PROJECT

Automatic Music Summarization via Similarity Analysis

By: Thibault Lahire (thibault.lahire@student.isae-supaero.fr)

Abstract
This report summarizes the main ideas and contributions of the reference article [1] published in 2002 by Matthew Cooper and Jonathan Foote. The theoretical ideas proposed in [1] are presented in the first part of this work, and are implemented and tested in the second part.

1 Main contributions

1.1 Set-up and context

Automatic music summarization is a research field which has emerged in the early 2000s with the popularization of internet and media-sharing platforms. Indeed, many media were uploaded and the need of an automatic summarization of the new and huge available content was growing. This article tackles the problem of music summarization, and tries to answer the question: Given a song, what can be done to extract a representative excerpt of this song ?

Answering this question is an important business challenge. Indeed, e-commerce music websites often let the customer listen to some excerpts to make him/her buy the entire song. With the amount of musics created every day, this task would be very costly if performed by humans. The reference paper of this report proposes a method which takes a few seconds to process a summary, relevant in some sense.

Indeed, one could immediately think of taking the first ten seconds of the song and considering it as being a good summary. If this can work for some songs, where the first ten seconds immediately gives the downbeat and the musical atmosphere of the whole work, this turns irrelevant for musics containing an introduction very different from the rest (which is sometimes the case for politically coloured musics, where the singer announces why he/she wrote the song...).

The proposed method is based on self similarity analysis. Roughly speaking, the song is divided into a certain number of fragments on which relevant features are extracted. Then, features of segments of the duration of the summary are compared to each other in order to extract the segment with the higher similarity. This method is then relevant to extract repetitive segments of music, such as a refrain.

Even if this method has been developed for music summarization, the main ideas could be applied to any other time dependent data, provided that relevant features

and relevant similarity measures are found. To give an overview of what was done in summarization at the time the article has been written, the authors give the example of statistical treatment of text data (through TF/IDF techniques) [2] [3], and the work on scene transition graphs is cited for video summarization [4].

1.2 Similarity analysis

The first pre-processing step consists in ensuring that all audio signals are in *the same format*. More precisely, this involves a resampling to obtain a 22.05 kHz mono format. Then, the audio signal is divided into 2048 sample "frames" at a 10 Hz rate, and then windowed by a Hamming window. Two types of parameterization were tested by the authors: Mel-Frequency Cepstral Coefficient (MFCC) [5] and spectrogram-based parameterizations. All this will be discussed later in the report.

To make sure the method we implemented works well, we apply it to the test example of the paper, which consists in a signal concatenating 3 sine waves: 30 seconds at 1 kHz, 40 seconds at 500 Hz, and 30 seconds at 2 kHz (see Fig. 1). Because the 500 Hz portion is the longest, the ideal summary (in the sense defined in this article...) should consist of the 500 Hz signal.

Once the paramerization of frames i and j has been extracted and embedded in the vectors v_i and v_j, a measure is needed to compute the similarity between i and j. In this work, we work with the cosine similarity defined by:

$$d_c(v_i, v_j) = \frac{\langle v_i, v_j \rangle}{||v_i||\ ||v_j||} \tag{1}$$

where $\langle v_i, v_j \rangle$ denotes the scalar (dot) product between vectors v_i and v_j. Contrarily to other measures, like the Euclidean distance, this measure remains relevant even when the frames i and j do not have the same energy (because of the normalization with the norms at the denominator).

A similarity matrix $\mathbf{S}$ can then be built from the distances [6] [7]. We introduce:

$$\mathbf{S}(i, j) = d_c(v_i, v_j) \tag{2}$$

Since d_c is symmetric, the matrix $\mathbf{S}$ is symmetric as well. Moreover, the maximum coefficients of $\mathbf{S}$ are found on the diagonal since every frame is maximally similar to itself.

It is possible to visualize the similarity matrix $\mathbf{S}$. The higher the value of $d_c(v_i, v_j)$, the brighter the pixels associated to the column (resp. row) i and row (resp. column) j. This can help us discovering the structure of the sound. The results obtained for the toy problem can be seen on Fig. 2.

1.3 Summarization technique

The authors consider that the segment which represents best the entire song is the one with the maximum similarity to the whole. Hence we expect that the longest

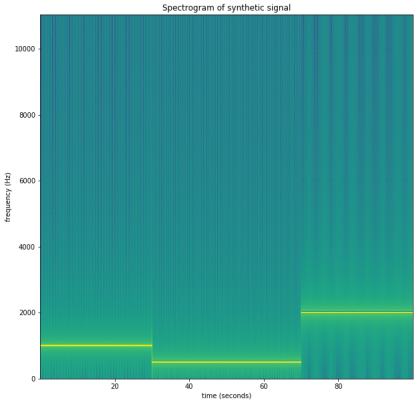

Figure 1: Spectrogram of the synthetic signal

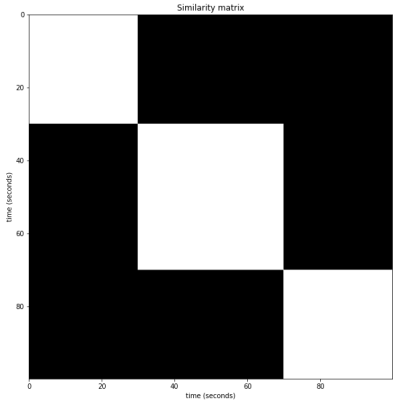

Figure 2: Similarity matrix for synthetic signal

element or most repeated element of the song is the best candidate. This is motivated by the fact that modern musics contain refrains. To illustrate what is done by the proposed algorithm, we start with a simple example.

Given the sequence ABBBCC, we want to find the most representative sequence of length 3. Following what has been obtained for the synthetic signal, we consider we have:

$$\mathbf{S} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{3}$$

We are interested in computing the average similarity between every subsequence of length 3. To do this, we summ the consecutiv rows (or columns) of length 3. In our example, we have 4 possible subsequences: ABB, BBB, BBC, BCC, with column sums seven, nine, eight, and seven, respectively. The highest scoring subsequence is BBB, with a score of nine. Hence this is the optimal 3-element contiguous summary of the sequence ABBBCC.

We introduce $L = q - r$ the length of the segment (with q the end and r the start). The average similarity of the segment is calculated as the sum of the self-similarity between the segment and the entire work (see Fig. 3), normalized by the segment length:

$$\bar{\mathbf{S}}(q, r) = \frac{1}{N(r-q)} \sum_{m=q}^{r} \sum_{n=1}^{N} \mathbf{S}(m, n) \tag{4}$$

where N is the length of the entire work (width and height of $\mathbf{S}$). A simple interpretation of $\bar{\mathbf{S}}(q, r)$ is the average of similarity matrix rows over the interval $q, ..., r$ (or equivalently,

the columns). Note that a weighting function w can also be applied:

$$\bar{\mathbf{S}}_w(q,r) = \frac{1}{N(r-q)} \sum_{m=q}^{r} \sum_{n=1}^{N} w(n)\mathbf{S}(m,n) \tag{5}$$

This weighting function can give more importance to excerpts located at the beginning of the song for example.

To find the optimal summary of length L, we find the excerpt of that length with the maximum summary score (eq. (4)). We introduce the score $Q_L(i)$ as:

$$Q_L(i) = \bar{\mathbf{S}}(i, i+L) = \frac{1}{NL} \sum_{m=i}^{i+L} \sum_{n=1}^{N} \mathbf{S}(m,n) \tag{6}$$

for $i = 1, ..., N-L$. The best starting point for the excerpt is the time q_L^* that maximizes the summary score:

$$q_L^* = \arg \max_{1 \le i \le N-L} Q_L(i) \tag{7}$$

Hence, the best summary starts at time q_L^* and ends at time $q_L^* + L$.

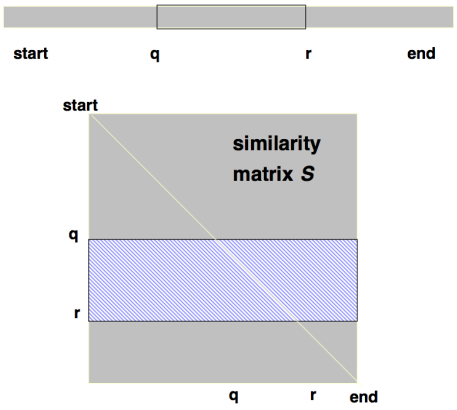

Figure 3: Calculating $\bar{\mathbf{S}}(q,r)$ by summing the similarity matrix over the support of the excerpt $q, ..., r$

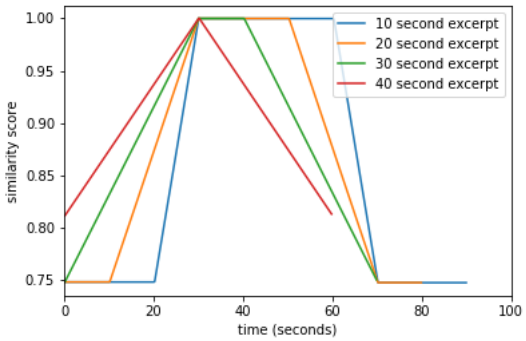

Figure 4: Summary scores $Q_L(i)$ computed from the similarity matrix of Fig. 2 for $L = 10, 20, 30$, and 40 seconds.

In the implementation we did, we produced Fig. 2 and Fig. 4 to make sure there was no bug. Fig. 4 shows the score with respect to the time for different length. This is exactly what was obtained in the article.

2 Implementation and critics

In this chapter, we try to challenge the article and discuss the choices made by the authors. When it is possible, a link with the course is done.

2.1 Results of the implementation and some challenges

We present in Fig. 5 the similarity matrix corresponding to the musical piece Vivaldi's Spring, $\sim$ 1725. The summary was a success. The similarity is high where we expect it to be high. Note that the results are different from those obtained by the authors because we probably do not use the same song (i.e. 2 different orchestra). In Fig. 6, we present the score over time for the same piece.

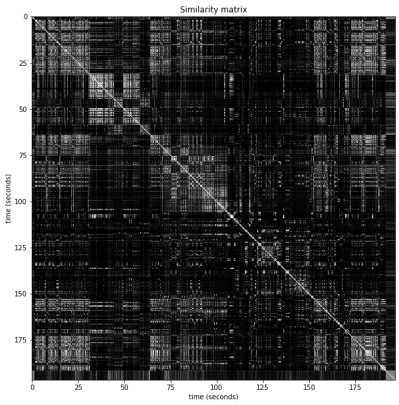

Figure 5: Similarity matrix for Vivaldi's Spring

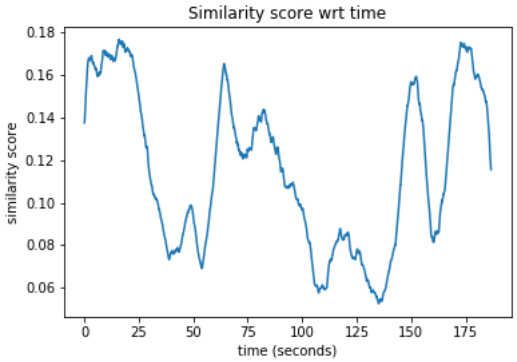

Figure 6: Summary scores $Q_L(i)$ computed from the similarity matrix of Fig. 5 for $L = 10$ seconds.

Another successful summary has been realized for the song Hair, The Cowsills, 1969. This song has been recorded on a scene and not in studio. The song does not start immediately, and there is a quite long speaking introduction. The similarity score captures what is expected by humans, as shown in Fig. 7 and 8.

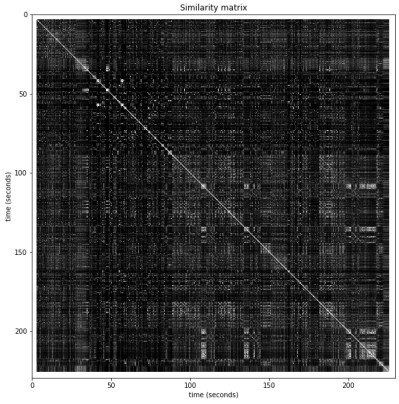

Figure 7: Similarity matrix for Hair

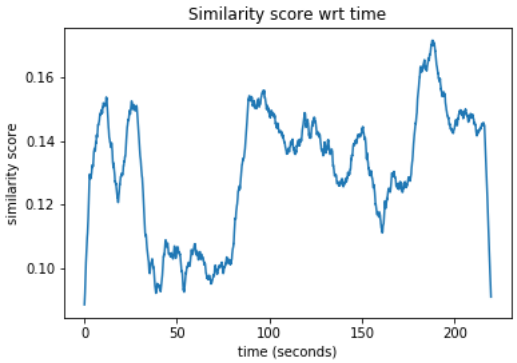

Figure 8: Summary scores $Q_L(i)$ computed from the similarity matrix of Fig. 7 for $L = 10$ seconds.

It is interesting to note that the algorithm has difficulties finding the highest similarity where it is difficult for humans. Indeed, the song Daddy cool, Boney M, 1976,

presents a similarity score almost constant over time, as can be seen on Fig. 9 and 10.

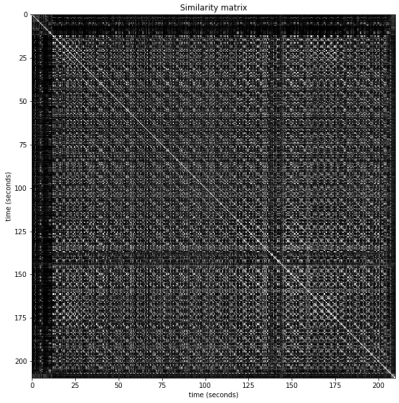

Figure 9: Similarity matrix for Daddy cool

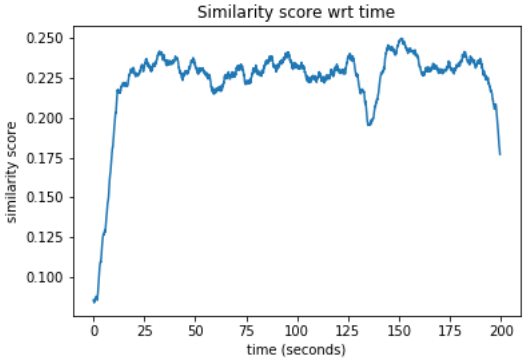

Figure 10: Summary scores $Q_L(i)$ computed from the similarity matrix of Fig. 9 for $L = 10$ seconds.

Now, we test the robustness of the proposed method by adding noise to the song. Two types of noise are tested: a white Gaussian noise and a salt-and-peper noise. In both cases, the method gives the same results, except when the level of noise is so high that the song is no more recognizable by a human.

Since many radio channels often accelerate the musics they broadcast in order to maintain a time framework (for example for interviews that have to be done in live...), we test the proposed algorithm on slightly accelarated songs. This transformation has been done using the TD-PSOLA (Time Domain Pitch Synchronous OverLap Add) method, but it is probable that radio channels use more powerful methods. Up to a factor $\times 1.03$, the proposed algorithm selects the same excerpt for summarization. The robustness to this transformation and to noise can be explained by the fact that no frequency has been added to the signal.

2.2 Time / frequency trade-off

As we saw in class (see Fig. 11), the size of the window has a huge impact on the spectrogram. Indeed, due to the uncertainty principle, the resolution in time and frequency are dependent. A spectrogram very localized in frequency (bottom of Fig. 11) is used to see the frequencies / harmonics of the studied sound, whereas a spectrogram more localized in time is used to see the formants (middle of Fig. 11), and is therefore more adapted to speech recognition tasks.

The authors used a window of length 100 ms (approximately), hence it can be deduced that they decided to work at the frequency level, which is a bit surprinsing since they want to capture refrains in the song. This choice is perfectly appropriate for classical musics such as Vivaldi's Spring. However, for songs with singing parts, we could

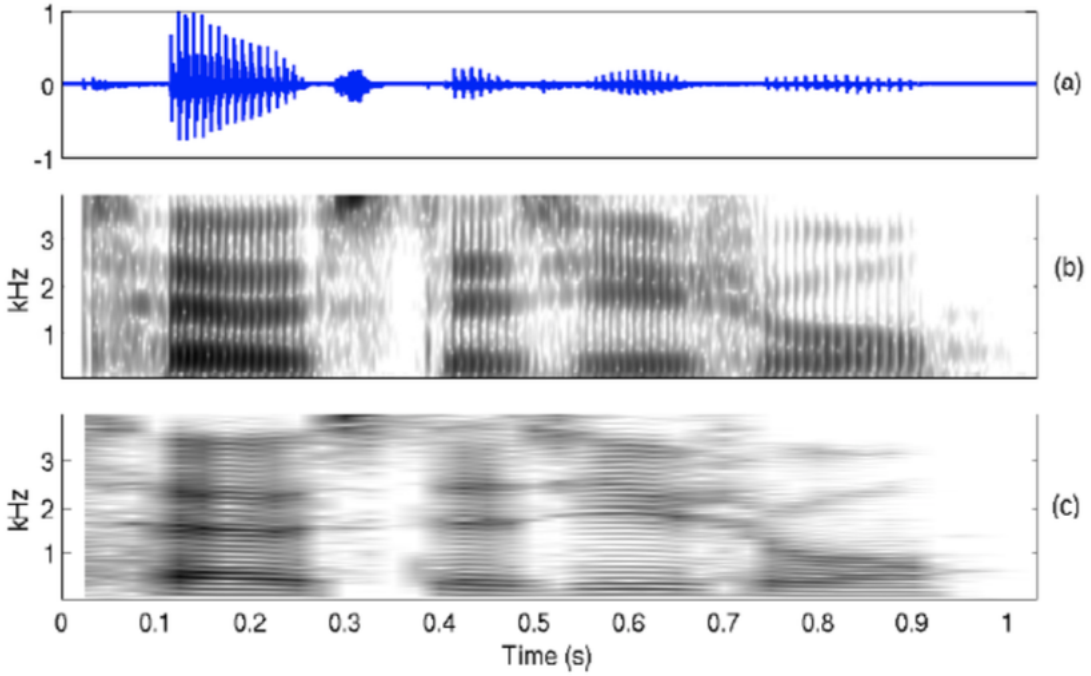

Figure 11: Spectrogram with narrow versus wide band

expect that a window allowing a formant analysis would be more appropriate.

Even if this remark makes sense, this is not true in practice and a window of length 100 ms seems to work in many cases. This is justified by the fact that the scalar product between the features v_i and v_j is a sum over all frequency bins. This sum does not change a lot between the frequency-level and the formant-level (see Fig. 11 (middle and bottom)). Moreover, we are doing two sums in equation 4. There is a sum over all time indices at each time index, and there is also a sum over multiple time indices to compute the similarity score. At the end of all these computations, too many averages have been done and we have lost the information given by the size of the window.

Abba's song Gimme gimme gimme, 1979, is an example that leads the proposed algorithm to a failure. Indeed, the algorithm has selected the segment 180s - 190s (see Fig. 12 and 13), which consists in a very "stable" and quite long instrumental part. This is unsatisfactory since many people would agree on the fact that a good summary of this song would contain the refrain "Gimme gimme gimme". However, from a *technical* point of view, this is perfectly coherent, since the algorithm captured the similarities in terms of frequencies.

We changed the window size and, as explained before, we did not managed to obtain a satisfactory summary for this song. It is interesting to note that the proposed method is invariant to the window size. We also have to mention that, in theory, another parameter has to be tuned when using a spectrogram: the size of the overlap. In practice, the method is also invariant to different sizes of overlap, so we keep it constant to a

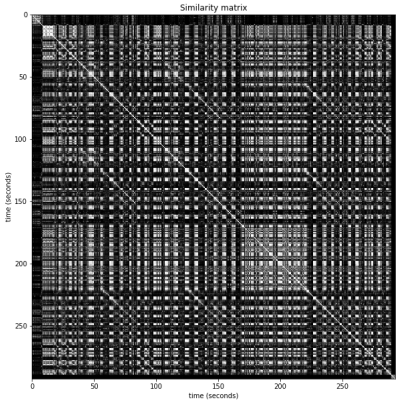

Figure 12: Similarity matrix for Gimme gimme gimme

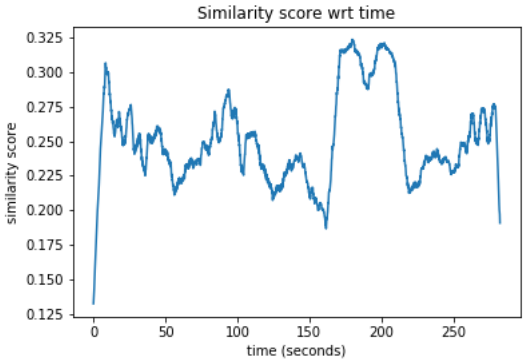

Figure 13: Summary scores $Q_L(i)$ computed from the similarity matrix of Fig. 12 for $L = 10$ seconds.

certain value over experiments.

Hence, it can be deduced that the method is very robust because it does not depend on hyper-parameters. This is quite unusual nowadays, where all the proposed methods requires hyper-parameters tuning (e.g. deep neural networks). We may wonder if introducing some hyper-parameters would not help doing a better summarization. Intuitively, we may think of a method whose performances are dependent on some hyper-parameters, which could be interesting because the algorithm would not rely on the same parameterization for each song.

However, how these hyper-parameters would be fine-tuned ? The answer to this question is not obvious at all. If we think of a Maximum Likelihood Estimation (MLE), we have to remember that maximizing a log-likelihood is equivalent to minimizing the KL-divergence with respect to an "ideal" distribution p^*. In our case, there is no "ideal" distribution p^*, or at least there is no **objectiv** ideal distribution, since everyone does not necessarily agree on the definition of a "good" summary.

To sum up, we started this subsection thinking the window size and the overlap size were two parameters to be fine-tuned. Due to the scalar product and the double sum in equation 4, it is not the case, and the method hence appears to be very robust, even if it can sometimes be fooled by "difficult" musics (e.g. Gimme gimme gimme).

2.3 Spectrogram or MFCC ?

In our implementation, we have mainly conducted our tests with spectrograms. However, the authors of the article mention the fact that MFCCs have also been applied successfully. Whereas there is no parameter to tune when using a spectrogram (as explained before), the number of parameters for MFCCs is quite high, as can be seen on Fig. 14.

Among the parameters which appear in Fig. 14, *lowerf*, *upperf* do not have to be

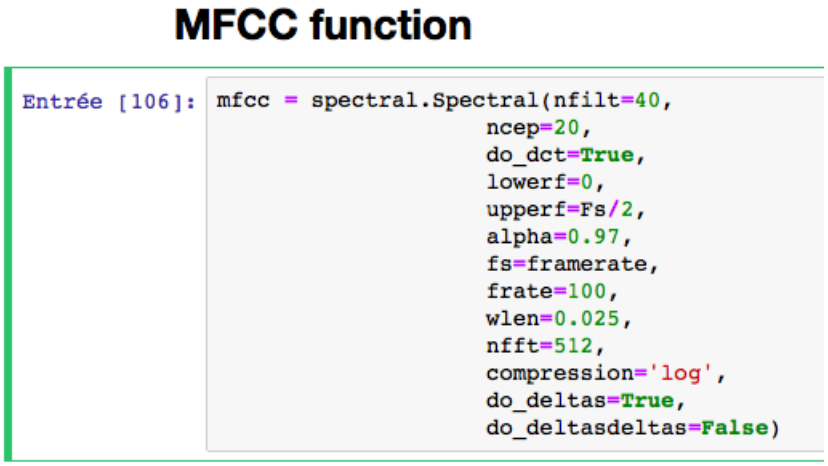

Figure 14: A large number of parameters has to be fine-tuned in order to get a parameterization through MFCCs. NB: The values on this picture were not used for our implementation and just serves as an illustration.

discussed. It corresponds to the frequency range of the analysis, starting from 0 Hz to half of the sampling frequency. However, there is no clear compromise for the other parameters, and at first sight the best comment to do is to say that they must be fine-tuned.

nfilt is the number of mel-filters to average spectrograms, and *ncep* is the number of cepstral coefficients to use for MFCCs. *do_dct* is True to compute MFCC (otherwise mel-filterbanks are the output). *alpha* is a parameter used for the pre-emphasis. *frate* is the number of frames per second (1/frate is the stride of the windows in seconds) and *wlen* is the length of windows in seconds. *nfft* is the number of frequency bins used to compute the spectrogram and *compression* is the final compression performed on the mel-filterbanks (before DCT when *do_dct*=True). Finally, *do_deltas* and *do_deltasdeltas* can be used to compute the first and second derivatives of MFCCs.

Even if classical values of some of the parameters above can be found in the literature, we might wonder if better results could be obtained by changing some of them. In particular, it is not clear whether mel-log filterbanks or MFCCs are better. Indeed, the difference between the two is a discrete fourier transformation which sometimes leads to a loss of information.

3 Conclusion

The article we studied presents a way to extract a small segment of a song representative of the whole. To do this, the authors compute similarity scores with the help of similarity matrices.

The proposed method for audio music summarization is robust to noise and transformations such as a slight acceleration. In this report, we discussed the choice of the

window length made by the authors, and concluded to the robustness of the method with respect to this parameter, which was not obvious at first sight.

In our implementation we only used spectrograms for features extraction. According to the authors, good results can also be obtained with MFCCs, but many parameters have to be fine-tuned in this case, and it becomes difficult for us to compare our implementation with the one of the authors.

References

[1] Matthew Cooper and Jonathan Foote. Automatic music summarization via similarity analysis. *Analysis*, pages 81–85, 2002.

[2] Jose Abracos and Gabriel Pereira Lopes. Statistical methods for retrieving most significant paragraphs in newspaper articles. In *Intelligent Scalable Text Summarization*, 1997.

[3] Klaus Zechner. Fast generation of abstracts from general domain text corpora by extracting relevant sentences. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*, COLING '96, page 986–989, USA, 1996. Association for Computational Linguistics.

[4] Boon-Lock Yeo and Minerva M. Yeung. *Classification, simplification, and dynamic visualization of scene transition graphs for video browsing*, volume 3312 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 60–70. 1997.

[5] L. R. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*. Englewood Cliffs, NJ: Prentice Hall, 1993.

[6] Jean-Pierre Eckmann, Sylvie Kamphorst, and D. Ruelle. Recurrence plots of dynamical systems. *Europhysics Letters (epl)*, 4:973–977, 11 1987.

[7] Jonathan Foote. Visualizing music and audio using self-similarity. In *Proceedings of the Seventh ACM International Conference on Multimedia (Part 1)*, MULTIMEDIA '99, page 77–80, New York, NY, USA, 1999. Association for Computing Machinery.