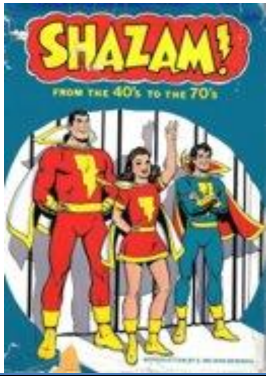
Music Clip Identification using Randomized LSH Tables

Pádraig Lysandrou and Samuel Wishnek

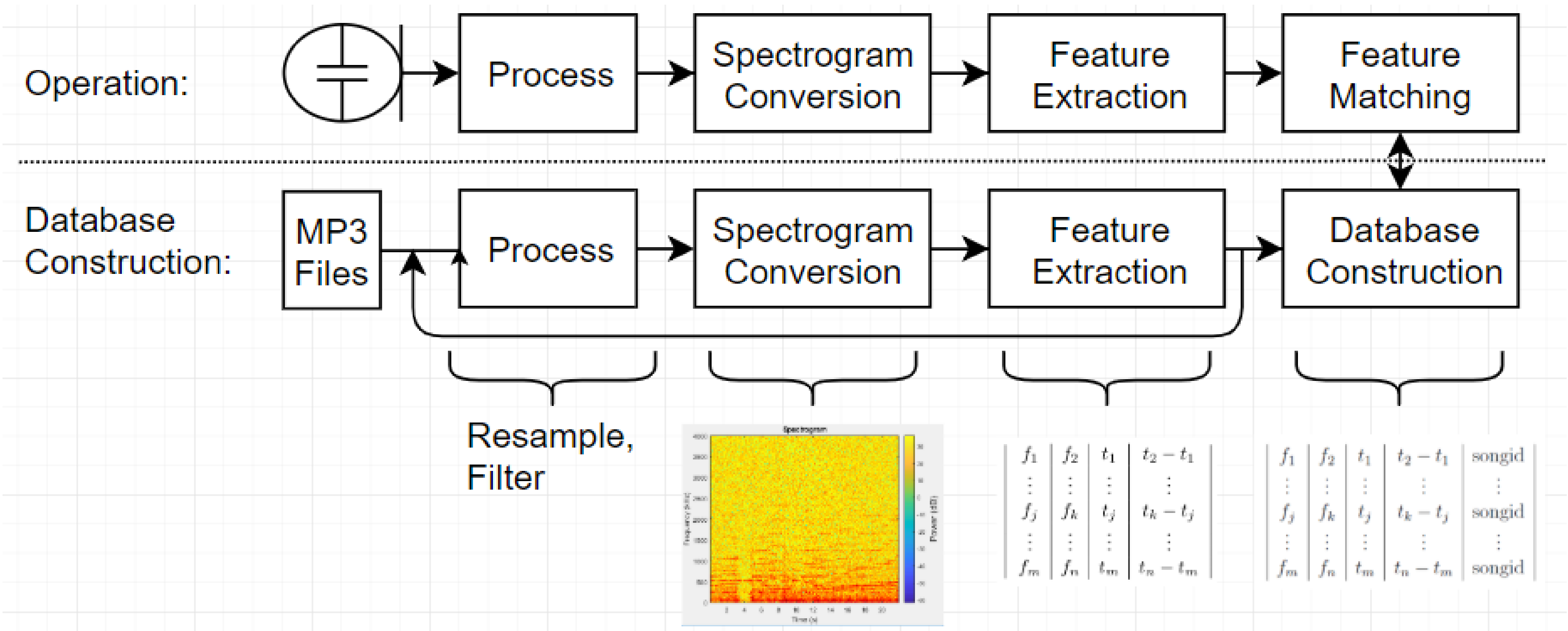
Problem Statement and Motivation

- Find the identity/source of a song given a small clip (<30 seconds)
- Interesting and nontrivial due to large amounts of detail in an sound wave, as well as random initial conditions and exogenous factors
- Development of these algorithms is motivated by people interested in discovering new artists/music



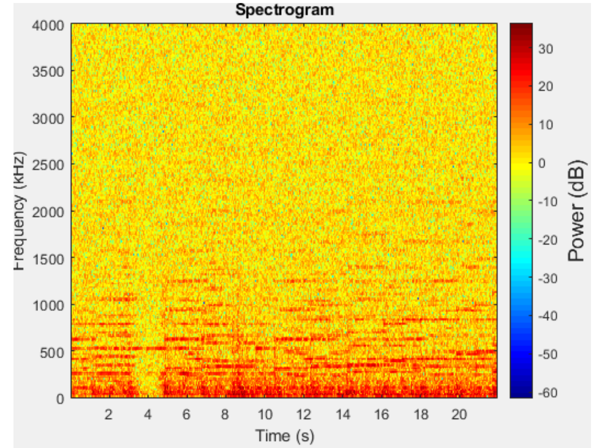
General Operation

- There seem to be many opportunities to implement RAs here



Spectrogram and Feature Selection

- Frequency and amplitude variations wrt time. Eg: Fallout 4 theme by Inon Zur



- Features first filtered by amplitude on a log scale, filtered to 30 per second, sort by magnitude, group by fanout distance.
- Frequency pairs are generated, with their coupled time differences into a single table per song

$$\begin{vmatrix} f_1 & f_2 & t_1 & t_2 - t_1 \\ \vdots & \vdots & \vdots & \vdots \\ f_j & f_k & t_j & t_k - t_j \\ \vdots & \vdots & \vdots & \vdots \\ f_m & f_n & t_m & t_n - t_m \end{vmatrix}$$

Shingle Generation and minHash function

- First a "shingle" is generated using a conventional hashing mechanism.

$$\boldsymbol{x} = T(:,2) + 2^8 T(:,1) + 2^{16} T(:,4)$$

$$h(x) = \min\left(\boldsymbol{a}x + \boldsymbol{b} \mod w\right) \in \mathbb{R}^k$$

- Then, we perform minHash on this shingle, for w being largest next prime number.
- Dimension of x can be anything, hash output dimension is fixed
- The coefficients a and b are randomly chosen integers less than the max value of shingle x

Matching Algorithm

- Just make a table and corresponding hash sequence of the clip, then search the database, taking advantage of locality sensitivity
- Find the highest value per table, and return ID

Algorithm 1 Simple Matching Function

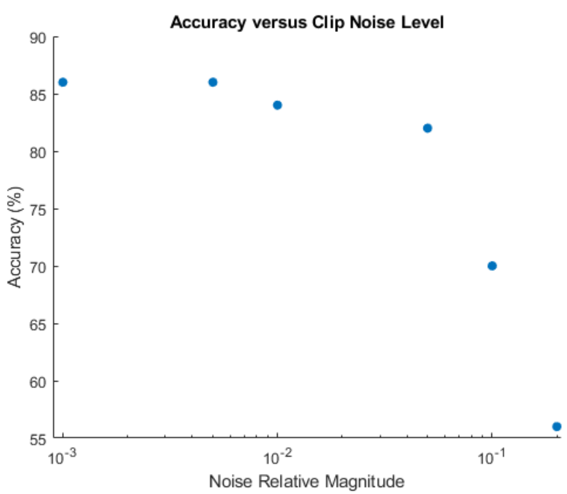
$Cliphashtable = minhash(make\_table(clip));$
$k = \text{const},\ \dim(h(x))$
$clip\_score = 0$
for i = 1:N **do**
    $hashTable\_sentence = minhashTable((i-1)*k+1 : i*k, 1)$
    $local\_sens\_bool = abs(Cliphashtable - hashTable\_sentence) \leq 100$
    $local\_score = \sum(local\_sens\_bool)$
    if $song\_score < local\_score$ **then**
        $song\_score = local\_score$
        $songName = i$
    end if
end for

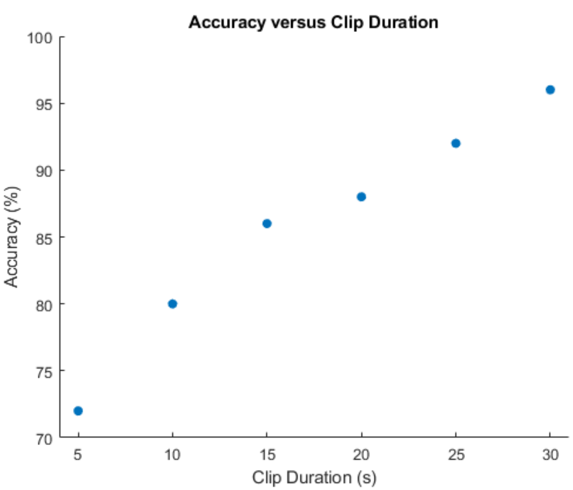
Alternative Approaches and Obstacles

- Various LSH Algorithms
  - MinHash
  - Euclidean Norm Distance Hash
  - SimHash
- Various Matching Algorithms
  - K nearest neighbours
  - Direct Equality (ends up being same as epsilon ball)

Plots and Demonstration

- Savings in required table size
  - Fifty test songs ~2 GB
  - Key Features ~200 kB (0.01%)
  - Randomized minHash table ~50 kB (0.0025%)





Final Results and Conclusions

For k=500, low noise = ~0.1*A, high noise ~1*A

- Detect 66% of 5 second clips with low noise
- Detects 58% of 5 second clips with high noise (basically static)
- Detects 88% of 15 second clips with low noise
- Detects 94% of 22 second clips with low noise

- MinHash algorithm works well, having static database entries for varying spectrogram thresholds is very useful

Future Work and Extensibility

- Alternative methods for association and classification
- Pulsar detection from radio observations
- Performing the lost-in-space attitude problem with an image of stars and a star catalog
- Many more

Questions?

References

- Original Shazam paper
- Hashing for Similarity Search: A Survey
- Near Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimension
- Near Duplicate Image Detection:min-Hash and tf-idf Weighting
- Computer Vision for Music Identification