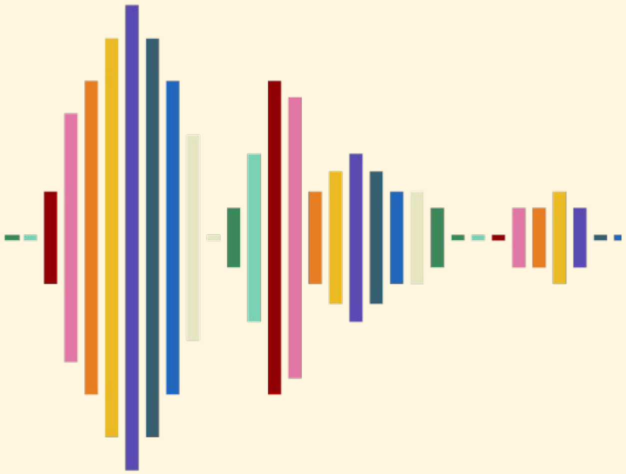
Sound Similarity using Locality Sensitive Hashing



Lukka Wolff

Project Impetus

Problem

In the world of music production, producers often download a plethora of sound packs that often contain sounds they may already have in their arsenal. Although they may be the same sounds, they are often called different things in different packs. This causes two distinct problems :

1. **Storage Congestion**
2. **Inefficient Sound Selection**

Mission

To **efficiently** minimize the number of duplicate sounds for music producers!

Implementations

Trivial Solution

- Generate sound signatures
- Loop through all sounds
- Evaluate cosine similarity for each pair of sounds
- Delete duplicates for high similarity sounds

Cons

- Time consuming for large data sets

Locality Sensitive Hashing

- Generate SimHash Signatures
- Bucketing with Hashing
- Identify Similar Items

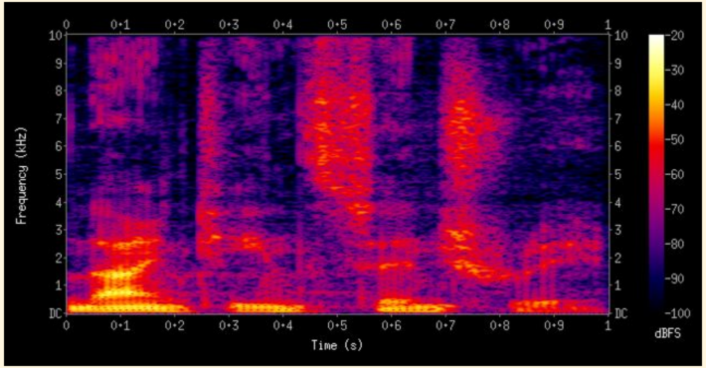
Pros

- Saves computational time !
- Sounds cooler

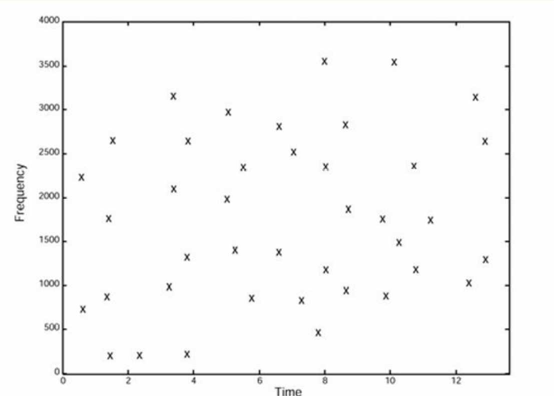
Preprocessing

Compile Sounds

Spectrogram



Peak Constellation Map

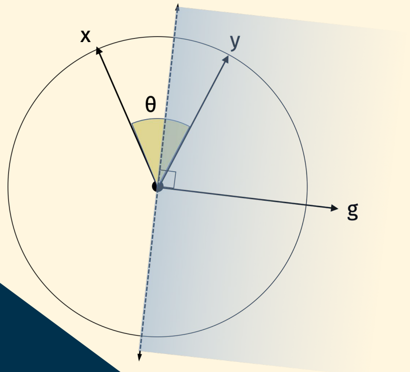


Vector Representation

LSH SimHash Implementation

Generate Signatures

Create a SimHash signature (cosine similarity) for each item in the dataset, which compactly represents its features.



Bucketing with Hashing

Divide each signature into multiple bands and hash these bands to buckets, such that similar items are more likely to be hashed to the same bucket.

Identify Similar Items

Search for potential matches by comparing items within the same buckets, considering them as candidates for being similar or duplicates.

Bands

3

Tables

8

Buckets

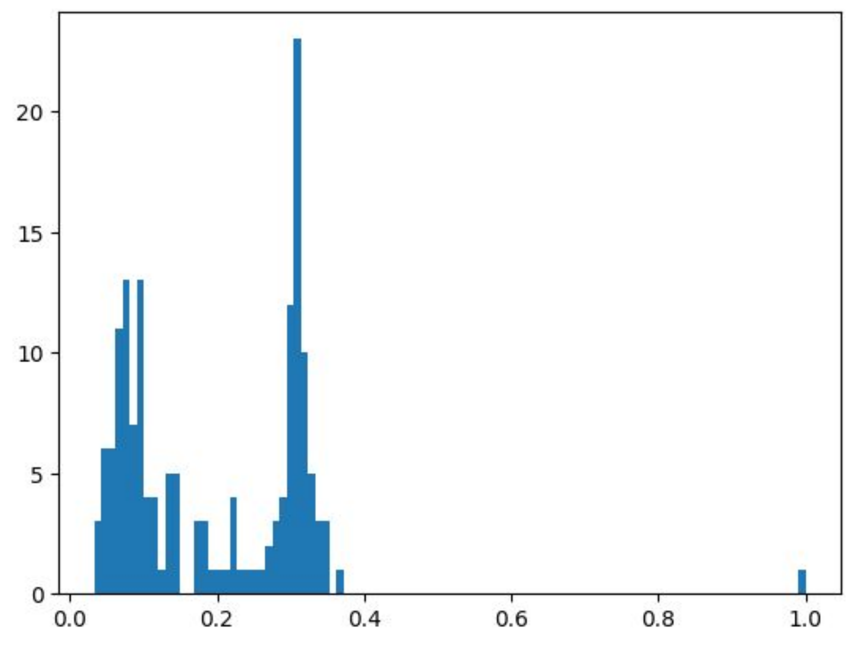
50,000

```
import math

upperbound = 0.5
lowerbound = 0.3
U_theta = math.acos(upperbound)
L_theta = math.acos(lowerbound)

def find_r_by_t(U_theta, L_theta):
    for t in range(1,1000001):
        r = math.floor( math.log((1 - ((1/10) ** (1/t))), (1 - (U_theta / math.pi))) )
        if (1 - ((1 - (L_theta ** r)) ** t)) <= 0.1:
            print(f"Bands: {r}\nTables: {t}\n")
            break


find_r_by_t(U_theta, L_theta)
```



Results

```
{('Clean Chant.wav', 'My Fav Chant.wav'): 1.0,
 ('Basic Trap Clap .wav', 'Go To Yeat Clap 1.wav'): 0.5693484,
 ('YeatxKan Hat 3.wav', 'YeatxKan Hat 4.wav'): 0.5872212,
 ('Go To Snare 3.wav', 'Go To Snare 4.wav'): 0.5597797,
 ('Go To Snare 3.wav', 'Uh Huh Snare.wav'): 0.5346267,
 ('Go To Snare 4.wav', 'Uh Huh Snare.wav'): 0.6612502,
 ('High Up There Snare.wav', 'Jackpot Snare.wav'): 0.9999999,
 ('Click Hat.wav', 'YeatxKan Hat 3.wav'): 0.5318401,
 ('Click Hat.wav', 'YeatxKan Hat 4.wav'): 0.5007596}
```

Project Improvements – Future

DAW Interface

Optimize Runtime

Improved Constellation Mapping

Deletion Implementation

Thank you !

Lukka Wolff - CSCI 1052: Randomized Algorithms for Data Science