

FYP Final Report
KY1801
Automatic piano reduction-Chord
Identification

Supervisor: Kevin Yip

Supervisor: Lucas Wong

Student: Li Shuhe(1155062051)

Student: ZHANG Zhaochen(1155062190)

Content

1.	Introduction	4
1.1	Objective.....	4
2.	Music Theory	4
2.1	Notes	4
2.2	Pitch	5
2.3	Interval.....	5
2.4	Accidentals	5
2.5	Beat	6
2.6	Meter	6
2.7	Measure	7
2.8	Time signature.....	7
2.9	Key signature.....	7
2.10	Chord.....	8
2.11	Roman numeral analysis	8
2.12	Key	8
2.13	Tonality	9
3.	Previous works.....	9
3.1	Last semester	9
4.	Theory.....	9
4.1	Decision tree.....	9
4.2	Hidden Markov model.....	10
4.3	Long short-term memory.....	11
5.	Overall system	15
5.1	preprocessing module.....	16
5.1.1	Key Identification.....	16
5.1.2	Multiple Chords Division.....	19
5.1.3	Data Extraction	19
5.2	predicting module	20
5.3	evaluating module.....	22
6.	Implementation details	24
6.1	Predicting module	24
6.1.1	Decision tree	24
6.1.2	Hidden Markov model	24
6.1.3	Long Short-term memory	26
7.	Experiment and analysis	28
7.1	Assumption.....	28
7.2	Training Data	28
7.3	experiments	29
7.3.1	Pretrained model vs non-pretrained model.....	29
7.3.2	HMM vs LSTM	30
7.3.3	Duration vs appearance	31
8.	Contribution.....	33
8.1	Work done by Zhaochen ZHANG	33
9.	Conclusion	35
10.	Reference.....	35
11.	Appendix:	36

11.1	Data source:	36
11.2	Sample pretrain data:.....	36

1. Introduction

Music plays an important role in both professional and casual life. Throughout history, preeminent musicians have created innumerable musical masterpieces, which build the foundation of the music world. However, those masterpieces are restricted to be used because of the limitation of instruments. On one hand, people are not able to be master in all instruments. A musical score is useless for ordinary people if he is unfamiliar with those instruments. On the other hand, some scores involve dozens of uncommon instruments. For instance, there are four groups of related musical instruments in a symphony orchestra, which are woodwinds, brass, percussion, and strings. It is impossible for a single person to perform so many instruments simultaneously. Thus, a reduction tool to convert complicate music pieces into concise sheets is crucial and always be a popular topic.

According to Wikipedia, a piano reduction or piano transcription is sheet music for the piano that has been compressed and simplified to fit on a two-line staff and be playable on the piano. The goal of this project is to develop software to perform the piano reduction of classical music automatically.

Before reduction, chord identification is extremely important. The accuracy is firmly related to the result of the reduction. A chord, in music, is any harmonic set of pitches consisting of two or more notes that are sound simultaneously. Because of the ambiguity of chord in classical musical sheet, recognize and identify chord in a musical sheet is difficult and fuzzy. Even professional musicians may hold different views because of their unique understanding of music. Besides, chord identification by hand is extremely time-consuming. For a single musical score, a professional musician may need several hours to identify all the chord. Clearly, it is unrealistic to finish this work by hand.

1.1 Objective

The objective of the project is to build a system which is able to identify the chords in a score of musicXML format accurately and output the recognition result for each measure in the score.

2. Music Theory

2.1 Notes

In music, a note is the pitch and duration of a sound represented in musical notation. A note is the basic element of the music which is important in understanding and analyzing the music. There are 12 notes in music, usually presented in English letters with accidentals. There are seven letters C, D, E, F, G, A, and B, which represents different notes with different pitches. The notes table is as followed. There is also a special note named rest note, which has no pitch component. It means rest for such duration.

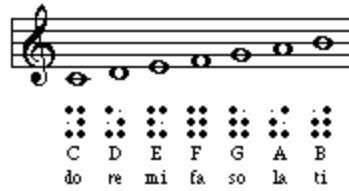


Figure 1.1

2.2 Pitch

Pitch is an attribute to represent the physic frequency of the note. Pitch is a perceptual property and decides how the note sounds like. In order to simplify and normalize the procedure of playing and composing, the standard pitch is decided. Nowadays, the notes representation is normalized, and you can easily understand which note it is in a score.

2.3 Interval

In order to accurately describe the difference of different pitch, the concept of the interval was created to solve this problem. An interval is a difference in pitch between two sounds. A semitone is the smallest interval. The following table records the common intervals used in music.

Number of semitones	Interval types	Short
0	Perfect unison	P1
1	Minor second	m2
2	Major second	M2
3	Minor third	m3
4	Major third	M3
5	Perfect fourth	P4
7	Perfect fifth	P5
8	Minor sixth	m6
9	Major sixth	M6
10	Minor seventh	m7
11	Major seventh	M7
12	Perfect octave	P8

2.4 Accidentals

Accidentals are the descriptive notation of the notes which modify the pitch of the note. There are total 5 accidentals, sharp sign \sharp , flat sign \flat , double-sharp double sharp, double-flat double flat, and natural sign \natural . The sharp sign raises a note by a semitone. On the contrary, the flat sign lowers a note by a semitone. Obviously, double-sharp or double-flat raises or

lowers two semitones. The natural symbol is different. The symbol is used to restore the notes with a flat or sharp symbol. It will recover the notes into the pitch before the sharp and flat. The chromatic scale is the most common scale used in music. It describes a total of twelve notes in an octave.

1	2	3	4	5	6	7	8	9	10	11	12
C	C sharp	D	D sharp	E	F	F Sharp	G	G sharp	A	A sharp	B
	D flat		E flat			G flat		A flat		B flat	

2.5 Beat

A note is a pitch continuing for a certain time period. Naturally, time becomes a crucial component. Same pitches with different time duration will be played and perceived differently. It is important to develop a rule for time counting. A beat is how the musician represents the duration in music. It is the basic unit of time. Beats are classified into different types according to the length of the time duration. The following table records the common types of note beats used in music.













REST NAME	REST SYMBOL	REST LENGTH	NOTE SYMBOL
Whole Note (<i>Semi-breve</i>)		4 beats	
Half Note (<i>Minim</i>)		2 beats	
Quarter Note (<i>Crotchet</i>)		1 beat	
8th Note (<i>Quaver</i>)		½ beat	
16th Note (<i>Semiquaver</i>)		¼ beat	
32nd Note (<i>Demisemiquaver</i>)		⅛ beat	

Figure 1.2

2.6 Meter

Meter refers to the regularly recurring patterns of beats with accents. The meter defines the number and type of beats in one measure. Please refer to the appendix1 for full meter table.

2.7 Measure

A measure is the higher level of time unit in music. A measure is comprised of a specific number of beats, in which each beat is represented by a particular note. According to the beats numbers and beat type, the duration of measure becomes different in music. The following image is an example of a measure.

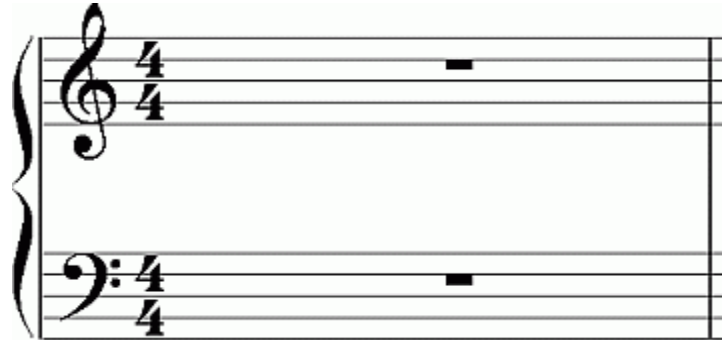


Figure 1.3

2.8 Time signature

A time signature is used to specify the beat number, beat value, and accent of the measure. The upper number defines the beat duration in the measure. For example, when it is 2, it means half note as a beat.

The lower number defines the beat number in a measure. For example, when it is 4, it means there are four beats in a measure. The time signature also implies the accent and we default set the first beat to downbeat, which is stronger than other beats.

Time Signature	Beat Duration	Number of Beats
$\frac{3}{2}$	$\frac{3}{2} =$	$\frac{3}{2} =$
$\frac{3}{4}$	$\frac{3}{4} =$	$\frac{3}{4} =$
$\frac{3}{8}$	$\frac{3}{8} =$	$\frac{3}{8} =$
$\frac{3}{16}$	$\frac{3}{16} =$	$\frac{3}{16} =$

Figure 1.4

2.9 Key signature

A key signature is a set of accidentals placed together on the staff. The key signature refers to notes in the score. The key signature designates notes played especially (sharp or flat) in a score. In addition to referring the special notes of the score, the key signature also indicates the mode and key of the score.



Figure 1.5

2.10 Chord

As a fundamental element of the music score, the chords play a very important role. The chord is any harmonic set of notes consisting of three or more that are heard simultaneously. Since there are 12 notes in music. Counting the combination of the notes, there are a number of chords. Fortunately, not all chords are meaningful in classical music. In western classical music, the most frequently encountered chords are triads. According to the numbers of the notes, the chords can be classified into different types, triad, seventh chord, ninth chord, eleventh chord, thirteenth chord, and so on. Triad contains three notes. The seventh chord contains four notes. The ninth chord and eleventh chord's notes number add in the same way. Since we only focus on classical music and the common chords, the number of chords is limited. For such reason, we only need to recognize some of the triad chord, seventh chord, and augmented sixth chord. According to the music interval between the different notes, the chords are classified into a major chord, minor chord, a dominant chord, augment chord, diminish chord and other types. For notes in a chord, the first note is most important, which is named root notes. The other notes can be calculated by the interval according to the chord type. Please refer to the appendix for all needed chords.

2.11 Roman numeral analysis

In classical music, a Roman numeral analysis was used to represent and analyze chords. The method uses Roman numerals to represent chords. The Roman numerals not only denote scale degrees of the chord's root note, but also denote the chord type. With this information provided with the key, the chord can be represented and understood by the musician. In addition to presenting chords' notes and types, the Roman numeral analysis also reveals the Tonality of the music.

2.12 Key

The key is the set of pitches of notes. The key is the basic idea of classical composition. The key usually identifies the tonic note. There are total of 24 keys in music. The key is the center notes of the set. Once the key is confirmed, we can deduce other notes according to the mode of the key.

2.13 Tonality

Tonality is a principle that the composer used to compose a score. It is the arrangement of the notes and chords in a reasonable way. Tonality is decided by key and the mode. The tonality is the rules that how the composer composes the music. The style and the feeling of music changes with the tonality. The tonality also provides the direction of music. Using different notes regularly in particular scales, the music will demonstrate strong direction feeling.

3. Previous works

3.1 Last semester

In the previous semester, we focus on chord identification with HMM Model. Due to the limitation of HMM Model, we tried to reduce the data dimension from 12 to 7 by divide the chords to different types and pre-recognize the chord types by some special notes. In addition to the HMM Model, we also did the data pre-processing to extract the data from scores. For each measure, we extract one vector of 7 dimensions as output and each dimension represent the fraction of the duration of one note in scale.

4. Theory

This section will illustrate decision tree, Hidden Markov model and Long short-term memory model, which will be used in our system. For each of them, the architecture of the model, predicting algorithm and learning strategy will be shown in detail.

4.1 Decision tree

Decision tree is a well-known flowchart-like structure in which each internal node represents a “test” on an attribute. The leaf node represents the clusters that satisfy the sequence of “test” in certain path. The model can be used for classification and has shown effective performance on decision selection problems.

Information gain is a measure that describes the change of entropies. It can be used to decide which feature to split. The detailed definition is shown below.

$$H(T) = I_E(p_1, p_2, \dots, p_J) = - \sum_{i=1}^J p_i \log_2 p_i$$

$$\begin{aligned} \overbrace{IG(T, a)}^{\text{Information Gain}} &= \overbrace{H(T)}^{\text{Entropy (parent)}} - \overbrace{H(T|a)}^{\text{Weighted Sum of Entropy (Children)}} \\ &= - \sum_{i=1}^J p_i \log_2 p_i - \sum_a p(a) \sum_{i=1}^J - \Pr(i|a) \log_2 \Pr(i|a) \end{aligned}$$

By choosing a variable at each step that best splits the set of items, the tree can be constructed easily. The variable is chosen so that the result tree has the highest information gain.

4.2 Hidden Markov model

If a system can be modeled as a Markov process and the direct state is invisible, we can use Hidden Markov Model to estimate the states sequence.

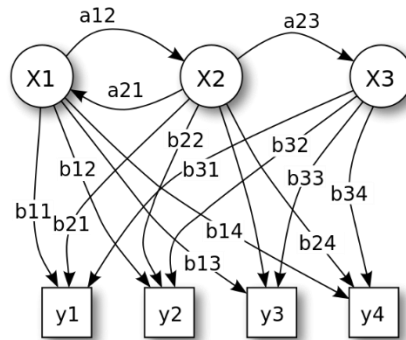


Figure 4.1

As illustrated in figure 2.2, X1, X2, and X3 are hidden states and the arrows between them describe the transition probabilities. Y1, Y2, Y3, Y4 are the observations. Hidden states are invisible and the observations are available. The arrows between hidden states and observations are called emission probability.

Given the parameters above, we can easily get the possible hidden state sequence based on observation sequence by forward algorithm, backward algorithm and Viterbi algorithm. In our model, we use Viterbi algorithm for prediction.

The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states. The pseudocode is illustrated below.

```

01. INPUT
02. O = {O1, O2, ..., ON} # observation space
03. S = {S1, S2, ..., Sk} # state space
04. Y = (Y1, Y2, ..., Yt) # sequence of observation
05. start probability SP of size 1 X K
06. transition probability TP of size K X K
07. emission probability EP of size K X N
08.
09. OUTPUT
10. X = (x1, x2, ..., xN) # most likely hidden state sequence
11.
12. function VITERBI(O, S, Y, SP, TP, EP): X
13.     for each state i in {1, 2, ..., K} do
14.         T1[i, 1] = SP[i] x EP[i, Y[1]]
15.         T2[i, 1] = 0
16.     end for
17.     for each observation i = 2, 3, ..., T do
18.         for each state j in {1, 2, ..., K} do
19.             T1[j, i] = max(T1[k, i-1] x TP[k, j] x EP[j, Y[i]])
20.             T2[j, i] = argmax(T1[k, i-1] x TP[k, j] x EP[j, Y[i]])
21.         end for
22.     end for
23.     zt = argmax(T1[k, T])
24.     xt = S[ZT]
25.     for i = T, T-1, ..., 2 do
26.         Z[i-1] = T2[Z[i], i]
27.         X[i-1] = S[Z[i-1]]
28.     end for
29.     return X
30. end function

```

Figure 4.2

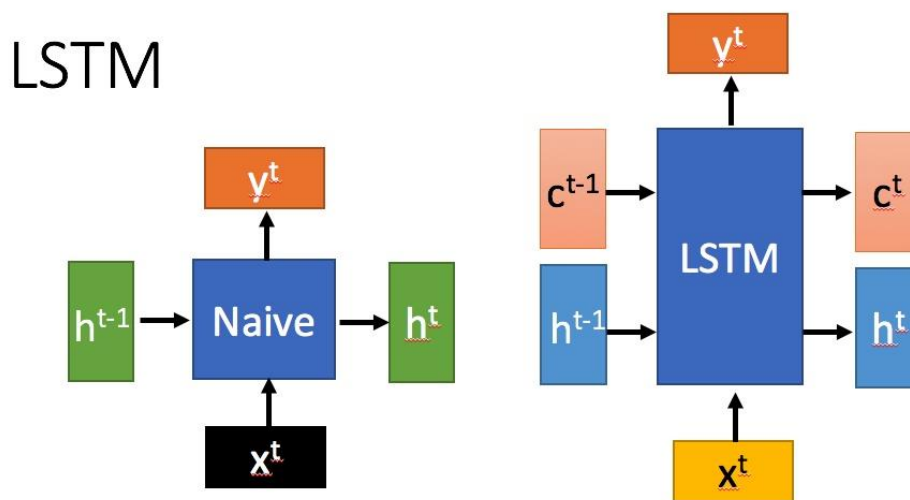
The algorithms will build two tables of size $k \times t$, where k is the number of states and t is the number of observation sequence. The algorithm runs in $O(T \times |S|^2)$.

Given observation sequence, HMM can be trained using Baum-Welch algorithm. In Baum-Welch algorithm, EM algorithm is used to find the maximum likelihood estimation. The algorithm contains two processes, forward and backward.

4.3 Long short-term memory

Before we talk about the Long short-term memory(LSTM), Recurrent neural network should be illustrated. A recurrent neural network(RNN) is a class of artificial neural network which can use their internal state to process sequences of inputs. The general architecture

of RNN can be found in the figure.



c change slowly $\Rightarrow c^t$ is c^{t-1} added by something

h change faster $\Rightarrow h^t$ and h^{t-1} can be very different

Figure 4.3

Normal RNN will use both input X and hidden state H to get output Y . Hidden state memorizes the previous input. Normally, y is calculated using sigmoid function.

$$y_t = \text{sigmoid}(h_t)$$

Problems of this architecture are gradient vanishing and gradient exploding. The model training is based on backpropagation. When we perform back propagation, we need to calculate the gradient of an error on weight. Error gradients can accumulate during an update and result in very large or very small gradients. An example of gradient vanishing and gradient exploding can be found below.

$$0.9^{20} = 0.1216$$

$$1.1^{20} = 6.7275$$

In order to solve this problem while maintaining the hidden state, LSTM was proposed. LSTM introduce gates to prevents back-propagated errors from vanishing or exploding. There is an input gate, an output gate and a forget gate in LSTM unit. The architecture of

LSTM unit can be found below.

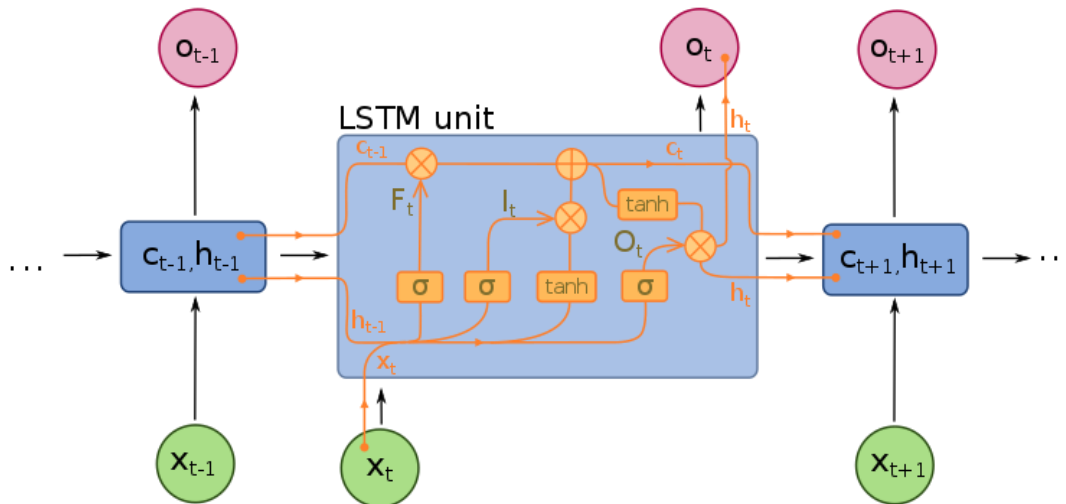
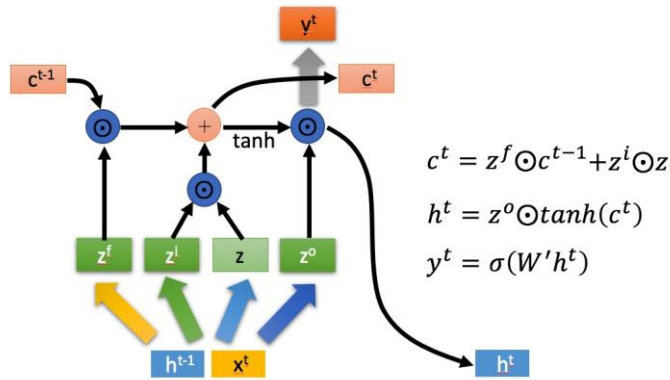


Figure 4.4

F_t is forget gate, I_t is input gate and O_t is output gate. Input gate controls the extent that the new value flows into the cell. Output gate controls the extent that the value is used to compute the final result. Forget gate control the extent that the value remains in the cell. Clearly, the output depends on input X , hidden state C and hidden state H . General speaking, C and H are both hidden states and are designed to remember the state of previous input. The definition of C and H are illustrated below. C corresponds to long memory and can change slowly. H corresponds to short memory and can change quickly.

The practical definition of gates can be found in the following figures.



$$z = \tanh\left(W \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix}\right)$$

$$z^f = \sigma\left(W^f \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix}\right)$$

$$z^i = \sigma\left(W^i \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix}\right)$$

$$z^o = \sigma\left(W^o \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix}\right)$$

Figure 4.5

The training of LSTM model is usually in a supervised fashion. On a set of training sequences (X, Y) , the parameters of the model can be tuned using gradient descent and hopefully to approximate the minimal of the error function.

LSTM has shown great performance in many applications. In speech recognition, handwriting recognition and music composition, it leads to the state of art. The advantages of LSTM motivate us to implement this algorithm for chord prediction.

5. Overall system

The system aims at predicting chords in the score. It should take a score as input and make a prediction. However, due to the characteristics in music, the prediction work is not easy. Firstly, the time unit of prediction is flexible. Time unit is a duration of score which is divided artificially. The system can only accept one chord in a time unit and it needs to figure out the best split rules for different scores. Last semester, we use measure as our time unit, which will affect our predicting accuracy and is not realistic. This semester, we propose a heuristic method to solve this problem. Then, some features may be missing because of musician's habits. Same notes may sound different in different environments. Human is able to figure out these unique features easily. But the computer needs some strategies for solving those problems.

Based on the problems mentioned above, we design an iterative system which is able to make the best prediction. The system consists of a preprocessing module, predicting module and evaluating module. As shown in the figure, the preprocessing module will extract the features from scores and pass them to the predicting module. The predicting module will make a prediction based on the recurrent model. After that, the evaluating module will check the prediction and decide whether the system should make a prediction again. If the evaluating module decides to do prediction again, the preprocessing module will make some changes according to the output of the evaluating module. Hopefully, the system will make the best prediction in this iterative processing. Unfortunately, if the prediction likelihood is not acceptable, the system goes back to preprocessing Module. For each bad prediction, there are totally three circumstances which might lead to bad prediction. Firstly, the key suddenly change. Since different chords might share the same notes and the notes represent different value in different keys. It is important to identify the key before identifying the chord. When key suddenly changes, for example, from C major to G major, the data extracted from the same notes will be different. Secondly, there might be several chords in a time unit. If there exists more than one chord in a time unit, it is difficult for Predicting Module to identify the chord. It is necessary to divide the time unit into smaller units with one chord for each smaller unit and predict for each unit. Thirdly, there might no chord in this time unit, the prediction is reasonable, and we need to remove this time unit. Evaluating module will check the prediction based on likelihood passed from predicting module. It will search the likelihood sequence and find the position that could be the reason for the bad likelihood. After that, it will notify the preprocessing module and tell preprocessing module this position. The preprocessing module gets the position and makes some adjustments. In this way, the system will search for the best solution for certain score. The details could be found in following sections.

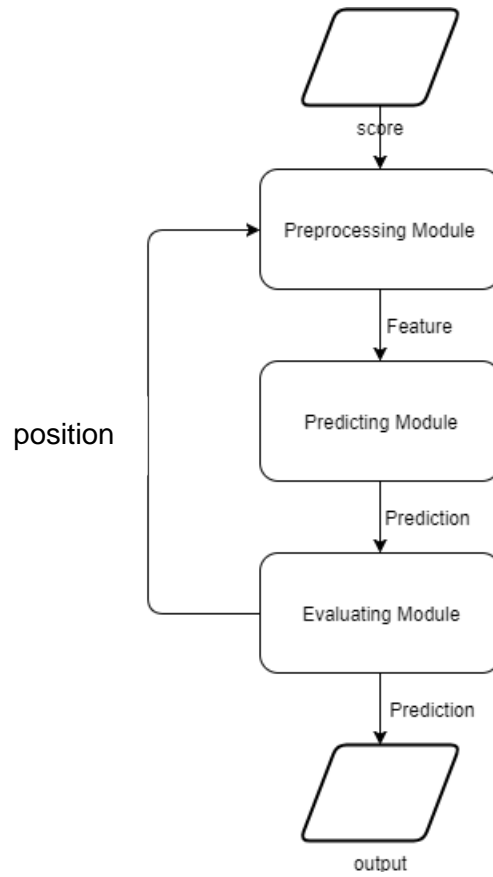


Figure 5.1

5.1 preprocessing module

In Pre-processing, the system needs to extract the data for predicting module from music scores. The original scores are in musicXML format. There are totally three steps to process the data, Key identification, Multiple chords division, Data Extraction. Key identification aims to identify and normalize the key information so that the key information will not affect the chords prediction. Multiple chords division aims to obtain a suitable time unit for predicting Module. The objective of data extraction is to obtain the trainable data and extract important features of the score.

5.1.1 Key Identification

In music, key transition is a common tool for musicians to make the music more flexible and vivid. The key transition does not change the probability of the chord prediction. However, it changes the notes the chords comprised of. It is necessary to identify the key and normalize the key. we use Krumhansl-Schmuckler key-finding algorithm to identify the key. After identification to key transition, we normalize the key information by scale all key to C. When the evaluating Module returns a bad likelihood, there might be a key transition in the score and the preprocessing Module identifies the current key and normalizes it to C.

Krumhansl-Schmuckler key-finding algorithm

- **Correlation coefficient**

Correlation coefficient c is a number from -1 to +1 which indicates the linear correlation between two variables. For $c < 0$, two variables have a negative linear correlation. For $c > 0$, two variables have a positive linear correlation. For $c = 0$, two variables have no linear correlation.

- **Key-finding algorithm**

The algorithm first calculates total durations for each note in the score. The following experience explains how algorithm decides the key of the score (Rebert Hart, 2012).

C	C#	D	D#	E	F	F#	G	G#	A	A#	B
432	231	0	405	12	316	4	126	612	0	191	1

total durations in "Toroko's Theme"

Then it pairs each note duration with two profiles for major key and minor key. The profile data are according to the experiments from Krumhansl and Edward J. Kessler.

do	do#	Re	re#	mi	fa	fa#	so	so#	la	la#	ti
6.35	2.23	3.48	2.33	4.38	4.09	2.52	5.19	2.39	3.66	2.29	2.88

Major profile

la	la#	Ti	do	do#	re	re#	mi	fa	fa#	So	so#
6.33	2.68	3.52	5.38	2.60	3.53	2.54	4.75	3.98	2.69	3.34	3.17

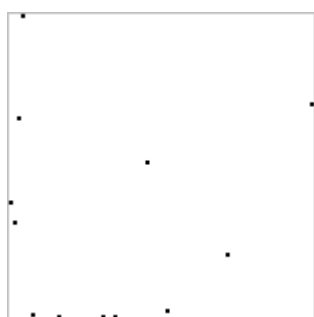
Minor profile

For each major and minor key, pair each note with different key scales. For example, the data pair of C major and C# major is followed. Since each key has a different scale, the relationship between notes durations and profile is different.

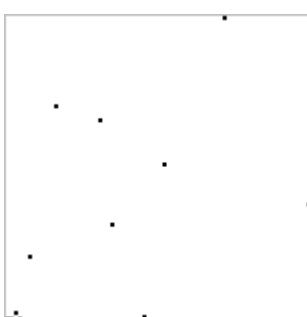
(do,C)	(6.35,432)	(do,C#)	(6.35,231)
(do#,C#)	(2.23,231)	(do#,D)	(2.23,0)
(re,D)	(3.48,0)	(re,D#)	(3.48,405)
(re#,D#)	(2.33,405)	(re#,E)	(2.33,12)
(mi, E)	(4.38,12)	(mi, F)	(4.38,316)
(fa,F)	(4.09,316)	(fa,F#)	(4.09,4)
(fa#,F#)	(2.52,4)	(fa#,G)	(2.52,126)

(so,G)	(5.19,126)	(so,G#)	(5.19,612)
(so#,G#)	(2.39,612)	(so#,A)	(2.39,0)
(la,A)	(3.66,0)	(la,A#)	(3.66,191)
(la#,A#)	(2.29,191)	(la#,B)	(2.29,1)
(ti,B)	(2.88,1)	(ti,C)	(2.88,432)
C Major		C# Major	

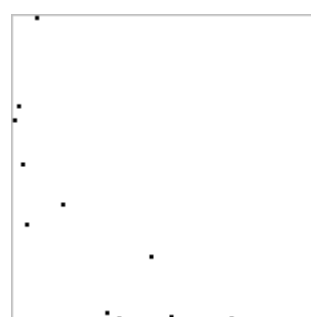
After calculating the correlation coefficient between the profile and durations for each key, we can get different results. For the key with the highest correlation coefficient is the most suitable key for the score. The result image is followed and the G# major has the highest correlation coefficient, the score is in G# major.



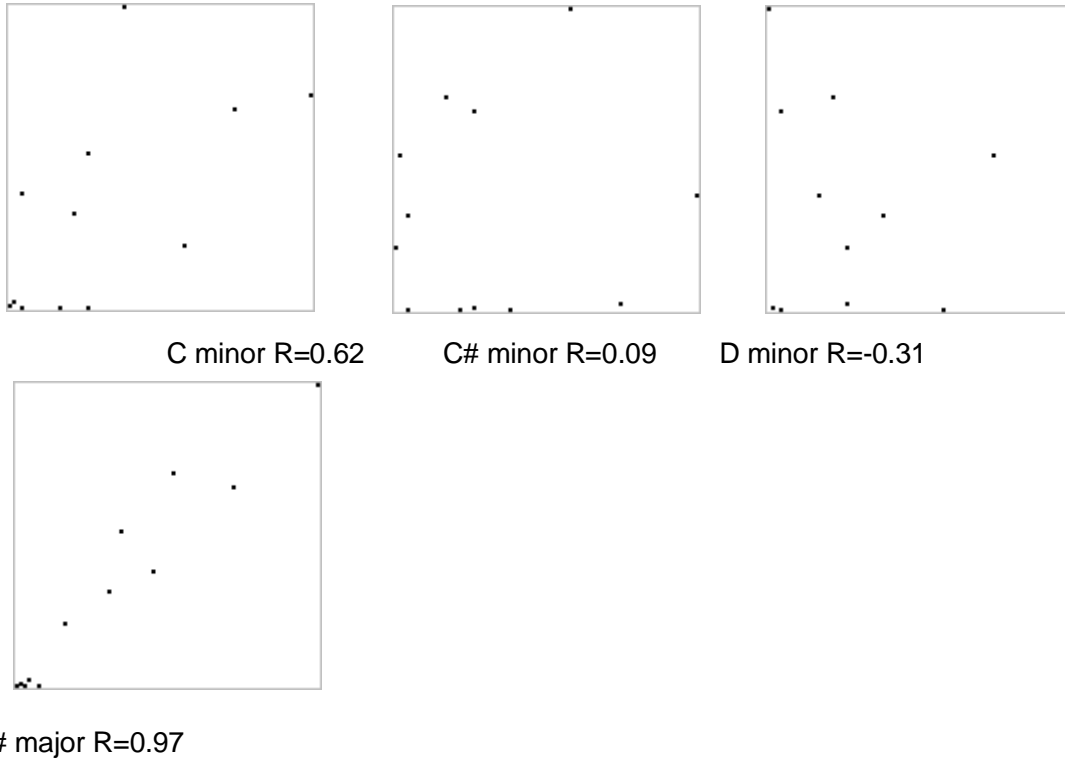
Cmajor R=-0.00009



C# major R=0.54



D major R=-0.74



5.1.2 Multiple Chords Division

In music score, it is common that there are several chords in a single measure. It is necessary to divide measure into small time units so that there is only one chord in a time unit. Firstly, the Module takes measure as a basic unit. For each unit, the Module divides it into small time units according to the beats. For example, if the beat of the measure is double beats, the Module divides the measure into two time units. Then the Module checks each unit and withdraws the division if there are no possible chords. The Module checks whether there are notes combinations which form a chord. If there are possible chords, the Modules confirm the division. The Module keeps the division and checking until there is only one beat in time unit or there are no possible chords in the time unit.

5.1.3 Data Extraction

The score can not directly be used as input data for training model. It is necessary to extract the data from the score. For each time unit, we extract one vector in 12 dimensions. Each dimension represents one note in the chromatic scale of C (After key normalization). For each time unit, we count the total duration for each note and the total duration of the time unit. Then for each note, we calculate the fraction between note duration and unit duration as the input data of the predicting module. The data format will be explained in chapter 7.

5.2 predicting module

The predicting module will take features as input and predict chords. In this module, the input is N-dimensions features list and the labels are chords. All describes in this section will use the same terms.

We have implemented several methods for this module. A straightforward approach is using decision tree for simple feature mapping. Decision tree is a machine learning model that will learn a tree by splitting the source set into subsets based on labels and attributes. It is a well-known method and has shown high performance in classification. The decision tree could map each N-dimensions feature to a chord.

However, this approach could only use current features and could not make use of previous information. In music theory, previous information is useful according to professional musician. The notes in precious time unit could influence the decision of musician. In order to achieve better accuracy, recurrent models are also implemented.

Hidden Markov model is a statistical Markov model. As shown in the figure, the model contains hidden states and observations. Chords are represented as hidden states and features are observations. Using the Viterbi algorithms, the model can predict the chords sequence based on the observation sequence. In order to build the HMM model and achieve better accuracy, we firstly apply our musical knowledge to parameters for initialization and then train the model using the Baum-Welch algorithm. The details of implementation can be found in the implementation section.

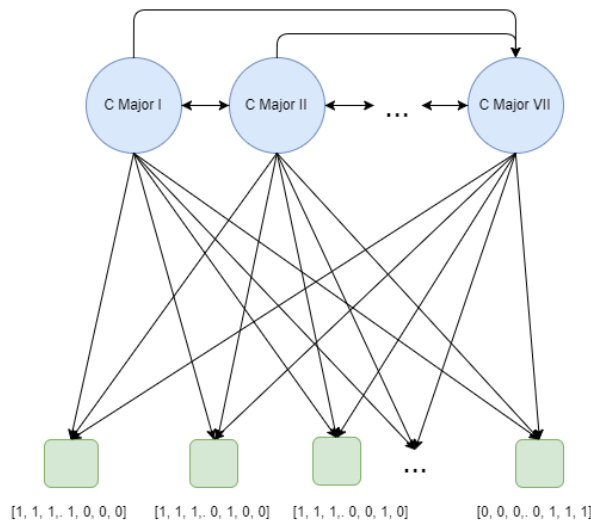


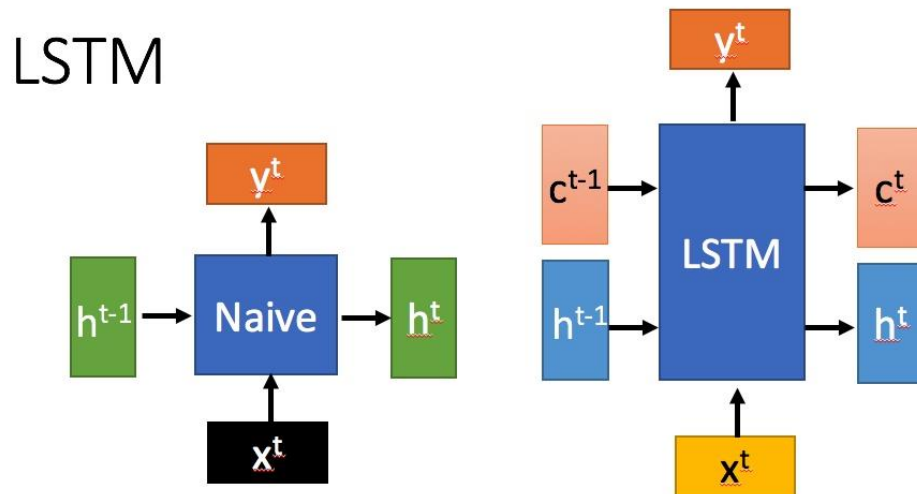
Figure 5.2

As mentioned before, in Hidden Markov model, hidden states are modeled as a first-order Markov chain. Thus, the state only influences the state at the next time unit. In other words, the prediction only makes use of current observation and the previous state. If the current state is influenced by states several time-unit before, the predicting could be hard. In order to capture the information a long time ago, we implement Long short-term memory model.

Long short-term memory(LSTM) is an artificial recurrent neural network architecture. It makes predictions based on time series data. The most important feature of LSTM is intensitivity to gap length. LSTM is designed to capture important events in a time series, between which there are lags of unknown duration.

The general architecture of LSTM is shown in the figure. X is the feature input. Y is the label. C and H represent the hidden states of LSTM. In our system, the model will take the

feature sequence as input and output chord list. X represents the note feature extracted from scores, and Y represents the corresponding chord. C and H are hidden states which are expected to carry the information of previous data. In theory, the model will make a prediction(Y) using current input(X) and information on previous input(C and H).



c change slowly $\Rightarrow c^t$ is c^{t-1} added by something

h change faster $\Rightarrow h^t$ and h^{t-1} can be very different

Figure5.3

We treat the result as the likelihood for prediction. The prediction is expected to have the highest likelihood among all possible solutions. We define score as the mean value of the highest likelihood for all testing input.

$$\text{score} = \frac{1}{n} \sum_{i=1}^n \max_j (\text{likelihood}(x_{ij}))$$

The score will be used in evaluating the module. We will talk about the details in the next section.

5.3 evaluating module

The evaluating module will first calculate the score of prediction from the output of the predicting module. If the score is lower than the threshold, it will notify the preprocessing module and do prediction again. Otherwise, the evaluating module will calculate the prediction according to the likelihood output by predicting module. Likelihood can be described as the certainty of the prediction. The higher value means that the system is more confident for the prediction. Ordinarily, we will pick the dimension with the highest likelihood as the prediction for a certain time unit.

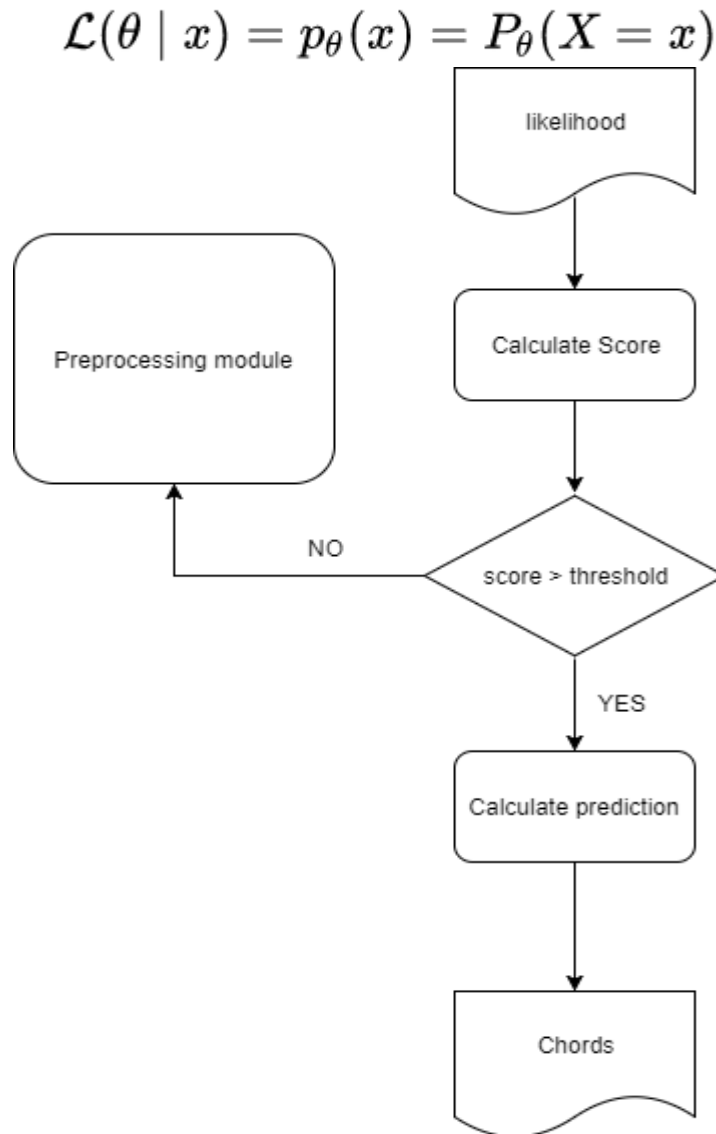


Figure5.4

Evaluating module will make a decision according to the score, which is defined in the previous section. From the perspective of the preprocessing module, if it is notified that the prediction is not acceptable, it will search for better result. As mentioned before, the bad performance could be caused by three reasons. Change of keys, multiple chords in same time unit and no chord attached to certain time unity are all possible cases. However, the

evaluating module does not know which one is the reason for bad performance. Thus, the module will try the solution for each possible case and brute force search the best solution.

The solutions to possible problems is illustrated in the following table.

Problems	solutions
key change	recognize the new key in pre-processing
multiple chords in same time unit	Divide time unit into smaller units
no chord in certain time unit	Remove input vector in sequence

Our naive method is to brute force search every possible case until we get a satisfactory result. However, our experiments show that iterative approach for time unit division could be hard to handle. The system could easily get trapped in plenty of unnecessary searches. Thus, we decide to always divide the time unit into the smallest one so that the second problem will never appear. As a result, evaluating module will only search solutions for case one and case three. It will search for the position that contains bad likelihood and then pass that position to preprocessing module. Finally, preprocessing module will calculate new key for this position. If the following prediction is not satisfactory again, the system believes this position does not contain chord. It will notify preprocessing again to remove this vector.

6. Implementation details

The system is built by Python2.7, supported by following packages.

- sklearn
- hmmlearn
- keras
- tensorflow

6.1 Predicting module

Predicting module will take vector sequence as input and make a prediction. The system could use decision tree, Hidden Markov model or LSTM model. This section will state the implementation details so that the structure of our system will become clear.

6.1.1 Decision tree

Decision tree is implemented using sklearn module. Since the input is a sequence of 12-dimension vector, each dimension of the vector is considered as a feature. The decision tree is trained using algorithms stated in the theory section. As a result, the decision tree will be able to do the classification job according to the previous data and make a decision for every single vector.

6.1.2 Hidden Markov model

Hidden Markov model is built using hmmlearn and sklearn package. Since the model assume the hidden states are under a first-order Markov chain, the Viterbi algorithm will maximise the likelihood and make prediction by single vector and previous prediction. The parameters of the Hidden Markov model are set by musical knowledge and data.

Recall the structure of HMM in our system. The parameters consist of start probability, transition probability, and emission probability. Start probability illustrates the probability of the state at the first time-step. Transition probability describes the probability of transition between hidden states. Emission probability shows the probability of certain state resulting in special observation. All of them can be represented by a probability matrix.

```
input: n * d
#state: m
#observation: k
start probability = d * m
transition probability = m * m
emission probability = m * k
```

In order to facilitate our musical knowledge and get rid of noisy data, we set the parameters by musical knowledge firstly.

We initialize the start probability by calculating the frequency of each chord in training dataset. Before calculation, we give every chord the same probability. For every chord I in the training dataset, we give one weight to corresponding start probability $SP(I)$. Finally, the parameters will be normalized so that it will sum to 1. The pseudocode can be found below.


```

1  INPUT
2  SP of size K x 1 # start probability before initialization
3  dataset # training dataset
4
5  OUTPUT
6  SP of size K x 1 # start probability after initialization
7
8  def INITIALIZE_SP:
9      # give every chord same probability
10     for i in range(K):
11         SP[i] = 1/K
12
13     # calculate probability according to dataset
14     for chord in dataset:
15         SP[chord] += 1
16
17     # normalize SP
18     SP = SP / sum(SP)
19
20     return SP

```

Thanks to professional musician's work, we generate a table of conversion rules between chord. The table describes the frequency that certain transition. We make use of this table and initialize the transition parameters according to this table.

In music theory, a fact is that for different notes on a certain chord have the highest probability of having the same lowest note. The probability will be reduced as the notes increase. For example, standard C Major chord is comprised of three notes, C, E and G. For any C Major chord, it has the highest probability to contain C, second highest probability to contain E and third highest probability to contain G. Based on this fact, previous group defines 'degree of similarity' to estimate the emission probability. We make use of this strategy for initialization. The pseudocode is shown below.

```

1  INPUT
2  note_vector # note
3  chord # the chord that note_vector belongs to
4  table # standard chord notes relationship
5
6  OUTPUT
7  similarity # the similarity between notes_vector and its standard notes
8
9  def SIMILARITY:
10     score = 0
11     st = table[chord]
12     if st[lowest] in note_vector:
13         score += 5
14     if st[second_lowest] in note_vector:
15         score += 3
16     if st[third_lowest] in note_vector:
17         score += 2
18     if st[highest] in note_vector:
19         score += 1
20     return score

```

```

1 INPUT
2 EP of size K x T # emission probability matrix before initialize
3 standard # standard chord notes relationship
4
5 OUTPUT
6 EP of size K x T # emission probability matrix after initialize
7
8 def INITIALIZE_EP:
9     for chord in range(K):
10         for notes in range(T):
11             score = SIMILARITY(standard(chord), chord, notes)
12             EP[score][notes] = score
13     EP = normalize(EP)
14     return EP

```

After initialization, we make use of dataset and tune the parameters using Baum-Welch algorithm.

6.1.3 Long Short-term memory

In order to make use of the memory characteristics of LSTM unity, we design the model in following way.

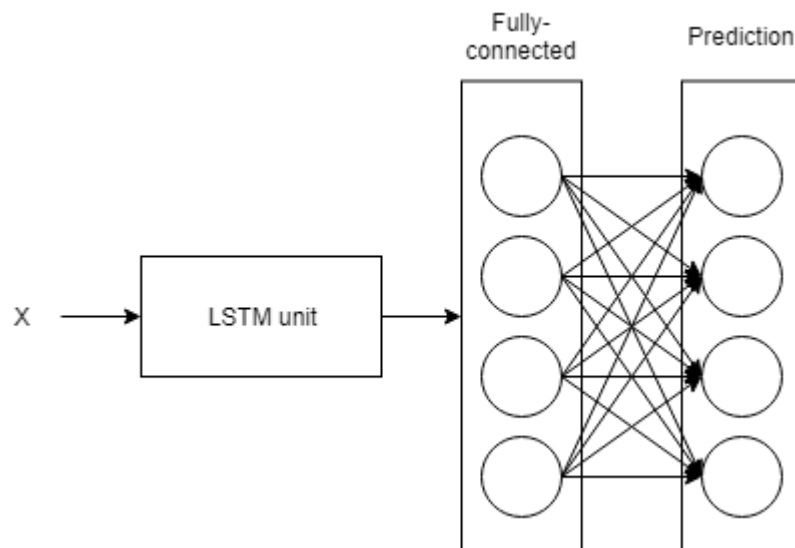


Figure6.1

As the figure above shown, the LSTM model will take the feature sequence as input. Then the sequence will be passed to the LSTM unit. LSTM unit will process the input value, and output a vector with 64 dimensions. Following the LSTM unity, there is a fully-connected layer with #chord units. The activation function for the LSTM unit and Fully-connected layer is sigmoid function. We regard the output as likelihood so that the system can evaluate the result and decide whether to run the model again.

The training of this model is based on gradient descent. We want to minimize the Euclidean distance between output and target label (chord). L2 distance is used in our approach. The objective function is listed below.

$$\text{minimize}((\text{output} - \text{label})^2)$$

The training method above only makes use of the dataset. To make use of our musical knowledge and make our model more robust, we design a training strategy. According to the initialization method stated in the HMM model, the transition between chords and conversion from chord to feature vector are available from musical knowledge. In order to make use of this information, we designed a dataset for pretrain.

According to professional musician, we generate a table that shows the appearance of the possible notes for every chord. The table can be found in the appendix. As we mentioned in the previous section, we have another table that contains the chords transition probability. Based on this prior knowledge, we design dataset in the following way.

Each input is defined as: $[(X1, Y1), (X2, Y2)]$. $X1$ and $X2$ are 12 dimension vectors. $Y1$ and $Y2$ are corresponding chord labels. $Y1$ is randomly chosen from the whole available chords. Thus, $Y1$ is uniformly distributed. $Y2$ is chosen randomly using the transition table. The transition table contains the probability of the transition from chord to chord. Thus, if we have chosen one chord, the possible next chord is chosen under the probability of transition table. As a result, $Y1$ and $Y2$ are closely connected and collection of the whole fake dataset is expected to contain the transition rules inside.

The conversion table contains the probability of emission from chord to notes patterns. $X1$, $X2$ are randomly chosen under this probability. The goal for this dataset is to teach the LSTM model first-order chord transition and conversion. Our LSTM model is trained with such fake dataset firstly. And they feed the real training dataset to it for training. We hope this strategy could teach model some musical knowledge so that the model could be more robust using same amount of training data.

7. Experiment and analysis

This section will show the experiments of various kinds of input data and model. We aim at analyzing the result, hoping this information will benefit future work.

7.1 Assumption

Firstly, since there are patterns in classical music composition, the musicians use some methods to organize the chords and notes. It is assumed that with certain prior knowledge, the model will give a more accurate prediction.

Secondly, it is reasonable to believe that the early chords have an influence on the late chords with musical consideration. Since HMM model can only predict the next state according to the previous one state. Compared with LSTM, which is capable to cover previous chords instead of only one, it is assumed that LSTM model will perform better than HMM model.

Finally, the notes are comprised of duration and pitch. The notes with different duration play different roles in music and it is reasonable to believe that duration attribute has an influence on the chords.

7.2 Training Data

The training data are vectors in different dimensions. Each dimension represents a note in a chromatic scale or diatonic scale. For the chromatic scale, there are total 12 notes in the scale and it covers all the notes in an octave. For diatonic scale, there is a total of 7 notes in the scale. Although it better suits the numeral Roman numeral analysis, some data are ignored. Last semester, because 12 dimension of data leads to huge HMM model, we decide to use the diatonic scale represent the notes. For ignored information, we designed to add a dimension to represent the chord type. Currently, because of the stronger computing capability of LSTM, we decided to use the chromatic scale which is able to cover all information from the score.

For data with only appearance information, the data are binary. For example, the first measure of the happy farmer. The data of the measure two vectors represent possible chords in measure: $\{[1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0], [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0]\}$ The element with value 1 represents the occurrence of the note in the measure, on the contrary, the element with value 0 represents the notes do not occur in this measure.

For data with duration information, there are also two vectors represent possible chords in measure: $\{[0.62, 0.0, 0.0, 0.0, 0.12, 0.0, 0.0, 0.25, 0.0, 0.0, 0.0, 0.0], [0.12, 0.0, 0.0, 0.0, 0.38, 0.0, 0.0, 0.5, 0.0, 0.0, 0.0, 0.0]\}$. For each element with value which are not zero, it

represents the note duration ratio of the total duration.



Figure7.1

All the data are from consultant Lucas Wong and youtuber Timon. They help us recognize the chord and key in scores. Besides Lucas also help us with musical background and issues. He answered our musical questions and explains the puzzles for us. Thanks for their help in this project.

7.3 experiments

All experiments in this section are under the following setting. We use 12-dimensions vector sequence as input. The vector contains the information of duration for a single time unit. 10% of the training set are considered as a validation set to avoid overfitting. After obtaining the model, we test the model with the following scores. The LSTM model is trained using Adam optimizer. The maximum iterations for pretrain is 20 and the maximum iterations for LSTM training is 50.

7.3.1 Pretrained model vs non-pretrained model

According to the implementation section, we design a special dataset for pretrain. The sample of pretrain dataset can be found in the appendix. The result is shown below.

Testing Score	LSTM with pretrain	LSTM without pretrain
The happy farmer	87.5%	80%
Von fremden Ländern und Menschen	85%	77.5%
Piano Sonata No.11 K.331 3rd Movement Rondo alla Turca	83.3%	75%
Eine Kleine Nachtmusik - by Wolfgang Amadeus Mozart	72.9%	78.6%

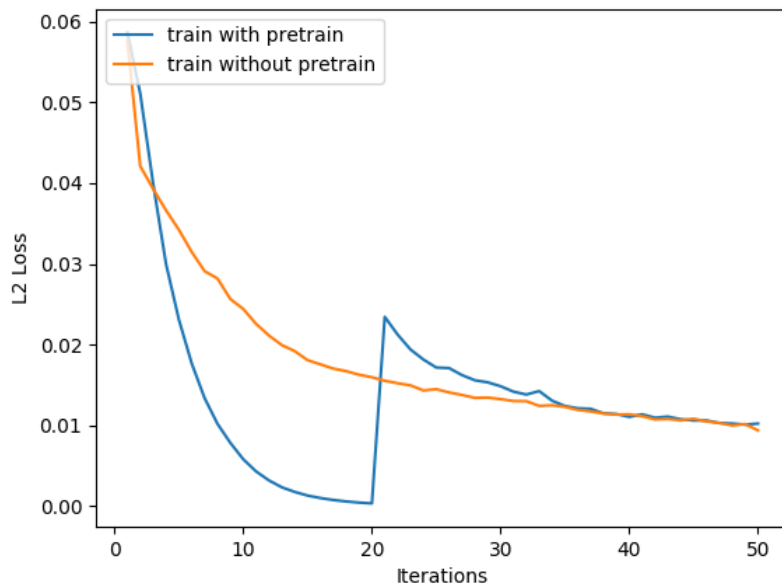


Figure7.2

The pretrained model is fed fake data for 20 iterations firstly. After that, it will be feed real data. As shown in the previous figure, there is a cascade at 20 iterations. That is the point that the model starts to be trained using real data. The pretrained model and no-pretrained model will finally converge to the same loss.

The model is expected to learn transition and conversion rule from our pretrain dataset. The experiments show that the pretrain process indeed makes some contributions and is able to reduce the training iterations. Another advantage of pretrain strategy is that the model could be more robust. Because the available dataset is limited, we collected the dataset manually. As a result, the data we have is extremely valuable and the amount is still not enough. Unavoidable, the data may be biased. Since we seek help from youtuber, the score owned by him is also limited by his preference. In order to make the model robust to new data, general musical knowledge is needed. From this aspect, the pretrain strategy is useful and necessary.

7.3.2 HMM vs LSTM

As a statistical model, HMM is expected to be better explained and easier modeled. The musical knowledge is better facilitated because parameters are more intuitive. Compared to HMM, LSTM is based on Neural Network and is more like a “black box” guessing. Although we define special dataset using musical knowledge, expecting the model can learn some transition and conversion rules, the training is still based on gradient descent which could only lead to local optima. The training result is not predictable and is hard to analyze. It is possible that the model could learn something we do not intend to. Thus, from the complexity of analyzing aspect, HMM is not easier and welcome. However, LSTM is driven from the dataset directly and is able to capture information a long time ago. Besides, because Hidden Markov model will treat notes features as discrete observations. [C, D, E] and [C, E, G] are totally different from each other. It is possible that both notes exist in a time unit. It is difficult for preprocess to figure out which one is the feature we want. As a result,

the model highly depends on the preprocess and the quality of feature extraction. However, as a kind of Neural Network, the LSTM treat the input continues. [C, D, E] and [C, E, G] are close to each other in input space. The value of each dimension could contribute to both training and testing processing. This characteristic makes it is possible that the model could learn the rules quickly and easily. Thus, from the practical aspect, LSTM model is expected to get a better result.

This experiment aims to compare the performance of these two models for the same task. In this experiment, we train LSTM model with pretrain. The splitting strategy is the same as the previous experiment.

Testing Score	HMM accuracy	LSTM accuracy
The happy farmer	77.5%	87.5%
Von fremden Ländern und Menschen	72.5%	85%
Piano Sonata No.11 K.331 3rd Movement Rondo alla Turca	66.7%	83.3%
Eine Kleine Nachtmusik - by Wolfgang Amadeus Mozart	61.5%	78.6%

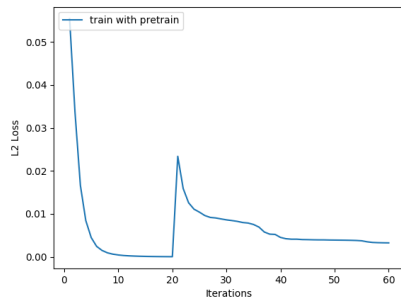
The experiment validates our analysis. Under the same environment, HMM could achieve 76% accuracy. Compared to the bad performance of HMM, LSTM achieves around 90% accuracy. The result of LSTM is highly closed to professional musician's identification. If we focus on the wrong prediction, we find that almost all of the different parts are controversial. Because the chord identification is subjective, different musicians may have different opinions. Thus, our LSTM model indeed achieves a good result.

7.3.3 Duration vs appearance

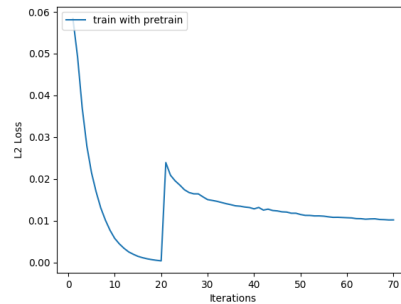
Clearly, if we collect the duration of the note in certain time unit, we are able to keep more information compared to count appearance. Intuitively, the duration for note is useful for chord identification. However, if we care appearance only, the model will be reduced since we only have finite kinds of note combination. Of course, the training of a finite model is easier. The experiment to compare the performances of the model in dealing with duration input and appearance input is necessary.

Testing Score	Appearance input	Duration input
The happy farmer(40)	90%	87.5%
Von fremden Ländern und Menschen(40)	77.5%	85%
Piano Sonata No.11 K.331	75%	83.3%

3rd Movement Rondo alla Turca(12)		
Eine Kleine Nachtmusik - by Wolfgang Amadeus Mozart	71.5%	78.6%



appearance



duration

Although we get higher accuracy using appearance input for the happy farmer, the appearance input will get worse performance in overall testing inputs. Because the value of appearance input is either 0 or 1, the gradient of may be a large number compared with the proportion number in duration input. As a result, the model will converge faster and the loss can reduce quickly. However, If we consider appearance only, the noise is treated the same as a useful note. Normally, the noisy note has a short duration. If we consider duration, the noise could make few contributions to the training process. Because of the limited amount of dataset, the noise cannot be ignored. This may be the main reason for the bad performance of appearance input.

8. Contribution

The system is built by Li Shuhe and Zhang Zhaochen. We aim at building a chord identification system for music reduction. The system is designed and tested by us together. Most of the time we were working together so that the discussion could be easy. Of course, this system is a huge project so that we need to divide the workload.

In a word, the system consists of a preprocessing module, predicting module and evaluating the module. Preprocessing module is mainly built by Zhang Zhaochen. Predicting module is mainly built by Li Shuhe. Compared with these two modules, the evaluating module is music and machine learning. We believe it would be best to work together for this part.

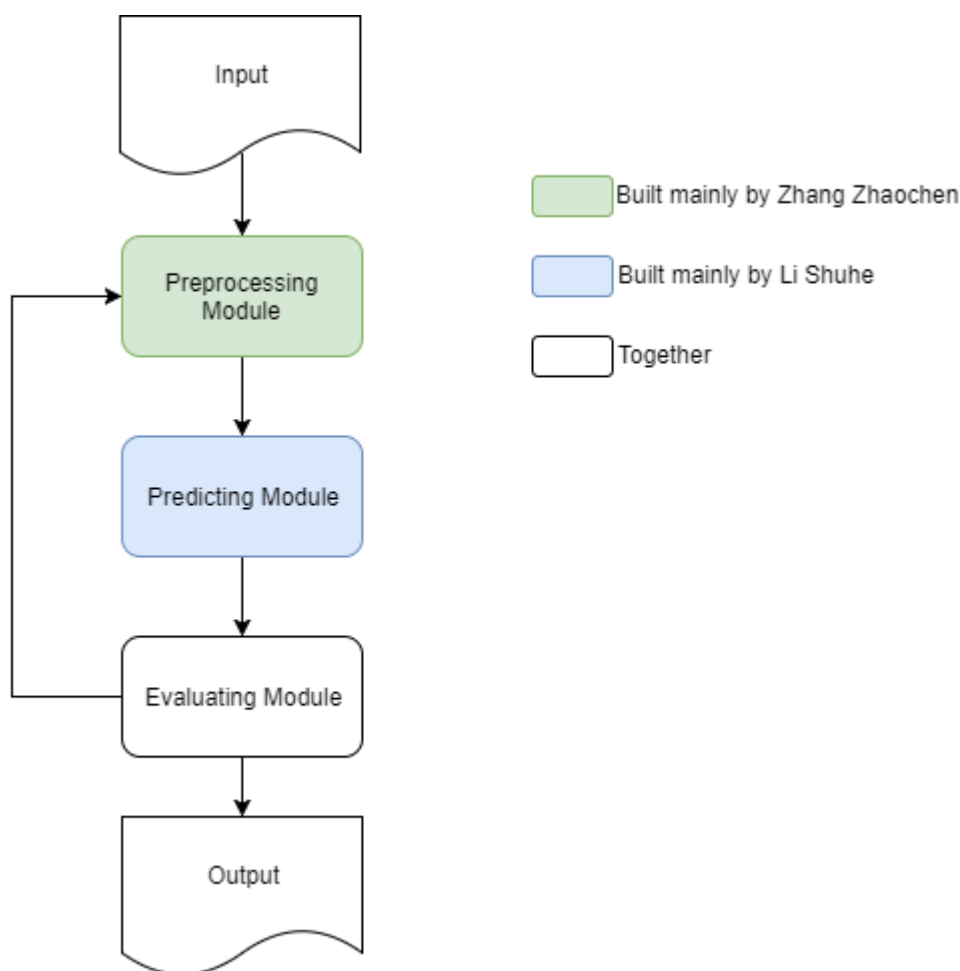


Figure8.1

8.1 Work done by Zhaochen ZHANG

As shown in the figure, Zhaochen builds the pre-processing model of the system. Zhaochen uses the key-finding algorithm of Krumhansl-Schmuckler to identify the key information of the music. Then Zhaochen implements the multiple chords division of the pre-processing mode. Zhaochen designs a function of chord division which detects the possible

chord of the measure and divides the measure into smaller time units. In order to get the trainable data, Zhaochen first normalizes all the major(minor) key to C major(minor) and then extracts the data of notes with appearance and duration information. Besides the pre-processing model, Zhaochen is also in charge of data collecting. Zhaochen collects all the scores and chords information from internet and consultant and then generate trainable data and manually labeling and proofreading the chords of the data. Finally, Zhaochen is also responsible for all music background of the system, including system design and models design.

9. Conclusion

After the experiment, we can make conclusions about the assumptions. Firstly, the composition pattern is helpful in chord recognition. We use pre-train method to train the model with the self-made data which follows the music composition pattern and get the better result. Secondly, LSTM model is more suitable in chord recognition, because LSTM can handle the event with a long time delay. As for classical composition, the pattern of the melody usually begins with a certain chord and ends with the same chord. The flow of the chords takes time to finish and the HMM model just ignore this information. The average accurate of LSTM model is higher than HMM Model. Thirdly, the model trained with data of duration performs better than the model trained with data of appearance. The duration of the notes is also important in chord recognition. Musically, the notes with higher duration are usually more important in music score and performance. Although in the experiment of the happy farmer, the accuracy of appearance input is higher than duration input, the overall performance of duration input is higher appearance input. It is possible that the shortage of data causes this unreasonable result. Finally, chord recognition is totally practicable using neural network. The average accuracy of LSTM is more than 80% which is acceptable for the project. With adding music features, such as note duration and composition pattern, we get an even better result. Although music composition is artificial, it is still possible and feasible to build a model to recognize the chord.

10. Reference

- Chung J, Gulcehre C, Cho K H, et al. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling[J]. Eprint Arxiv, 2014.
- Graves A. Long Short-Term Memory[J]. Neural Computation, 1997, 9(8):1735-1780.
- Rabiner, L. R. (1990). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Readings in Speech Recognition, 267-296. doi:10.1016/b978-0-08-051584-7.50027-9
- Peter K, Manda R. (2016). Markov Chains of Chord Progressions.
- Robert, Hart. (Aug. 19, 2012). Key-finding algorithm <http://rnhart.net/articles/key-finding/>

11. Appendix:

11.1 Data source:

Score name
Concerto for Flute and Harp in_C Major K.299 -by Wolfgang Amadeus Mozart
Sonata_No._16_1st_Movement_K._545 -by Wolfgang Amadeus Mozart
The happy farmer -by Robert Schumann
Von fremden Ländern und Menschen -by Robert Schumann
Eine Kleine Nachtmusik -by Wolfgang Amadeus Mozart

11.2 Sample pretrain data:

.pretrain	data generated by musical knowledge																
0	[[0.0 0.0 0.33 0.0 0.0 0.33 0.0 0.0 0.0 0.0 0.0 0.33] [0.25 0.0 0.0 0.0 0.25 0.0 0.0 0.25 0.0 0.25 0.0 0.0]]															VII	VI
1	[[0.33 0.0 0.0 0.0 0.33 0.0 0.0 0.0 0.0 0.33 0.0 0.0] [0.33 0.0 0.0 0.0 0.33 0.0 0.0 0.33 0.0 0.0 0.0 0.0]]															VI	I
2	[[0.0 0.0 0.33 0.0 0.0 0.0 0.0 0.33 0.0 0.0 0.0 0.33] [0.0 0.0 0.25 0.0 0.0 0.25 0.0 0.25 0.0 0.0 0.0 0.25]]															V	V
3	[[0.0 0.0 0.0 0.0 0.33 0.0 0.0 0.33 0.0 0.0 0.0 0.33] [0.0 0.0 0.33 0.0 0.0 0.0 0.33 0.0 0.0 0.0 0.33]]															III	V
4	[[0.0 0.0 0.33 0.0 0.0 0.33 0.0 0.0 0.0 0.33 0.0 0.0] [0.33 0.0 0.0 0.0 0.33 0.0 0.0 0.33 0.0 0.0 0.0 0.0]]															II	I
5	[[0.25 0.0 0.0 0.0 0.25 0.25 0.0 0.0 0.0 0.25 0.0 0.0] [0.0 0.0 0.25 0.0 0.0 0.25 0.0 0.0 0.25 0.0 0.25 0.0]]															IV	VII
6	[[0.25 0.0 0.0 0.0 0.25 0.0 0.0 0.25 0.0 0.0 0.0 0.25] [0.25 0.0 0.0 0.0 0.25 0.0 0.0 0.25 0.0 0.25 0.0 0.0]]															I	VI
7	[[0.0 0.0 0.33 0.0 0.0 0.33 0.0 0.0 0.0 0.0 0.0 0.33] [0.0 0.0 0.25 0.0 0.0 0.25 0.0 0.25 0.0 0.0 0.0 0.25]]															VII	V
8	[[0.0 0.0 0.33 0.0 0.0 0.0 0.0 0.33 0.0 0.0 0.0 0.33] [0.0 0.0 0.33 0.0 0.0 0.0 0.33 0.0 0.0 0.33 0.0 0.0]]															V	V
9	[[0.0 0.0 0.25 0.0 0.25 0.0 0.0 0.25 0.0 0.0 0.0 0.25] [0.33 0.0 0.0 0.0 0.33 0.0 0.0 0.33 0.0 0.0 0.0 0.0]]															III	I
10	[[0.25 0.0 0.0 0.0 0.25 0.0 0.0 0.25 0.0 0.25 0.0 0.0] [0.0 0.0 0.0 0.0 0.33 0.0 0.0 0.33 0.0 0.0 0.0 0.33]]															VI	III
11	[[0.0 0.0 0.33 0.0 0.0 0.33 0.0 0.0 0.0 0.33 0.0 0.0] [0.0 0.0 0.25 0.0 0.25 0.0 0.0 0.25 0.0 0.0 0.0 0.25]]															II	III
12	[[0.25 0.0 0.0 0.0 0.25 0.0 0.0 0.25 0.0 0.25 0.0 0.0] [0.25 0.0 0.0 0.0 0.25 0.25 0.0 0.0 0.0 0.25 0.0 0.0]]															VI	IV
13	[[0.0 0.0 0.33 0.0 0.0 0.0 0.0 0.33 0.0 0.0 0.0 0.33] [0.0 0.0 0.33 0.0 0.0 0.0 0.33 0.0 0.0 0.33 0.0 0.0]]															V	V
14	[[0.0 0.0 0.33 0.0 0.0 0.33 0.0 0.0 0.0 0.33 0.0 0.0] [0.25 0.0 0.25 0.0 0.0 0.25 0.0 0.0 0.0 0.25 0.0 0.0]]															II	II
15	[[0.0 0.0 0.33 0.0 0.0 0.33 0.0 0.0 0.0 0.0 0.0 0.33] [0.33 0.0 0.0 0.0 0.33 0.0 0.0 0.0 0.33 0.0 0.0 0.0]]															VII	VI
16	[[0.0 0.0 0.25 0.0 0.25 0.0 0.0 0.25 0.0 0.0 0.0 0.25] [0.25 0.0 0.0 0.0 0.25 0.25 0.0 0.0 0.0 0.25 0.0 0.0]]															III	IV
17	[[0.0 0.0 0.25 0.0 0.25 0.0 0.0 0.25 0.0 0.0 0.0 0.25] [0.0 0.0 0.25 0.0 0.0 0.25 0.0 0.0 0.0 0.25 0.0 0.25]]															III	VII
18	[[0.25 0.0 0.0 0.0 0.25 0.25 0.0 0.0 0.0 0.25 0.0 0.0] [0.0 0.0 0.25 0.0 0.0 0.25 0.0 0.0 0.0 0.25 0.0 0.25]]															IV	VII
19	[[0.33 0.0 0.0 0.0 0.33 0.0 0.0 0.0 0.0 0.33 0.0 0.0] [0.0 0.0 0.25 0.0 0.25 0.0 0.0 0.25 0.0 0.0 0.0 0.25]]															VI	III
20	[[0.0 0.0 0.25 0.0 0.0 0.25 0.0 0.25 0.0 0.0 0.0 0.25] [0.0 0.0 0.25 0.0 0.0 0.25 0.0 0.25 0.0 0.0 0.0 0.25]]															V	V