Aligning Music with Dynamic Time Warping Algorithm

Alfred Saidlo[1], Andreas Pung[1], Thamara Luup[1]

[1]Institute of Computer Science, University of Tartu

Introduction

Dynamic time warping is an algorithm for comparing how similar two time series are. It has many applications in a wide range of fields, including music. In this project, we attempt to apply time warping to simple melodies and see how it warps the music. We will verify the results visually (from a graph) and use the warping matrix to stretch/compress/shift parts of an audio file to align it with a similar one. Then we can listen to the resulting warped melody. The source code of the project is available in our Github repository [2].

DTW Algorithm

Dynamic time warping for time series a and b with lengths n and m can be achieved with the following python code:

```
def DTWDistance(a,b):
    n,m = len(a), len(b)

    DTW = [[numpy.infty for i in range(n)] for j in range(m)]

    DTW[0, 0] = 0

    for i in range(1,n+1):
        for j in range(1,m+1):
            d = distance(a[i], t[j])
            DTW[i, j] = d + min(DTW[i-1, j  ],
                                DTW[i  , j-1],
                                DTW[i-1, j-1])

    return DTW[n, m]
```

Clearly, this is an example of dynamic programming. In fact, it bears a remarkable resemblance to how we calculate edit-distance. Note that the *distance* function can be any distance metric we want depending on our data.

This algorithm fills in an $(n+1) \times (m+1)$ matrix row-by-row and returns the element at the end of the last row. That value represents how different those two sequences are when aligned optimally. While this value is useful, we need to examine the matrix more thoroughly, if we want to align those two sequences.

We can construct a list of pairs by backtracking from the last square of the resulting matrix. We can use this data to see which positions in one array correspond to positions in the second array when aligned optimally.

Time & Space Complexity

If we examine this algorithm, we can easily estimate the time complexity required to construct the time-warping matrix. Since we fill one cell at every step with a constant amount of work each time, we must do $O(N \cdot M)$ amount of work. Similarly, the matrix takes $O(N \cdot M)$ amount of memory to store.

While there are innovative ways to use less memory (for example only storing the last two rows in memory) these make it hard (or impossible) to backtrack.

If we want to backtrack then we use no extra memory but do some extra work. Still, the overall time complexity remains the same.

Musical Data

We wanted to use dynamic time warping to align two pieces of music. First, we must consider the data format we have. For simplicity, we used .wav files.

The Python Librosa [1] library enables us to load audio files as a time series of amplitude measurements. This, of course, is a usual way to represent music. However, one must consider the amount of data. With a 10-second music file, we had a time series with a length over 200 000 entries.

With two such melodies our matrix would be several hundreds of gigabytes in size. Clearly not optimal for any practical purposes. Especially considering the fact that we are talking about only a 10-second clip of music, not a whole song. Therefore, we clearly need a way to reduce the amount of data.

Chroma Feature

The solution, luckily, can also be found in the Librosa [1] library - chromagrams. Essentially the library:

- Divides the time series into chunks of size r.
- For each chunk it calculates how intensely twelve different pitches are represented.

Now we have a smaller time series consisting of vectors of size 12. Note that we can pick and choose r to make sure we do not use more memory than we can spare.

Below can be seen, how a small melody looks when represented using a spectrogram. The x-axis is time, while the y-axis represents different notes. The intensity is represented on a white-black scale.

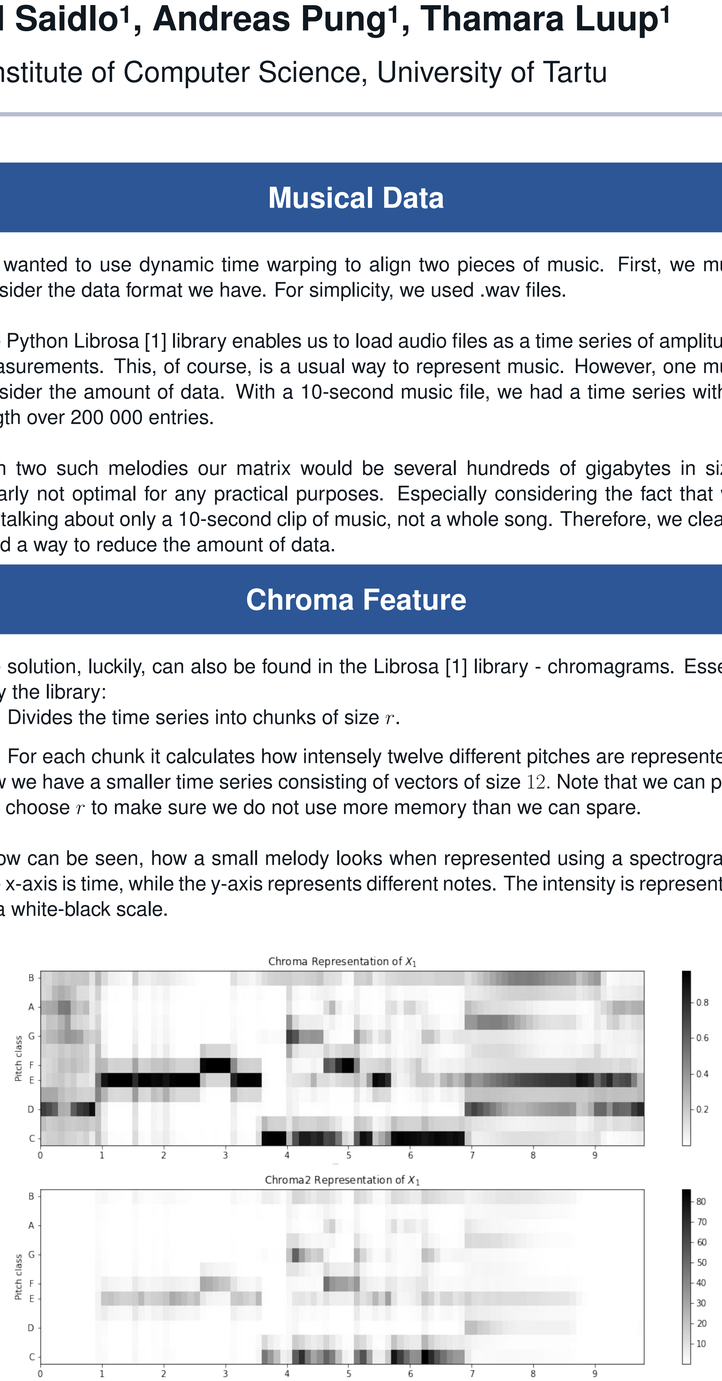

Fig. 1: DTW with different files

References

[1] Brian McFee et al. “librosa: Audio and music signal analysis in python”. In: *Proceedings of the 14th python in science conference*. Vol. 8. 2015, pp. 18–25.

[2] Alfred Saidlo, Andreas Pung, and Thamara Luup. *Aligning Music with Dynamic Time Warping Algorithm*. `https://github.com/andreaspung/dtw-music`. 2021.

Results

After calculating the chromagram, we had few enough features to apply the time warping algorithm. While for some examples, the warping seemed to perform quite well, on others, even looking at its visualisation showed it to be quite poor.

We used the result of time warping to modify one of the existing audio files to match the other (according to the time-warping output). Then we played it to see how similar the results sounded. The results varied from decent to awful and were almost always full of some noise.

Below you can see visualisations of how the algorithm aligned different versions of the same melody.

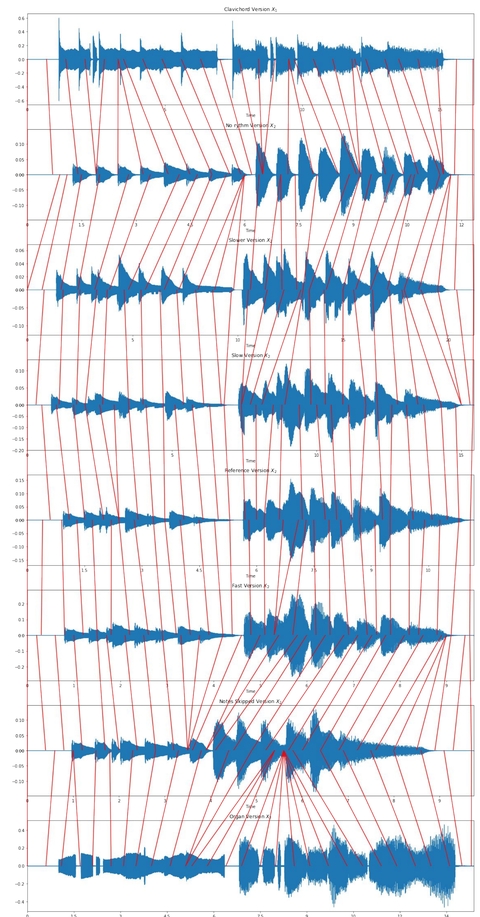

Fig. 2: DTW with different files