Software Architect @ **Oto.net** / **Istanbul**

*B.Sc. Astronautical Eng.*
*M.Sc. Software Eng.*

Yes, a pizza lover!

ar@da.yalciner@gmail.com

wizardctp

ardayalciner
OTA Audio Identification

Matching an audio sample with a pre-recorded sound clip

- **Music track** recognition
- **Radio / TV station** detection
- Licensing
- **Second screen** applications
  - Previously on *<insert TV Show here>*
  - Track watched movies / TV shows
  - Nearby concerts of playing artist
  - Information on a currently speaking movie / TV show character
• In nature, sound propagates as sound waves.

• We measure sound pressure at specific intervals. This interval is called sample rate.

• A sample rate of 44.1 kHz means, we measured the sound pressure 44100 times per second.

• These discrete signals represent sound in a digital form.
Digital Sound Signals
Digital Sound Signals

- **Properties:**
  - **Bit depth:** # of bits a sample occupies
  - **Channels:** # of simultaneous recordings
    (1: mono, 2: stereo, etc.)
  - **Endianness:** Big-endian vs. Little-endian

- **File Formats:**
  - Uncompressed: **PCM, Wave**
  - Compressed:
    - Lossless: **FLAC**
    - Lossy: **MP3, AAC, Ogg**
Frequency Analysis

- Record or play audio signals in the *time domain*: SPL vs. Time

- Analyze audio signals in the *frequency domain*: Frequency vs. Amplitude vs. Time

![Diagram of frequency analysis](image)
Covers frequencies up to $0.5 \times \text{sample\_rate} \ [\text{Hz}]$

Divided into bins. Each bin represents the average amplitude for $0.5 \times \text{sample\_rate} / \text{fft\_points}$ wide of frequencies
Sensitive either in time dimension or frequency dimension: not both
Fingerprinting

Problem:

We need to uniquely summarize a part of an audio recording despite various challenges.

Approach Using:

- Music information retrieval (MIR)
- Acoustic fingerprinting
Fingerprinting: MIR

“What can we retrieve?”

More **specific**:  
- Musical features (*notes, chords, harmony, rhythm, …*)  
- Speech  
- Instruments  
- Melody: *Query by Humming*

More **abstract**:  
- Time-frequency peaks
Fingerprinting: Challenges

- **Noise**
  - **Duration**: instantaneous / continuous
  - **Frequency range**: small / wide
  - **Loudness**: quiet / loud

- **Echo**

- **Changes in tempo**

- **Changes in pitch**

- **Attenuation or boost in certain frequencies** (e.g., Equalization)
Fingerprinting: Time-Frequency Peaks

- Divide the spectrum into $N$ equal areas (e.g., 16 parts)
- For each area, find the **frequency bin** that provides the peak amplitude
Fingerprinting: Packing

<table>
<thead>
<tr>
<th>FFT Points</th>
<th>$P = 1024$</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Areas</td>
<td>$N = 16$</td>
</tr>
<tr>
<td># of Bins / Area</td>
<td>$0.5 \times P / N = 32$</td>
</tr>
<tr>
<td>Sample Rate</td>
<td>$SR = 11025$</td>
</tr>
<tr>
<td>Max. Frequency</td>
<td>$SR / 2 = 5513$</td>
</tr>
</tbody>
</table>

We can represent 5513 using a 16-bits integer. 16 of them occupies **256-bits** (32 bytes).

However, we can represent 32 with 5-bits. It is possible to store them in **80-bits** (10 bytes).

$$fp = \sum_{i=0}^{16-1} (b[i] - i \times 32) \times 2^{5\times i}$$
Fingerprinting: Hashing

8x frequency bin offsets

(1) Select an area

(2) Find 1-vertical; 2-horizontal neighboring areas

(3) Generate combination vectors

<table>
<thead>
<tr>
<th>120607</th>
<th>090607</th>
<th>040607</th>
</tr>
</thead>
<tbody>
<tr>
<td>120603</td>
<td>090603</td>
<td>040603</td>
</tr>
<tr>
<td>120632</td>
<td>090632</td>
<td>040632</td>
</tr>
</tbody>
</table>
Selection of audio information
- Should be robust
- Should be as unique as possible

The FFT algorithm
- Managing losses due to the uncertainty principle
  - Time-resolution = 1 / Frequency-resolution
- Discrete-time FT or Short-time FT
- # of FFT points
Static Database

Audio File → Audio Converter → Track Analyzer → Fingerprint Database

Partition data. For a uniform distribution, use a generated hash as the partition key.

FFmpeg & SoX are perfect for audio conversion

Indexing Database

Meta Data

Cache frequently accessed data in memory.
Streaming Database

1. Audio Stream
2. Audio Converter
3. Stream Analyzer
4. Real-time Database

Consider latencies

Use a distributed filesystem. Give deterministic file names.
**Streaming Database**

**Stream name** | **Timestamp**
---|---

Content: $T = YYYYMMDDHHAB$ file contains fingerprints from the moment $T$ to $T + 4$ minutes

Reading: At $t = YYYYMMDDHHAB$ moment, the file corresponding to the $T = t - 2 - (B \& 1)$ timestamp will be opened.

Writing: At $t = YYYYMMDDHHAB$ moment, files corresponding to $T1 = t - 2 - (B \& 1)$, $T2 = T1 + 2$ timestamps will be written.

In $YYYYMMDDHHAB$ format
A: $\{0, 1, 2, 3, 4, 5\} \rightarrow$ High minute
B: $\{0, 2, 4, 6, 8\} \rightarrow$ Low minute

FOSDEM / 201601301648.fingerprint
Identification

*Find the best matching fingerprint, if there is any*

Strategy

- Reduce the search space by elimination
- Rank candidates by detailed comparison

Outcomes

- **True positive:** We found the correct match
- **True negative:** We found a correct non-match
- **False negative:** We couldn't find the correct match
- **False positive:** We found an incorrect match
For each hash, try to find exact matches.

For each matching hash, calculate the time difference.

Create a histogram for time difference vs. match count.

Eliminate candidates where the best histogram score is less than a predefined value.
Identification: Ranking

Spectrum score: 3
Window score: 106
Shift the window
Testing & Optimization

- Mix samples with:
  - White noise of varying volumes
  - Pre-recorded noise

- Record samples under different acoustic conditions

- Make the configuration dynamic and use a machine learning algorithm to select the best configuration
THANKS!

More will be at:

github.com/wizard/fosdem2016

- Links to open-source software
- Source code for everything we talked about
- Markdown documentation for this presentation
- Dockerfile
References

- FOSDEM icon: https://fosdem.org/2016/
- Email icon: https://thenounproject.com/term/mail-with-at-sign/71812/
- FFmpeg: https://www.ffmpeg.org/
- SoX: http://sox.sourceforge.net/
- Sonic Visualizer: http://www.sonicvisualiser.org/
- Audacity: http://audacityteam.org/
- PostgreSQL: http://www.postgresql.org/
- Redis: http://redis.io/
- Solr: http://lucene.apache.org/solr/