MIREX 2018 Submission: A Structural Chord Representation for Automatic Large-Vocabulary Chord Transcription

Junyan Jiang
Fudan University
jiangjy14@fudan.edu.cn

Ke Chen
Fudan University
kchen15@fudan.edu.cn

Wei Li
Fudan University
weili-fudan@fudan.edu.cn

Guangyu Xia
New York University Shanghai
gxia@nyu.edu

ABSTRACT

In this extended abstract, we propose a new model for practical chord transcription task. The core concept of the new model is to represent any chord label by a set of subparts (i.e., root, triad, bass) according to their common musical structures. A multitask classifier is then trained to recognize all the subparts given the audio feature, and then labels of individual subparts are reassembled to form the final chord label. A Recurrent Convolutional Neural Network (RCNN) is used to build the multitask classifier.

1. SYSTEM OVERVIEW

Submission ID	Algorithm	Training & Validation Sets
JLCX1	Proposed (no beat alignment, w/o weighted loss)	Isophonics, Billboard (public part), RWC Pop, and MARL collections
JLCX2	Proposed (with beat alignment, w/o weighted loss)	

Table 1. Summary of submitted systems

Large-vocabulary chord transcription is a difficult task, as the number of chord classes is large and the distribution of training chord classes is extremely biased. To solve the problem, we avoided training a classifier of chord labels directly. Instead, we decompose a chord into several components, each of which can be recognized more easily.

It is important that such decomposition can be applied universally to most chord types, and the properties of target components are useful for automatic chord transcription. We briefly introduce the chord representation in Section 1.1 and discuss how it can be utilized for chord transcription in Section 1.2 and 1.3.

1.1 Structural Chord Representation

The note combination of a chord may vary, yet some regular patterns can be detected in most chords. For example, the 7th note of a chord, if exists, is most likely to be one among `7`, `b7` and `bb7` (in `dim7` chords), and these three notes are mutually exclusive in a chord. Such properties are helpful when we decompose any chord into a concise form.

We represent each chord as the following tuple:

$$\text{Chord} = (\text{Root}, \text{Triad}, \text{Bass}, \text{Seventh}, \text{Ninth}, \text{Eleventh}, \text{Thirteenth}) \quad (1)$$

where every component corresponds to a set of possible values according to their musical meaning. Specifically:

$$\begin{aligned}
&\text{Root} \leftarrow \{\text{N}, \text{C}, \text{C}\#, \text{D} \dots, \text{B}\} \\
&\text{Triad} \leftarrow \{\text{X}, \text{N}, \text{maj}, \text{min}, \text{sus4}, \text{sus2}, \text{dim}, \text{aug}, 5, \dots\} \\
&\text{Bass} \leftarrow \{\text{N}, \text{C}, \text{C}\#, \text{D} \dots, \text{B}\} \\
&\text{Seventh} \leftarrow \{\text{X}, \text{N}, 7, \text{b}7, \text{bb}7\} \\
&\text{Ninth} \leftarrow \{\text{X}, \text{N}, 9, \#9, \text{b}9\} \\
&\text{Eleventh} \leftarrow \{\text{X}, \text{N}, 11, \#11\} \\
&\text{Thirteenth} \leftarrow \{\text{X}, \text{N}, 13, \text{b}13\}
\end{aligned}$$

where `X` means *not applicable*, and `N` means *none/not present*. The `5` in Triad category denotes the power chord (e.g. `C:5` is equivalent to `C:(1,5)`).

The `X` values are for special situations that some of the components are undecidable or meaningless for certain chords. For example, in a chord whose basic triad type is `sus4`, the 11th note has the same scale as the 4th note, so talking about added-11th would be meaningless under this situation. Components with value `X` are ignored when calculating the joint probability $p(\text{Chord})$.

Another note is that some modern popular music sheets annotate `C:(1,3,5,6)` as `C:maj6`. We here regard the added-6th note as an equivalent form of an added-13th note.

Table 2 shows our representation for some common chords.

Chord	Root	Triad	Bass	7th	9th	11th	13th
G:maj	G	maj	G	N	N	N	N
G:maj7	G	maj	G	7	N	N	N
G:7(b9)	G	maj	G	b7	b9	N	N
G:min7/b3	G	min	Bb	b7	N	N	N
B:hdim7	B	dim	B	b7	N	N	N
A:sus4(7)	A	sus4	A	b7	N	X	N
C:9(13)	C	maj	C	b7	9	N	13
N	N	N	N	N	N	N	N

Table 2. Some examples of the chord representation method

1.2 A Probabilistic Model for Chord Transcription

For automatic chord transcription, we are interested in $p(\mathrm{Chord}|\mathrm{Feature})$ where Feature represents the extracted audio feature around a certain frame, and Chord denotes the chord label for the corresponding frame. We assume that some certain components of a chord are conditional independent from each other given the audio feature, and rewrite the formula as:

$$\begin{aligned} & p(\mathrm{Chord}|\mathrm{Feature}) \\ = & p(\mathrm{Root}, \mathrm{Triad}, \mathrm{Bass}, \mathrm{Seventh}, \mathrm{Ninth}, \mathrm{Eleventh}, \mathrm{Thirteenth}|\mathrm{Feature}) \\ = & p(\mathrm{Root}, \mathrm{Triad}|\mathrm{Feature}) p(\mathrm{Bass}|\mathrm{Feature}) p(\mathrm{Seventh}|\mathrm{Root}, \mathrm{Feature}) \\ & p(\mathrm{Ninth}|\mathrm{Root}, \mathrm{Feature}) p(\mathrm{Eleventh}|\mathrm{Root}, \mathrm{Feature}) \\ & p(\mathrm{Thirteenth}|\mathrm{Root}, \mathrm{Feature}) \end{aligned} \quad (2)$$

The dependency of these random variables can be summarized into a Bayesian network as shown in Figure 1.

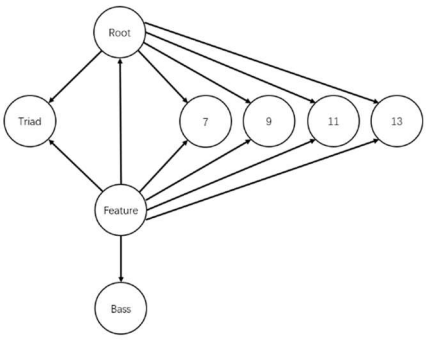

Figure 1. The graphical model of chord structure for automatic chord transcription

As the target probability is factored into six terms, each term can be modeled by a classifier.

1.2.1 Feature Extraction

A Constant-Q Transform (CQT) spectrogram is used as the input feature of the model. We calculate the CQT spectrogram with a sample rate of 22050 and a hop length of 512. 252 CQT filter banks are used ranging from midi note C1 to C7.

1.2.2 Model Architecture

We adopted a typical Recurrent Convolutional Neural Network (RCNN) architecture to build a multitask classifier, with Long Short-Term Memory (LSTM) cells as the recurrent layer unit. The model structure is shown in Table 3.

1.2.3 Training Methods

When training the model, we used the Adam Optimizer [1] with decreasing learning rate from 1e-3, 1e-4 to 1e-5. The model was trained 60 epochs in total. During each epoch, every training song (and its pitch-shifted version as augmented data) is iterated once and a 1000-frame (about 23 seconds) piece will be randomly selected from each song.

Layer Type	Parameters
Convolution	16×3×3
Convolution	16×3×3
Convolution	16×3×3
Max Pooling	3×3
Convolution	32×3×3
Convolution	32×3×3
Convolution	32×3×3
Max Pooling	3×3
Convolution	64×3×3
Convolution	64×3×3
Max Pooling	3×4
Bi-LSTM	128×2
Fully Connected	145

Table 3. RCNN model structure. A rectifier activation function is applied after each convolution layer, followed by a batch normalization layer. The stride of the window in any pooling layer is $(1, b)$ where b is the second dimension of its kernel size.

Output Dim.		Size	Function
1-25	1	1	$p(\mathrm{N}|\mathrm{Feature})$
	2-13	12	$p(\mathrm{Root}, \mathrm{Triad} = \mathrm{maj}|\mathrm{Feature})$
	14-25	12	$p(\mathrm{Root}, \mathrm{Triad} = \mathrm{min}|\mathrm{Feature})$
26-37		12	$p(\mathrm{Bass}|\mathrm{Feature})$
38-73		12×3	$p(\mathrm{Seventh}|\mathrm{Root}, \mathrm{Feature})$
74-97		12×2	$p(\mathrm{Ninth}|\mathrm{Root}, \mathrm{Feature})$
98-121		12×2	$p(\mathrm{Eleventh}|\mathrm{Root}, \mathrm{Feature})$
122-145		12×2	$p(\mathrm{Thirteenth}|\mathrm{Root}, \mathrm{Feature})$

Table 4. Network output explanation. Note that every different root uses a different output for seventh, ninth, eleventh and thirteenth components classification

The loss of the network is defined as the total loss of each classification subtask. For any of the seventh, ninth, eleventh and thirteenth components, every different root (except `N`) uses a different output for classification, so the loss will be only calculated on the classification output linked with the given root (except `N`). Additionally, if any component has a value `X`, the component will not contribute to the loss.

We found in early experiments that some uncommon decoration components (e.g. 9^{th}, 11^{th} ,13^{th}) are hard to train because of unbalanced training data, as only a small proportion of training data has such decorations. To overcome the problem, the weighted loss strategy can be applied to these components. For example, recognizing an add-9^{th} note as no decoration (`N`) would produce significantly higher loss than recognizing no decoration as an add-9^{th} note.

1.2.4 Inference

After the model is trained, we can predict a sequence of chord emission probability given the music feature. Instead of directly decode the chord with maximum joint probability, we first decode the chord with the maximum joint probability of roots, triads and basses, then the most likely decorations (seventh, ninth, eleventh, thirteenth) of the chords are decoded.

In practice we found another problem: as LSTM is intrinsically still a frame-wise classification algorithm, it sometimes suffers from an issue of frequent label change. To solve the problem, we mimicked the use of smoothing Hidden Markov Model (HMM) here to suppress frequent chord transition. Formally, we use dynamic programming to find the best sequence $c_1 \dots c_n$ that maximizes the cost J_1,J_2 in turn:

$$J_1(c_1, \dots, c_n|\text{Feature}) = \sum_{i=1}^{n} \log p(c_i|\text{Feature}) - \sum_{i=1}^{n-1} \alpha \cdot [c_i \neq c_{i+1}] \quad (3)$$

$$J_2(d_1, \dots, d_n|\text{Feature}, c_1, \dots, c_n) = \sum_{i=1}^{n} \log p(d_i|\text{Feature}, c_i) - \sum_{i=1}^{n-1} \beta \cdot [d_i \neq d_{i+1} \wedge c_i = c_{i+1}] \quad (4)$$

where c_i denotes the decoded triad chords and basses for the i-th frame and d_i denotes the decoded decorations for the i-th frame. The best values of transition penalty α, β are acquired by performing a grid search over the training set.

Moreover, if the beat positions of a music piece are provided, it is more likely that a chord transition happens on the beat position. Our second submission restricted the chord transition to align with beat positions, which are provided by DBNDownBeatTracker [4] from the Madmom python package.

2. EXPERIMENTS

2.1 Datasets

We used 1217 songs from Isophonics, Billboard, RWC Pop, and MARL collections collected by Humphrey and Bello [2, 3] to form the dataset. All chord labels were first converted from string to our proposed representation. For MIREX evaluation, we further selected a subset of the components as valid values for training, and marked other values as an `X`:

$$\text{Triad} \leftarrow \{\text{X}, \text{N}, \text{maj}, \text{min}\}$$
$$\text{Seventh} \leftarrow \{\text{X}, \text{N}, 7, \text{b7}\}$$
$$\text{Ninth} \leftarrow \{\text{X}, \text{N}, 9\}$$
$$\text{Eleventh} \leftarrow \{\text{X}, \text{N}, 11\}$$
$$\text{Thirteenth} \leftarrow \{\text{X}, \text{N}, 13\}$$

Data augmentation was done by performing pitch-shifting operations (from -5 semitones to +6 semitones) on the training set. The annotated roots and basses are shifted accordingly.

2.2 Model Evaluation

We performed 4-fold cross-validation over the dataset, with about 304 songs per subset. Three subsets are used for training the model, and the other is used for testing. No data augmentation is performed on the testing set.

We calculated the WAOR (Weighted Average Overlap Ratio) scores [6] on typical metrics for chord estimation with python package mir_eval. Table 5 shows the results.

Metrics	[5]	P	P+B	P+W	P+W+B
Root	0.7660	**0.8377**	0.8364	0.8368	0.8342
MajMin	0.7449	**0.8305**	0.8293	0.8298	0.8273
MajMinBass	0.7151	**0.8099**	0.8086	0.8090	0.8062
Sevenths	0.5571	**0.7095**	0.7081	0.7016	0.6992
SeventhsBass	0.5338	**0.6956**	0.6942	0.6873	0.6847

Table 5. Model evaluation results. P denotes our proposed model, B denotes that beat alignment is adopted, and W denotes that weighted loss is applied for uncommon decorations (2.5 times for `7` and `b7`, 5 times for `9`, `11` and `13`) in training

Furthermore, to evaluate the model performance on large-vocabulary chord transcription, a confusion matrix on chord types for the RCNN model is presented.

The confusion matrix is calculated as:

$$\phi_{i,j} = \frac{\sum_{(est,ref)\in D} \delta_{i,j}(est,ref)}{\sum_k \sum_{(est,ref)\in D} \delta_{i,k}(est,ref)} \quad (5)$$

where D denotes all pairs of estimated chords and reference chords on the same frame. $\phi_{i,j}$ denotes the confusion degree of chord type i and j. Boolean value $\delta_{i,j}(c_1, c_2) = 1$ if and only if

(1) c_1 and c_2 are in the same root (here we assume chord `N` shares the same root with any chord), and

(2) chord c_1 has chord type i while chord c_2 has chord type j.

The result is shown in Figure 2 and Figure 3.

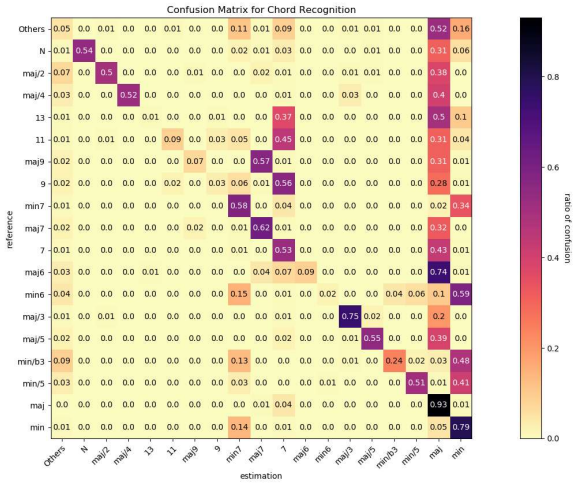

Figure 2. Confusion matrix on the testing set (model P)

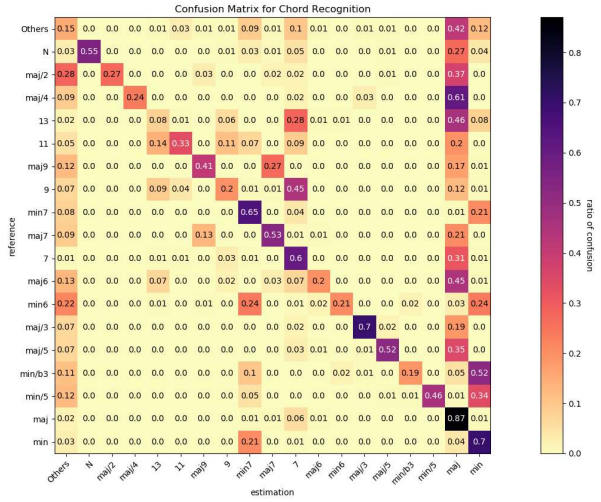

Figure 3. Confusion matrix on the testing set (model P+W)

We found in the results that both models perform well on triads and their inversions. The model with a weighted loss on training outperforms the other on recognition of uncommon decorations like `9`, `11` and `13`, which is in line with our expectations.

3. REFERENCES

[1] D. Kinga and J. B. Adam: "A method for stochastic optimization," *International Conference on Learning Representations (ICLR)*, Vol.5, 2015.

[2] B. McFee and J. P. Bello: "Structured training for large-vocabulary chord recognition," *ISMIR*, 2017.

[3] E. J. Humphrey and J. P. Bello: "Four Timely Insights on Automatic Chord Estimation," *ISMIR*, pp. 673–679, 2015.

[4] S. Böck, F. Krebs, and G. Widmer: "Joint Beat and Downbeat Tracking with Recurrent Neural Networks," *ISMIR*, pp. 255–261, 2016.

[5] M. Mauch and S. Dixon: "Approximate Note Transcription for the Improved Identification of Difficult Chords," *ISMIR*, pp. 135–140, 2010.

[6] J. Pauwels and G. Peeters: "Evaluating automatically estimated chord sequences," *Acoustics, Speech and Signal Processing (ICASSP)*, pp. 749–753, 2013.