Durukan Demir

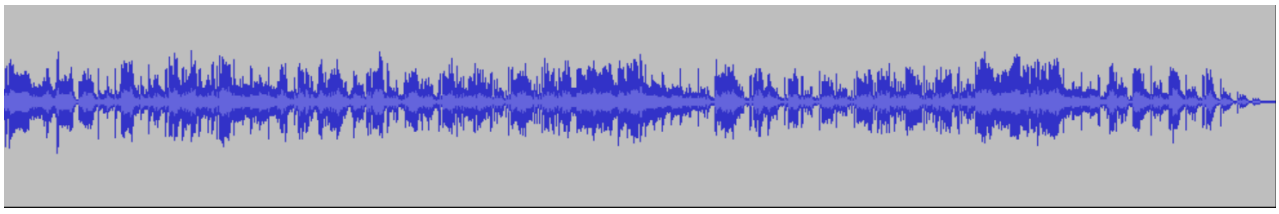
How Does Shazam Work?

Introduction

One of the mind-blowing pieces of technology was the app Shazam for me. If you're not familiar with it, it's a very compact mobile application, that when used listens to a song for at maximum 20 seconds, and then finds you the name of the song. Today, with the rise of AI, many people wouldn't be shocked to have such an application. However, let me tell you this, Shazam was created in 2002 and doesn't use machine learning at all. Learning how it worked truly amazed me as it used clear but still intricate mathematical tools, to create such an accurate and precise product. In this essay, we will dive deep into the field of signal processing, and when we're prepared, we will connect it back to Shazam. So let's try to tackle how we can recognize a song, just by listening a few seconds of it.

Fourier Series

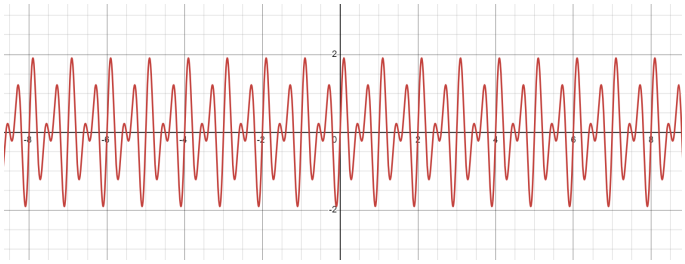
Sound is just waves of pressure moving in the air, so in theory, we can easily record the incoming sound, graph it, and check if the created graph matches with a preexisting graph i.e. a preexisting song. Right? Not really, because if you have a look at the graph for a real-world recording you'll quickly see that it's much more chaotic than what we imagine. Thus, the vague term "matching" works no more. So we must come up with a better method to identify different audio recordings.



Graph 1: Audio Recording of "It's a Man's Man's Man's World"

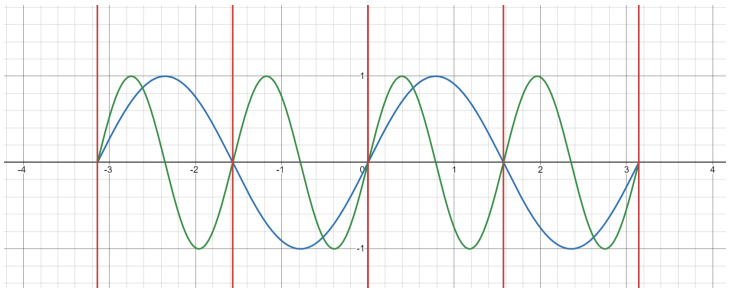
To do this, let's understand that in the real world, different sounds get combined and mixed, and our ears hear them all at the same time. So in this messy graph, it's just the multiple mixed waves, that make it challenging to identify some patterns. Hence, we must filter different layers of these waves to create some sort of I.D. tag for our recording.

Let's recall that every sinusoidal wave can be identified by three parameters: its frequency, amplitude and phase. In a mathematical formulation, this would be $f(x) = A * sin(\omega * 2\pi * x - h)$ where A is the amplitude, ω is its frequency and h is its horizontal shift i.e. the wave's phase. Our goal is to construct any given perodic function by using these sinusoidal waves. As a simpler example to work with, imagine that you receive the graph in Graph 2. I can tell you that it's the graph of $f(x) = sin(2 * 2\pi * x) + sin(3 * 2\pi * x)$. However, how could we find the different sine and cosine functions, and their amplitudes if we're just given the function?



Graph 2: graph of $f(x) = sin(2 * 2\pi * x) + sin(3 * 2\pi * x)$

One tricky method we'll use is the observation that the integral of sinusoidal waves equals zero, $\int_{-P}^{+P} sin(x)dx = 0$, because the positive sides cancel out the negative sides. However, $\int_{-P}^{+P} sin(x)^2 dx > 0$ as always $sin(x)^2 \geq 0$. However, when the frequencies of two waves that are multiplied and then integrated, don't equal, they cancel each other out too. Thus, $\int_{-\infty}^{+\infty} sin(Bx) * sin(Cx)dx = 0$ where B and C are non-negative integers and $B \neq C$. The proof can be made by integrating the function, however, an intuition can be gained just by looking at their graphs.



Graph 3: sin(2x) and sin(4x) in the interval [-π,π]

In graph 3, the blue line is $sin(2x)$, the green line is $sin(4x)$ and red lines separate every interval of length $\pi/2$. Notice how although the green line is the same between every red line, blue line's sign changes. Thus, when taking the integral, they cancel each other out. Thus, we can determine if a function f(x) contains a frequency ω, by checking whether $\int_{-\infty}^{\infty} f(x) * sin(\omega * 2\pi * x)$ equals 0 or not. As a note, this integral might equal to zero if the phase of the wave in $f(x)$ with frequency ω, doesn't match the phase of the sine wave at all, however, we'll discuss it later in this paper. Thus, we must check the same thing with cosine as well.

Now, we can determine if a certain frequency is present in a periodic function or not. There are two more parameters, amplitude and phase, of the waves that make that function. To talk more formally, let's define $f(x)$: $R \rightarrow R$ as a periodic function with a wavelength of $2L$. Then,

Durukan Demir

$f(x) = C + \sum_{1}^{\infty} A_n sin(n\pi x/L) + \sum_{1}^{\infty} B_n cos(n\pi x/L)$ where

$C = 1/2L * \int_{-L}^{L} f(x)dx,$

$A_n = 1/L * \int_{-L}^{L} f(x) * sin(n\pi x/L)\, dx$

$B_n = 1/L * \int_{-L}^{L} f(x) * cos(n\pi x/L)\, dx$

Wow, there's just too much to unpack here. As an initial warning, the calculations will be very messy, so I would suggest to not worry much if you can't follow them completely, as long as you manage to build the intuition.

Firstly, let's see that $sin(n\pi x/L)$ and $cos(n\pi x/L)$ are just the sine and cosine waves with a period $2L/n$. So the waves repeat n times in the interval $[-L, L]$. Furthermore, C can be thought as the mean or the mid line of the function (it's literally taking the average of the function in the interval $[-L, L]$). Now, let's focus on A_n. As explored priorly, if the wave with frequency n isn't present in A_n, then the integral will equal to 0. Thus, actually

$A_n = 1/L * \int_{-L}^{L} A * sin(n\pi x/L - k) * sin(n\pi x/L)\, dx$ where k is the phase of the wave with the desired frequency (assuming it's present) and A is its amplitude. When the integral is computed, it equals to

$A/4L * ((Lsin(k - 2\pi n))/(\pi n) + 2L * cos(k)) - A/4L * ((Lsin(k - 2\pi n))/(\pi n) - 2L * cos(k))$

$=A/4L * 4Lcos(k) = Acos(k)$

If the same is done for B_n,

$B_n = A/4L * (-2L * sin(k) - cos(k - 2\pi n)) - A/4L * (2L * sin(k) - cos(k + 2\pi n))$

$=- A/4L * 4Lsin(k) =- A * sin(k)$

So combining these two, if a wave with frequency $n/2L$ is present, then a term of $A_n sin(n\pi x/L) + B_n cos(n\pi x/L) = A * cos(k) * sin(n\pi x/L) - A * sin(k) * cos(n\pi x/L) = A * sin(n\pi x/L - k)$ weill be added, due to the sum of angles in the sine formula. Notice how finding A_n and B_n is the same procedure but for sine and cosine. Thus, it's as if finding the phase is determining how much sine and how much cosine is in a way.

In conclusion, with this formula, we've managed to separate all the waves that constituted the initial wave, find their respective amplitudes and phases. And congratulations, you've just learned Fourier Series.

Fourier Transform

Durukan Demir

Ok, we've managed to separate a wave into its constituent waves and it's much simpler to describe an audio recording. However, we have a few problems. Firstly, although we have a method, we don't have a function to determine how much a frequency is present in a function. Secondly, our calculations were based on a given function, nonetheless, an audio recording won't give any function to analyze but rather just data points.

First, let's go over the first problem. What we want is a transformation that takes function $f(x)$ and makes it $F(\omega)$, where $F(\omega)$ gives a value depending on whether the frequency ω is present in the function or not. This can be done with Fourier transform

So, let me introduce you to: $F(\omega) = \int_{-\infty}^{\infty} f(x)e^{-2\pi\omega xi}dx$

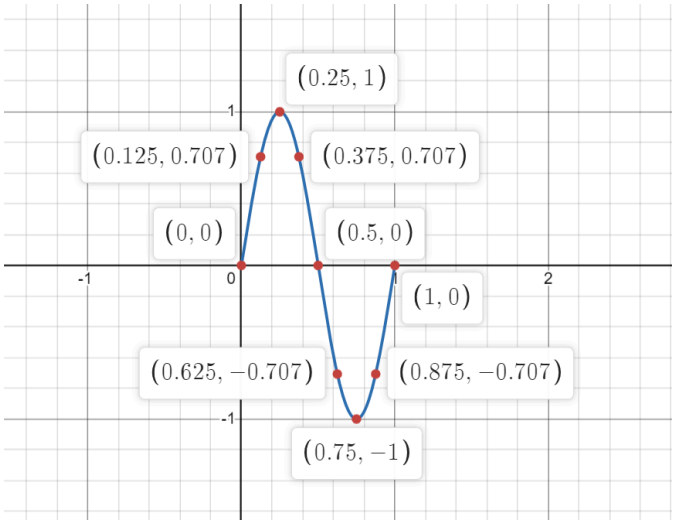
Wow, once again, there's much to unpack here. Firstly, don't let $e^{-2\pi\omega xi}$ scare you, because with Euler's formula $e^{ix} = sin(x)i + cos(x)$, this is just $e^{-2\pi\omega xi} = cos(2\pi\omega x) - sin(2\pi\omega x)i$.
But why are there even complex numbers in the first place? They're there just for convenience. Remember how we talked about determining how much sine and how much cosine there's in a function? There, we had to store the values in two different coefficients A_n and B_n. Instead of doing this, we'll store them in a single number where the number's imaginary part is how much sine there's, and its real part is how much cosine there's.
Thus, the equation is actually pretty simple

$$F(\omega) = \int_{-\infty}^{\infty} f(x)e^{-2\pi\omega xi}dx = \int_{-\infty}^{\infty} f(x) * cos(2\pi\omega x)\,dx - i\int_{-\infty}^{\infty} f(x) * sin(2\pi\omega x)\,dx$$

From the intuitions we've gained previously, we know that if frequency ω is not present in $f(x)$, then the integrals will equal zero, otherwise, because now the limits of our integrals are infinities, for periodic functions, the function will explode in those values.

Yet, we still haven't covered how to make this formula discrete, so let's do it now with discrete Fourier Transform.
First, we must get a few recording values which are evenly spaced.
Suppose that our audio recording looks like $sin(x * 2\pi)$, and we get N=8 samples, with an interval of 1/8.

Durukan Demir



Graph 4: Example Audio Recording's Graph

Here, let's store the values in a vector y, and let vector Y be the vector where Y_k gives the amplitude of a frequency k.

Then $Y_k = \sum_{n=0}^{N-1} x_n * e^{-2\pi i(k/N)*n}$

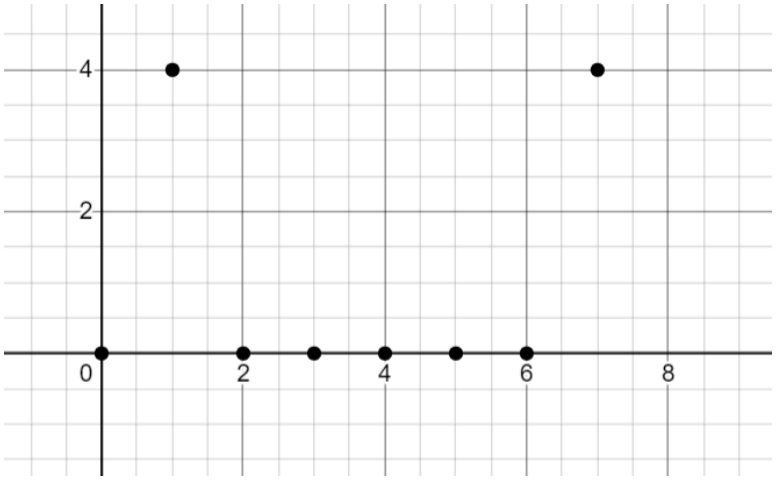
Here, compared to the original Fourier transform, we have k/N instead of ω, and n instead of x. If we compute the values for Y_0, since k=0 this is just the sum of the terms, thus it equals to 0.

For Y_1, thanks to the Euler's formula, we have

$$Y_1 = \sum_{n=0}^{7} x_n cos(2\pi * 1/8 * n) - i \sum_{n=0}^{7} x_n sin(2\pi * 1/8 * n) = -4i$$

This is logical as this is a sine function and there's no cosine in it, thus the real part is 0. Furthermore, Y_1 represents the value for the frequency of 1, and our sine function has a frequency 1. Then, we can just note the magnitude of this value as 4.

If we compute for all Y_n where $0 \leq n \leq 7$, only Y_1 and Y_7 will have a magnitude of 4, and the rest will equal 0.



Graph 5: Y values for the Discrete Fourier Transform of $sin(x * 2\pi)$

Having a non-negative magnitude at Y_1 is what we expected, however, why does Y_7 have one as well?

This is due to the Nyquist Limit, which is $sampling\ frequency/2$. In our case, the maximum sampling frequency was 8, so any value above 4 is going to be mirrored.

Understanding why this happens is simple, as

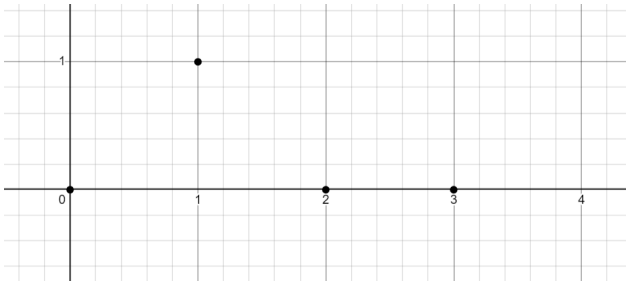
$$\sum_{n=0}^{N-1} x_n sin(2\pi * (N - k)/N * n) = \sum_{n=0}^{N-1} x_n sin(2\pi * n - 2\pi * k/N * n) = -\sum_{n=0}^{N-1} x_n sin(2\pi * k/N * n)$$

and for cosine,

$$\sum_{n=0}^{N-1} x_n cos(2\pi * (N - k)/N * n) = \sum_{n=0}^{N-1} x_n cos(2\pi * k/N * n).$$

So the values Y_{N-k} is the complex conjugate of Y_k. Since only the sign of the imaginary part coming from the sine value changes, the magnitude stays the same. Hence, we get rid of all the values at frequencies bigger than 4. Thus, if we want to measure signals with a maximum frequency n, then we must sample at least at 2n frequency.

Furthermore, since we've summed up all the values, we must divide by the number of left frequencies (which is 4 in this case) to normalize the amplitudes.



Graph 6: Final amplitudes of the frequencies.

Finally, we've managed to find the frequencies in this signal, and there's a wave with frequency 1 and amplitude 1. And congratulations once again, you've just learned discrete Fourier transform.

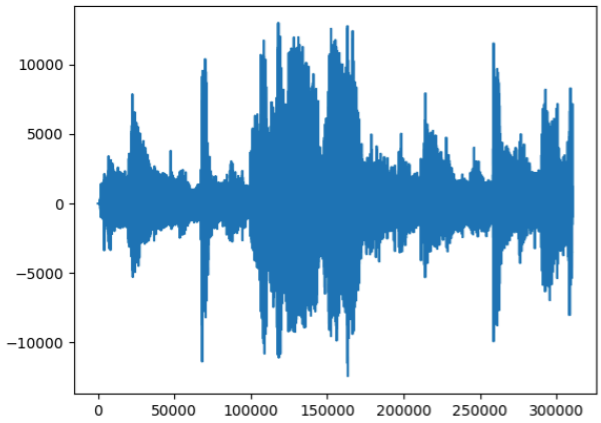
Actually, what Shazam uses is the Fast Fourier Transform, which is a family of algorithms that are faster versions of discrete Fourier transform and are crucial for real life applications. However, for the sake of brevity, I'll only encourage the reader to research it.

Applying our knowledge to a song

We've just covered Fourier series, Fourier Transform and Discrete Fourier Transform in a single paper. Wow. You might have lost track of the algebra at some point, but don't worry, as long as the key idea is there, you're good to go.
Fourier Series finds the coefficients for sine and cosine functions with different frequencies to sum and create the original function.
Fourier Transform creates a function which determines if a certain frequency is present in another function.
Discrete Fourier transform does what Fourier transform does on continuous functions, but with discrete data inputs. Furthermore, it helps to record the magnitudes of frequencies in a signal.

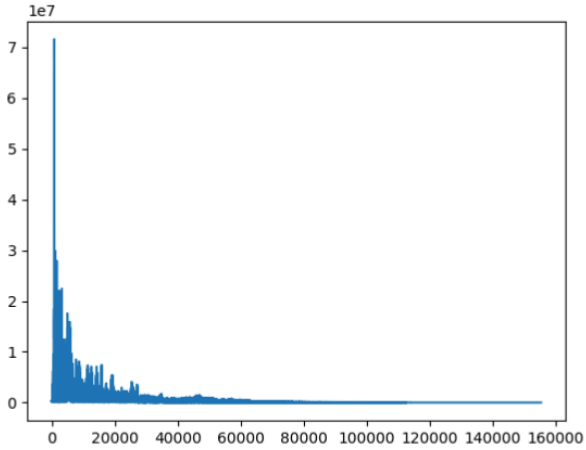
Now that we've learned to identify the components of a function, let's apply these to an audio recording, just by changing the context.

Here's an excerpt from the audio recording of James Brown's "It's a Man's Man's Man's World" plotted by Matplotlib in Python:



Graph 7: Voice Recording Graph of an Excerpt from "It's a Man's Man's Man's World"

Then, if we apply discrete Fourier transform to the song, it looks something like this:



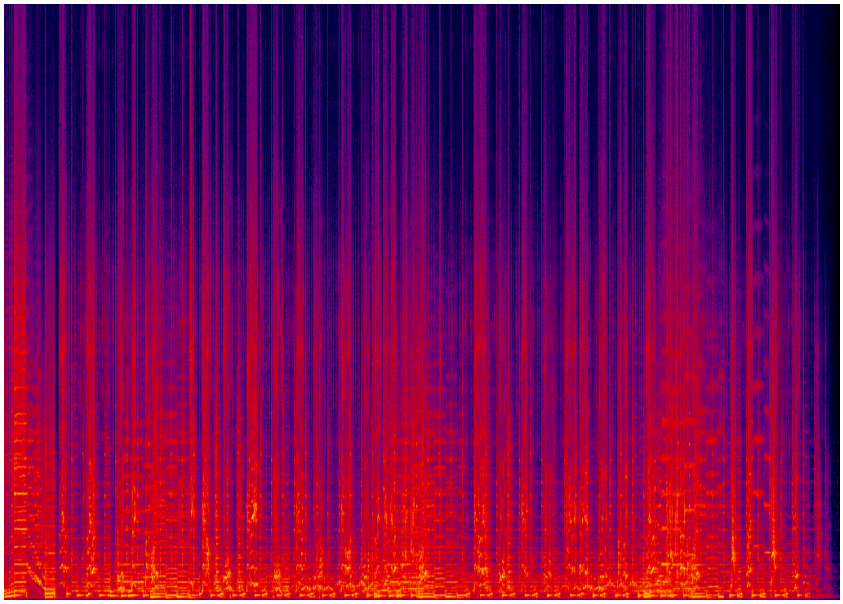
Graph 8: Frequency Graph of an Excerpt from “It’s a Man’s Man’s Man’s World”

Here, the peaks signal us that the excerpt contains sound waves with that certain frequency. You can see that the graph is skewed to the right too much, as there are much more possible large frequencies than there can be in the song, whereas there are only a limited number of small integer frequencies.

Spectrogram, Fingerprinting and Matching

Now that we can find the frequencies for a time interval in an audio recording, we can do this for the whole song. This will involve applying Fourier transform to very small time intervals, and noting the magnitudes of different frequencies with the time as well. After doing it, we can convert our data into a spectrogram where the horizontal axis is time, the vertical axis is a frequency value and the intensity of color in a coordinate is the magnitude of a certain frequency at that time instance.

Here is the spectrogram for the whole song “It’s a Man’s Man’s Man’s World” by James Brown



Spectrogram 1

Storing all of this data would be costly, so instead we'll just store values called peaks. Peaks are local maxima in this spectrogram. Since there's too much fluctuation in the data, instead of recording every local maximum, we will record values which are the maximum in a certain size area. By doing it and getting rid of every other information, we produce a graph named "constellation map", which was introduced by one of the founders of Shazam, Avery Li-Chun Wang, in his paper explaining Shazam's algorithm.

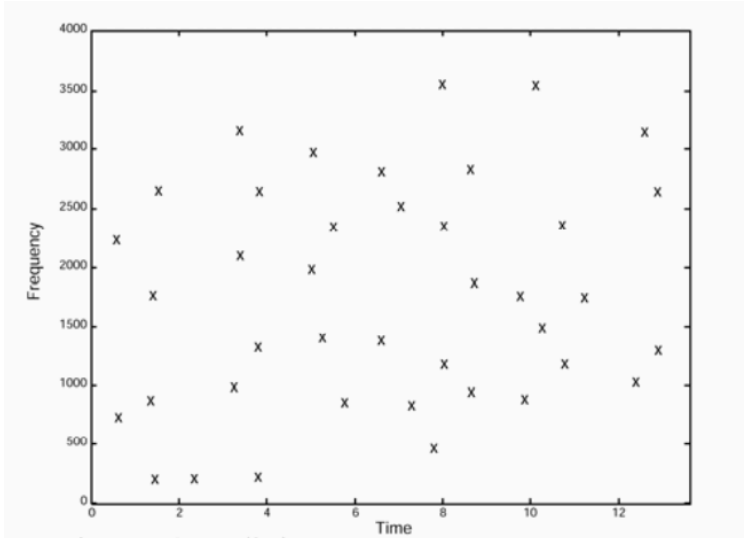


Figure 1: Constellation Map

Here, every point on the constellation map represents a peak in the audio recording. From the constellation map, "anchor points" will be chosen and a "combinatorial hash" will be applied. Although the general idea is present in Wang's paper, specific details such as the algorithm to

choose the anchor points or how the “target zone” is determined are not present. However, I still can provide a general description of the rest of the algorithm.

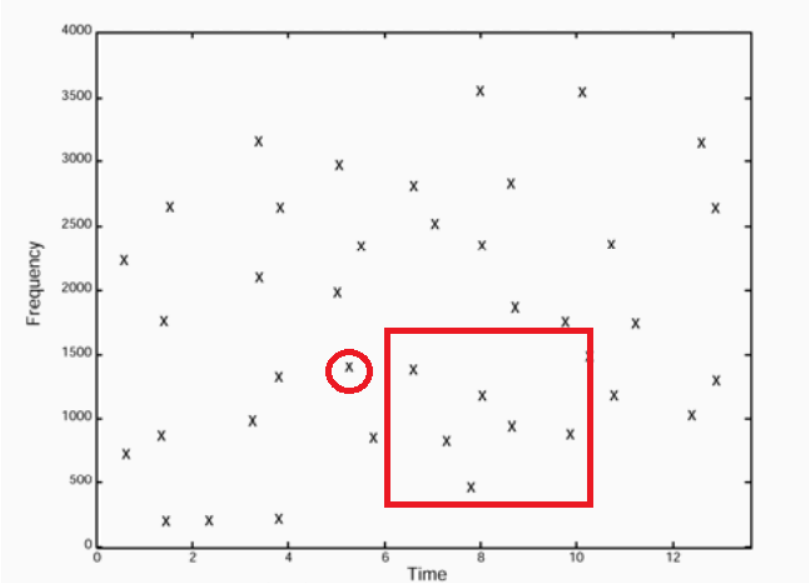


Figure 2: Anchor Point and Its Target Zone

Here, some of the points are chosen as “anchor points”, which are designated with a red circle in Figure 2. Then, a zone is chosen as the “target zone”, designated with a red rectangle in Figure 2. After this, the anchor point is paired with all of the points in the target zone and the values, frequency of the anchor point, frequency of the point in the target zone, and the difference of time between the two points are recorded in a vector.
After repeating this procedure a few times, we hash these values.
What’s even hashing? Hashing is the simple procedure of taking a set of values, defining and applying a function to them, and then recording the value. The key idea here is that the defined function should give values that seems completely random, and the chance of two different initial values having the same hash value should be really small.

For instance, a polynomial can be chosen as the hash function, such as $P(x, y, z,) = x * 2^{123321} + y * 3^{141356} + z * 5^{152342}$ where x and y are the frequencies and z is the difference in time, and the value could be the value of the polynomial in mod 1000000007. What are all these absurd values? The point is to make them really large and absurd, so that chance of collision is pretty low. Moreover, the hash function should always be the same, so that the same input always gives the same output. However, I must note that this isn’t the hash function that Shazam uses and is just an example.

Then, when such a hash function is chosen, all the aforementioned triplets are hashed and then combined to create a fingerprint for an excerpt of the song. Then, these values are recorded for many excerpts of millions of songs, then recorded in a database.

When the user uses Shazam, a new fingerprint is created for the audio recording and a song that has the same fingerprint is searched and then matched. Since similar songs, covers etc. might have the same fingerprint, in case of multiple matches, the audio recording is compared with the song excerpt to find the song with the highest match.

Conclusion:

And that's how Shazam could identify a song by listening for a few seconds, in 2002 without AI. To be honest, there was much more that I wanted to cover, especially in the final section as I tried to be quick in explaining some algorithms. I hope that this paper was a sneaky way to introduce you to Fourier Transforms and signal processing, topped with a load of clever algorithms.

Durukan Demir

Works Cited

The Discrete Fourier Transform (DFT). Directed by Steve Brunton, 2020. *YouTube*, www.youtube.com/watch?v=nl9TZanwbBk. Accessed 1 Apr. 2024.

Discrete Fourier Transform - Simple Step by Step. Produced by Simon Xu. *Youtube*, www.thefouriertransform.com/. Accessed 1 Apr. 2024.

"Fourier Series." *Wolfram Mathworld*, mathworld.wolfram.com/FourierSeries.html. Accessed 1 Apr. 2024.

"Fourier Transforms." *MathWorks*, www.mathworks.com/help/matlab/math/fourier-transforms.html. Accessed 1 Apr. 2024.

"An Industrial-Strength Audio Search Algorithm." *Columbia University Electrical Engineering*. *Columbia University Electrical Engineering*, www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf. Accessed 1 Apr. 2024.

"Introduction to Fourier Transform." *The Fourier Transform*, www.thefouriertransform.com/. Accessed 1 Apr. 2024.

James Brown - It's a Man's Man's Man's World. Directed by JamesBrownVevo, 2012. *YouTube*, www.youtube.com/watch?v=H77fRz1rybs. Accessed 1 Apr. 2024.

Jovanovic, Jovan. "How does Shazam work? Music Recognition Algorithms, Fingerprinting, and Processing." *Toptal*, www.toptal.com/algorithms/shazam-it-music-processing-fingerprinting-and-recognition#:~:text=One%20of%20the%20most%20popular,song%20you%20are%20listening%20to. Accessed 1 Apr. 2024.

MacLeod, Cameron. "abracadabra: How does Shazam work?" *Cameron MacLeod*, Feb. 2022, www.cameronmacleod.com/blog/how-does-shazam-work#why-is-song-recognition-hard-anyway. Accessed 1 Apr. 2024.