EFFECTIVE COVER SONG IDENTIFICATION BASED ON SKIPPING BIGRAMS

Xiaoshuo Xu, Xiaoou Chen, Deshun Yang

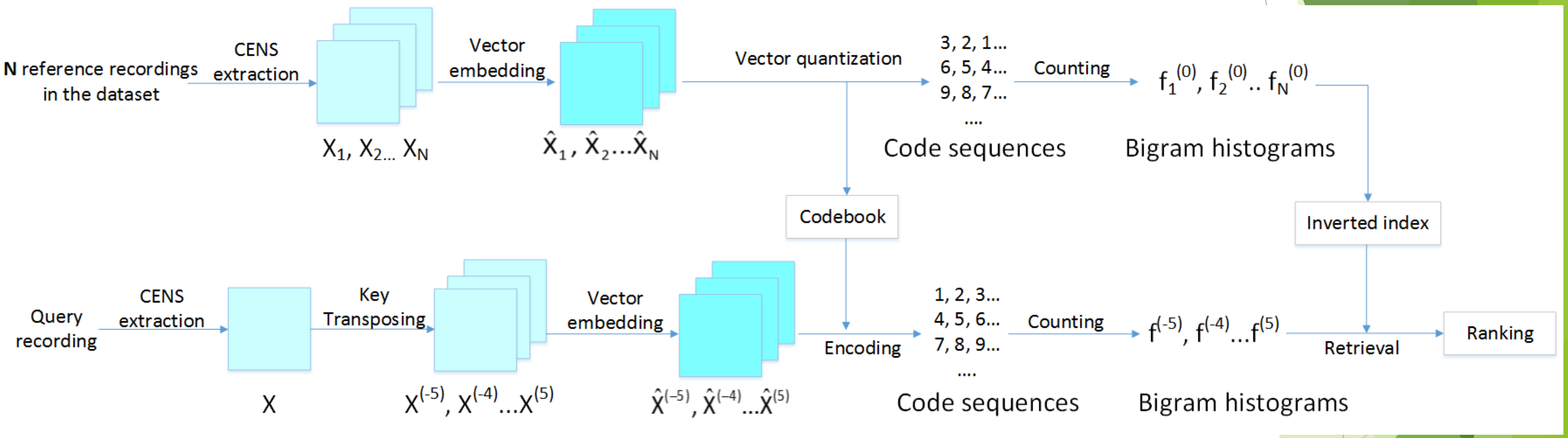
Institute of Compute Science and Technology, Peking University

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



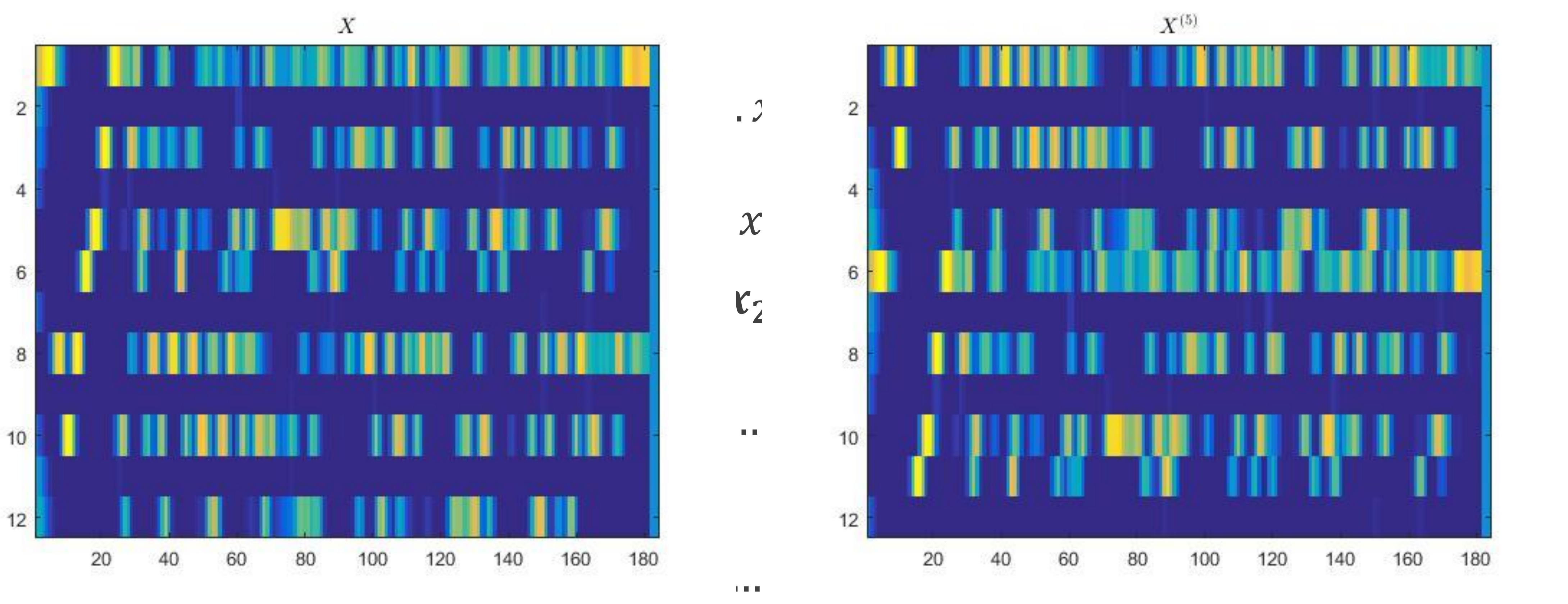
N reference recordings in the dataset
CENS extraction
X1, X2... XN
Vector embedding
X̂1, X̂2...X̂N
Vector quantization
3, 2, 1...
6, 5, 4...
9, 8, 7...
....
Code sequences
Counting
f1(0), f2(0).. fN(0)
Bigram histograms
Codebook
Inverted index
Query recording
CENS extraction
X
Key Transposing
X(-5), X(-4)...X(5)
Vector embedding
X̂(-5), X̂(-4)...X̂(5)
Encoding
1, 2, 3...
4, 5, 6...
7, 8, 9...
....
Code sequences
Counting
f(-5), f(-4)...f(5)
Bigram histograms
Retrieval
Ranking

Feature extraction

- Chroma Energy Normalized Statistics (CENS)
- Key transposition
  - Given a CENS vector $\boldsymbol{x} = (x_0, x_1 \dots x_{11})^T$, the transposed vector is defined as follows:
$$x^{(i)} = (x_{i\%12}, x_{(i+1)\%12} \dots x_{(i+11)\%12})^T$$
  - Given a CENS sequence $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2} \dots \boldsymbol{x_M}]$, the transposed sequence would be:
$$\boldsymbol{X}^{(i)} = [\boldsymbol{x}_{\boldsymbol{1}}^{(i)}, \boldsymbol{x}_{\boldsymbol{2}}^{(i)} \dots \boldsymbol{x}_{\boldsymbol{M}}^{(i)}]$$
- Vector Embedding
  - Embedded vector: $\widehat{\boldsymbol{x}_j} = \left[\boldsymbol{x}_{\boldsymbol{j}}^{\boldsymbol{T}}, \boldsymbol{x}_{\boldsymbol{j-1}}^{\boldsymbol{T}} \dots \boldsymbol{x}_{\boldsymbol{j-(m-1)}}^{\boldsymbol{T}}\right], j = m, m+1 \dots M$
  - Embedded sequence: $\widehat{\boldsymbol{X}} = [\widehat{\boldsymbol{x_m}}, \widehat{\boldsymbol{x_{m+1}}} \dots \widehat{\boldsymbol{x_M}}]$
  - Transposed embedded sequence: $\widehat{\boldsymbol{X}}^{(i)}$

Feature extraction

- Chroma Energy Normalized Statistics (CENS)



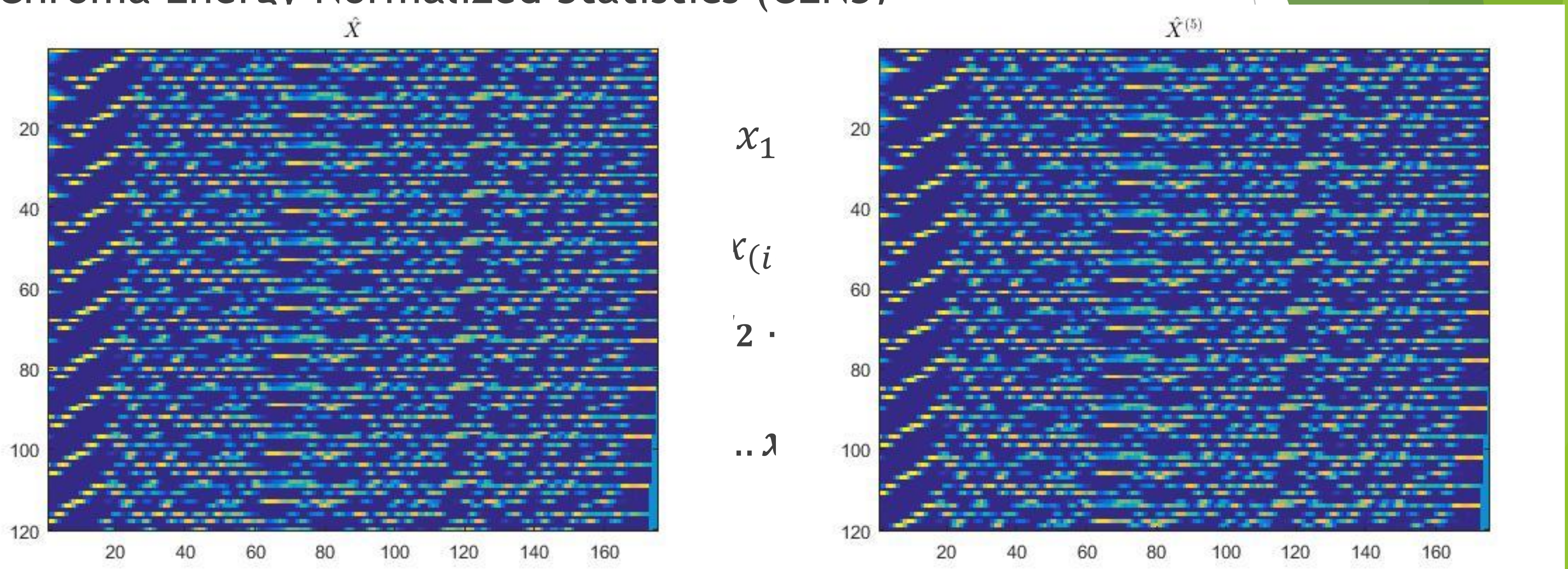
  - Embedded sequence: $\widehat{\boldsymbol{X}} = [\widehat{\boldsymbol{x}_m}, \widehat{\boldsymbol{x}_{m+1}} \dots \widehat{\boldsymbol{x}_M}]$
  - Transposed embedded sequence: $\widehat{\boldsymbol{X}}^{(i)}$

Feature extraction

- Chroma Energy Normalized Statistics (CENS)
- Key transposition
  - Given a CENS vector $\boldsymbol{x} = (x_0, x_1 \dots x_{11})^T$, the transposed vector is defined as follows:
    $$x^{(i)} = (x_{i\%12}, x_{(i+1)\%12} \dots x_{(i+11)\%12})^T$$
  - Given a CENS sequence $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2} \dots \boldsymbol{x_M}]$, the transposed sequence would be:
    $$\boldsymbol{X}^{(i)} = [\boldsymbol{x}_{\mathbf{1}}^{(i)}, \boldsymbol{x}_{\mathbf{2}}^{(i)} \dots \boldsymbol{x}_{\boldsymbol{M}}^{(i)}]$$
- Vector Embedding
  - Embedded vector: $\widehat{\boldsymbol{x}_j} = \left[\boldsymbol{x}_j^T, \boldsymbol{x}_{j-1}^T \dots \boldsymbol{x}_{j-(m-1)}^T\right], j = m, m$+1...M
  - Embedded sequence: $\widehat{\boldsymbol{X}} = [\widehat{\boldsymbol{x}_m}, \widehat{\boldsymbol{x}_{m+1}} \dots \widehat{\boldsymbol{x}_M}]$
  - Transposed embedded sequence: $\widehat{\boldsymbol{X}}^{(i)}$

Feature extraction

- Chroma Energy Normalized Statistics (CENS)

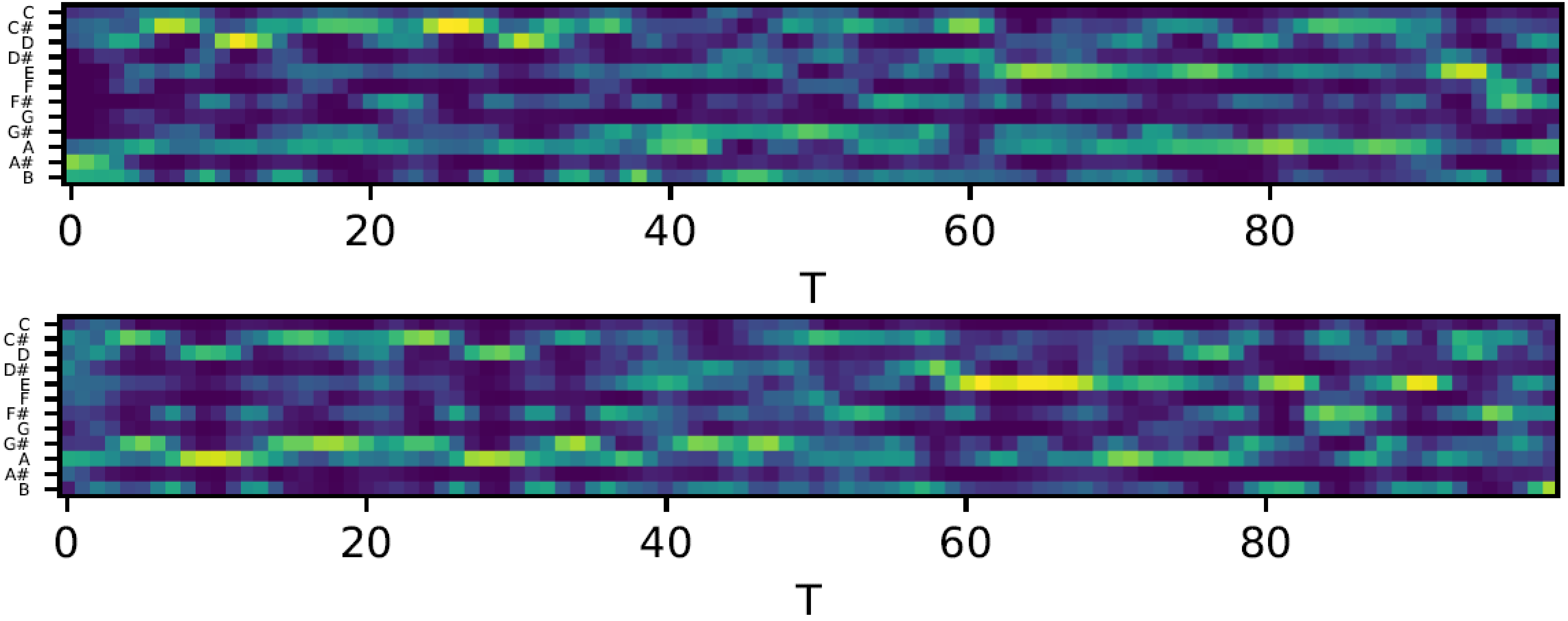


- Embedded vector: $\vec{x_j} = [x_j, x_{j-1} \dots x_{j-(m-1)}], j = m, m+1 \dots M$
- Embedded sequence: $\hat{X} = [\widehat{x_m}, \widehat{x_{m+1}} \dots \widehat{x_M}]$
- Transposed embedded sequence: $\hat{X}^{(i)}$

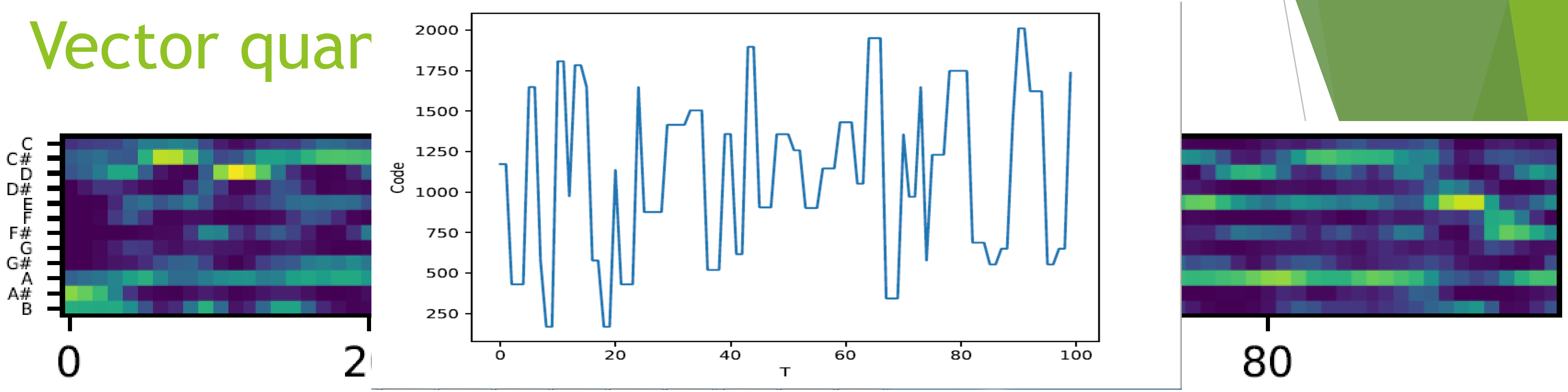
Vector quantization and encoding

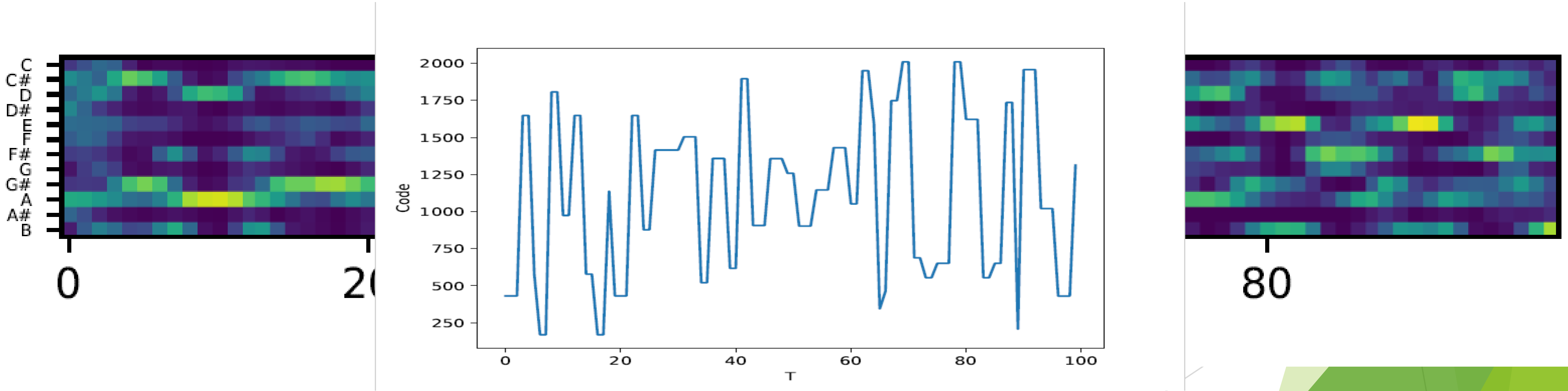
- Vector quantization is used to cluster embedded vectors and a codebook is learnt for encoding.
- Reduce the impact of structural variations.
- Code sequences of cover songs reveal high similarity, while code sequences of different songs show little similarity.

Vector quantization and encoding



Vector quar


C
C#
D
D#
E
F
F#
G
G#
A
A#
B
0
80
2000
1750
1500
1250
1000
750
500
250
Code
0
20
40
60
80
100
T


C
C#
D
D#
E
F
F#
G
G#
A
A#
B
0
80
2000
1750
1500
1250
1000
750
500
250
Code
0
20
40
60
80
100
T

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 $(\mathrm{a}, b) \in \left\{(a, b) \middle| f_u^{(i)}(a, b) > 0\right\}$, we find $\{(v, f_v^{(0)}(a, b)) | f_v^{(0)}(a, b) > 0\}$ 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

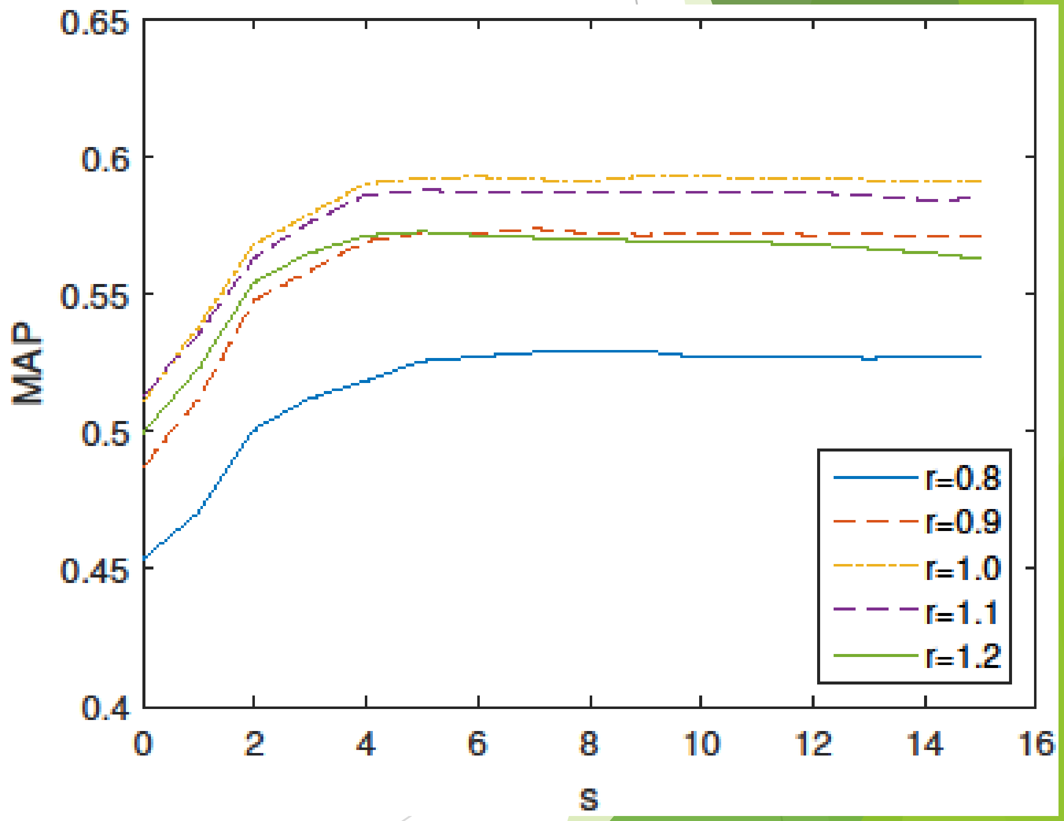
- Evaluation metric
  - Mean average precision (MAP)
  - Precision at 10 (P@10)
  - Mean rank of first correctly identified cover (MR1)
- Datasets
  - Youtube350
  - Music collection (MC)

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

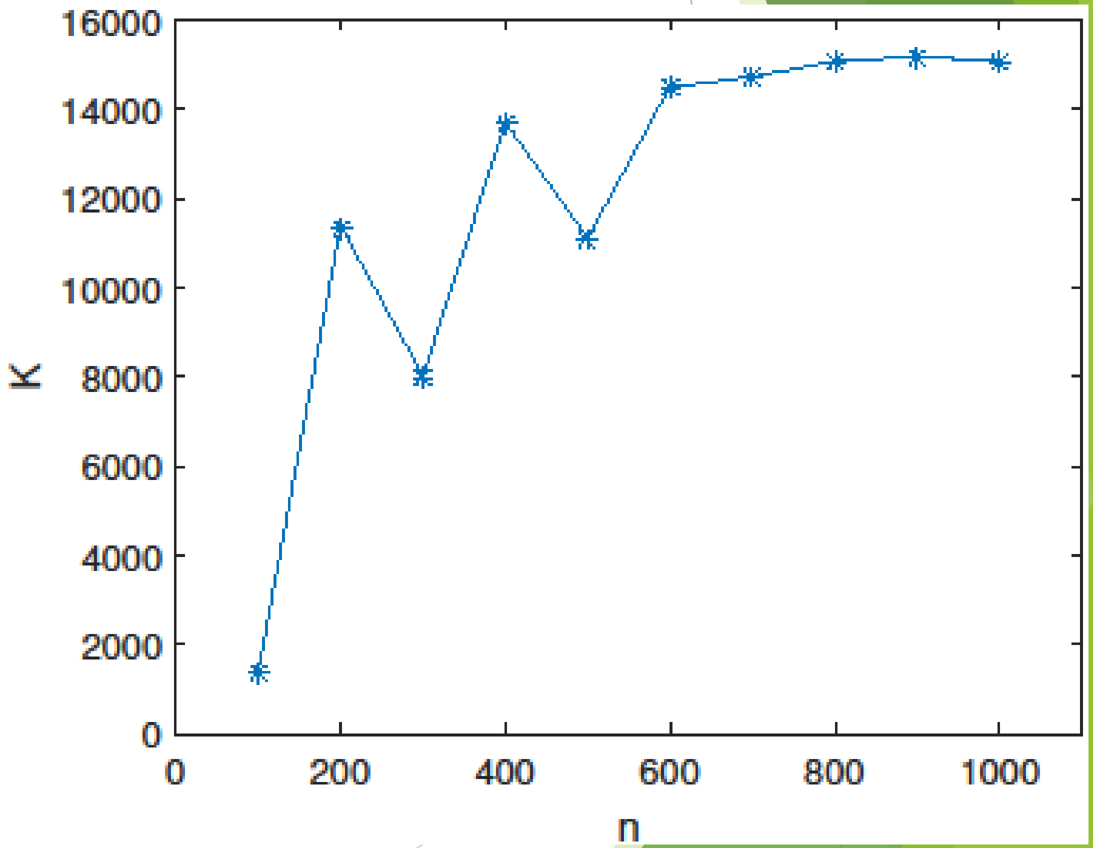


Influence of hyperparameters

- 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

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

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!