EFFECTIVE COVER SONG IDENTIFICATION BASED ON SKIPPING BIGRAMS

Xiaoshuo Xu, Xiaouou Chen, Deshun Yang
Institute of Compute Science and Technology, Peking University
128 Zhongguancun North Street, Haidian District, Beijing 100871, P.R.China
Outline

- What is cover song identification?
- Application: detect copyright infringement, music retrieval, etc.
- Challenge: Key transposition, structure and speed change
- Existing methods: Sequence alignment, Music representation
- Our approach
  - Represent music with skipping bigram histogram
  - Utilize inverted index to accelerate the calculation
Pipeline

1. **N reference recordings in the dataset**
   - CENS extraction
   - Vector embedding
   - Vector quantization
   - Counting
   - Code sequences
   - Bigram histograms

2. **Query recording**
   - CENS extraction
   - Key Transposing
   - Vector embedding
   - Encoding
   - Counting
   - Code sequences
   - Bigram histograms

3. **Codebook**
   - Inverted index
   - Retrieval
   - Ranking
Feature extraction

- Chroma Energy Normalized Statistics (CENS)
- Key transposition
  - Given a CENS vector \( x = (x_0, x_1 ... x_{11})^T \), the transposed vector is defined as follows:
    \[
x^{(i)} = (x_{i\%12}, x_{(i+1)\%12} ... x_{(i+11)\%12})^T
    \]
  - Given a CENS sequence \( X = [x_1, x_2 ... x_M] \), the transposed sequence would be:
    \[
    X^{(i)} = [x_1^{(i)}, x_2^{(i)} ... x_M^{(i)}]
    \]
- Vector Embedding
  - Embedded vector: \( \hat{x}_j = [x_j^T, x_{j-1}^T ... x_{j-(m-1)}^T], j = m, m+1 ... M \)
  - Embedded sequence: \( \hat{X} = [\hat{x}_m, \hat{x}_{m+1} ... \hat{x}_M] \)
  - Transposed embedded sequence: \( \hat{X}^{(i)} \)
Feature extraction

- Chroma Energy Normalized Statistics (CENS)

Given a CENS vector $\mathbf{x} = (x_0, x_1 \ldots x_{11})^T$, the transposed vector is defined as follows:

$$x(i) = (x_{i \mod 12}, x_i + 1 \mod 12 \ldots x_i + 11 \mod 12)^T$$

Given a CENS sequence $\mathbf{X} = [x_1, x_2 \ldots x_M]$, the transposed sequence would be:

$$\mathbf{X}(i) = [x_1(i), x_2(i) \ldots x_M(i)]$$

- Vector Embedding

Embedded vector:

$$\mathbf{x}_j = x_j^T, x_{j-1}^T \ldots x_{j-M}^T$$

Embedded sequence:

$$\mathbf{X} = [\mathbf{x}_m, \mathbf{x}_{m+1} \ldots \mathbf{x}_M]$$

Transposed embedded sequence:

$$\mathbf{X}^{(i)}$$
Feature extraction

- Chroma Energy Normalized Statistics (CENS)
- Key transposition
  - Given a CENS vector \( x = (x_0, x_1 \ldots x_{11})^T \), the transposed vector is defined as follows:
    \[
x^{(i)} = (x_{i \% 12}, x_{(i+1) \% 12} \ldots x_{(i+11) \% 12})^T
    \]
  - Given a CENS sequence \( X = [x_1, x_2 \ldots x_M] \), the transposed sequence would be:
    \[
    X^{(i)} = [x_1^{(i)}, x_2^{(i)} \ldots x_M^{(i)}]
    \]
- Vector Embedding
  - Embedded vector: \( \hat{x}_j = [x_j^T, x_{j-1}^T \ldots x_{j-(m-1)}^T] \), \( j = m, m+1 \ldots M \)
  - Embedded sequence: \( \hat{X} = [\hat{x}_m, \hat{x}_{m+1} \ldots \hat{x}_M] \)
  - Transposed embedded sequence: \( \hat{X}^{(i)} \)
Feature extraction

- Chroma Energy Normalized Statistics (CENS)

Given a CENS vector \( \mathbf{x} = (x_0, x_1, \ldots, x_{11})^T \), the transposed vector is defined as follows:

\[
x(i) = (x_{i \% 12}, x_{i + 1 \% 12}, \ldots, x_{i + 11 \% 12})^T
\]

Given a CENS sequence \( \mathbf{X} = [x_1, x_2, \ldots, x_M] \), the transposed sequence would be:

\[
\mathbf{X}(i) = [x_1(i), x_2(i), \ldots, x_M(i)]
\]

- Vector Embedding

Embedded vector: \( \mathbf{x}_j = [x_j, x_{j-1}, \ldots, x_{j-(m-1)}], j = m, m+1, \ldots, M \)

Embedded sequence: \( \hat{\mathbf{X}} = [\hat{x}_m, \hat{x}_{m+1}, \ldots, \hat{x}_M] \)

Transposed embedded sequence: \( \hat{\mathbf{X}}^{(t)} \)
Vector quantization and encoding

- Vector quantization is used to cluster embedded vectors and a codebook is learnt for encoding.
- Reduce the impact of structural variations.
- Code sequences of cover songs reveal high similarity, while code sequences of different songs show little similarity.
Vector quantization and encoding

Vector quantization is used to cluster embedded vectors and a codebook is learnt for encoding. Reduce the impact of structural variations. Code sequences of cover songs reveal high similarity, while code sequences of different songs show little similarity.
Vector quantization and encoding

- Vector quantization is used to cluster embedded vectors and a codebook is learnt for encoding.
- Reduce the impact of structural variations.
- Code sequences of cover songs reveal high similarity, while code sequences of different songs show little similarity.
Bigram histogram and similarity

- Count the bigram histogram $f$
- The similarity between two songs is defined as:
  \[ S(u, v) = \max_i \sum_{a,b} \min\{f_u^{(i)}(a, b), f_v^{(0)}(a, b)\} \]

- Why use skipping bigram?
  - Consider the structural variations in cover songs
  - A simple example: consider two code sequences $\{1, 2, 3\}$ and $\{1, 3\}$, the similarity of bigram histogram is zero
  - Consider a gap $s$ when constructing bigrams
Inverted index

- How to compute the similarity efficiently

\[ S(u, v) = \max_i \sum_{a,b} \min\{f_u^{(i)}(a, b), f_v^{(0)}(a, b)\} \]

- A table is established to maintain the mapping from \((a, b)\) to recording.

- Given a pair \((a, b)\), we could get

\[ \{(v, f_v^{(0)}(a, b)) | f_v^{(0)}(a, b) > 0\} \] quickly with the help of the table.
Retrieval

- Given a query $u$, code sequences are generated through embedding, transposition and encoding.

- Fixed $i$, for each bigram $(a, b) \in \{ (a, b) \mid f_u^{(i)}(a, b) > 0 \}$, we find $\{(v, f_v^{(0)}(a, b)) \mid f_v^{(0)}(a, b) > 0 \}$ with the help of table.

- Enumerating $i \in \{-5, -4 \ldots 5\}$, the algorithm computes the similarity between the query and the reference.
Experimental setting

- **Evaluation metric**
  - Mean average precision (MAP)
  - Precision at 10 (P@10)
  - Mean rank of first correctly identified cover (MR1)

- **Datasets**
  - Youtube350
  - Music collection (MC)
Influence of hyperparameters

- Resample CENS sequences to simulate different speed
- Skipping bigrams help improve the precision
Influence of hyperparameters

- Resample CENS sequences to simulate different speed
- Skipping bigrams help improve the precision
Influence of hyperparameters

- Explore how many codes are needed to ensure good performance
- Sub-linear relationship between N and K
Influence of hyperparameters

- Explore how many codes are needed to ensure good performance
- Sub-linear relationship between N and K
Comparison

- Highest P@10 and MR1 compared to state-of-the-art method
- Low time complexity

<table>
<thead>
<tr>
<th></th>
<th>MAP</th>
<th>P@10</th>
<th>MR1</th>
<th>Time/s</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTW [19]</td>
<td>0.425</td>
<td>0.114</td>
<td>11.69</td>
<td>56.50</td>
<td>$O(NM^2)$</td>
</tr>
<tr>
<td>Silva et al. [19]</td>
<td>0.478</td>
<td>0.126</td>
<td>8.49</td>
<td>3.71</td>
<td>$O(NMS)$</td>
</tr>
<tr>
<td>Serra et al. [21]</td>
<td>0.525</td>
<td>0.132</td>
<td>9.43</td>
<td>2419.20</td>
<td>$O(NM^2)$</td>
</tr>
<tr>
<td>Silva et al. [18]</td>
<td>0.591</td>
<td>0.140</td>
<td>7.91</td>
<td>18.72</td>
<td>$O(NM \log M)$</td>
</tr>
<tr>
<td>Rafii CQT [22]</td>
<td>0.521</td>
<td>0.122</td>
<td>9.75</td>
<td>-</td>
<td>$O(NM^2)$</td>
</tr>
<tr>
<td>Rafii fingerprint [22]</td>
<td>0.648</td>
<td>0.145</td>
<td>8.27</td>
<td>-</td>
<td>$O(NM^2)$</td>
</tr>
<tr>
<td>Skipping bigrams</td>
<td>0.617</td>
<td><strong>0.147</strong></td>
<td><strong>7.42</strong></td>
<td><strong>3.40</strong></td>
<td>$O(M \log K)$</td>
</tr>
</tbody>
</table>
Conclusion & Future work

- Propose a skipping bigram model robust against structure and speed variations
- Design an inverted index for acceleration
- Achieve a high MAP with low time cost on a recent cover song dataset
- Adapt our approach to large-scale datasets
Thank you!