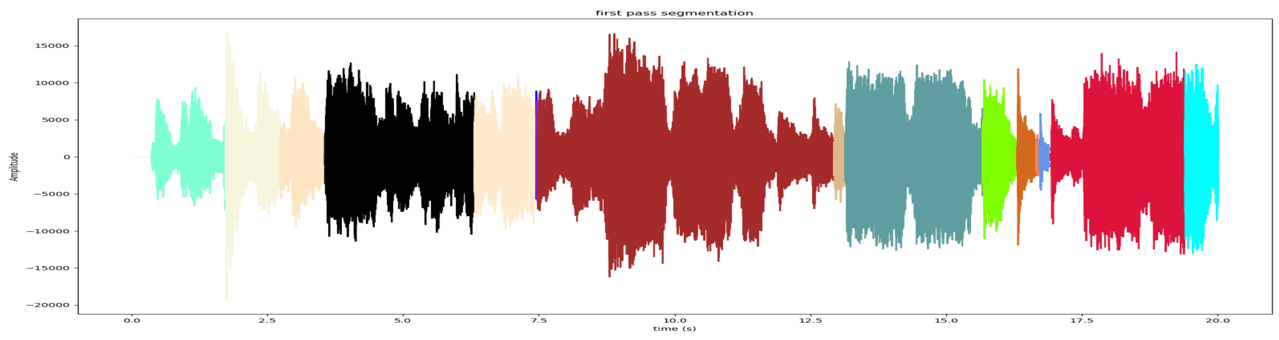

Figure 4: First pass segmentation, showing only the first 20 seconds of the audio

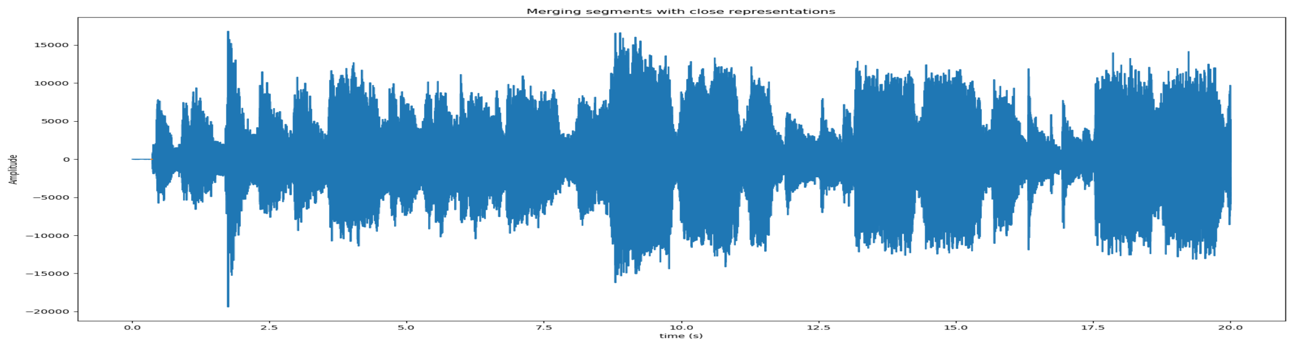

Figure 5: Second pass segmentation, after merging some segments, showing only the first 20 seconds of the audio

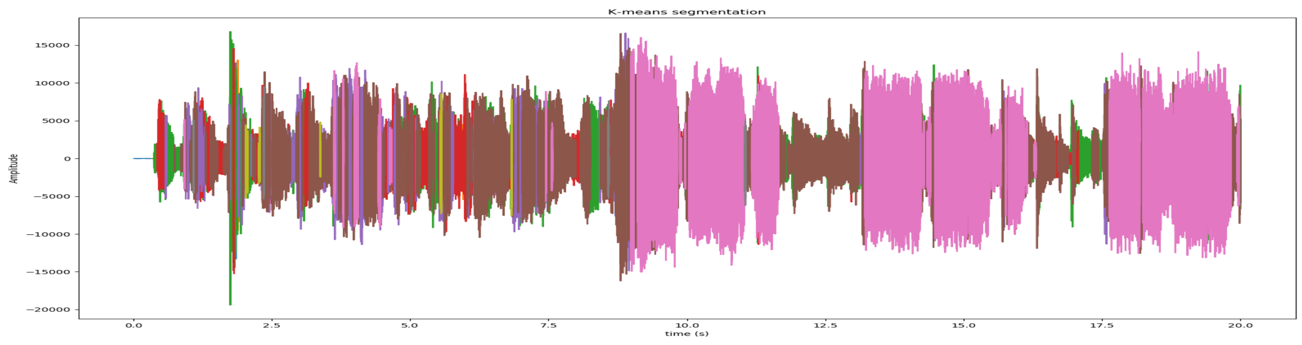

Figure 6: K-means clustering, showing only the first 20 seconds of the audio

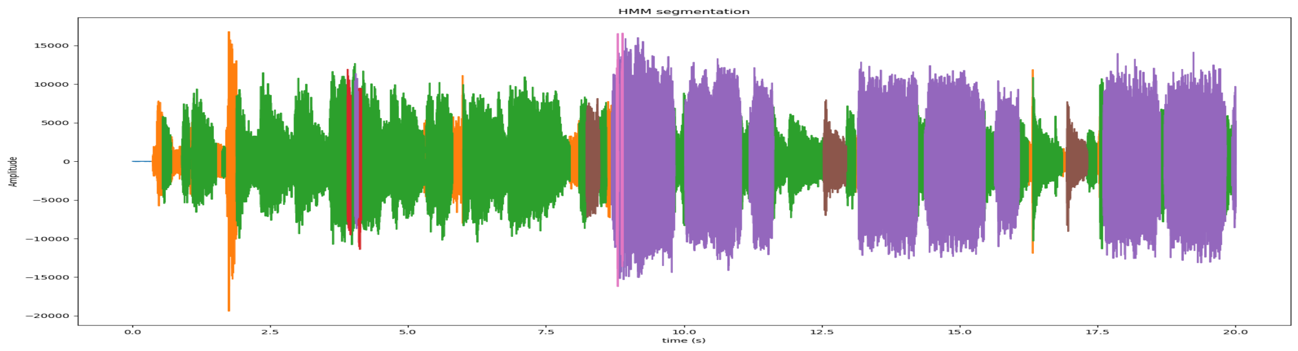

Figure 7: HMM prediction of states, showing only the first 20 seconds of the audio