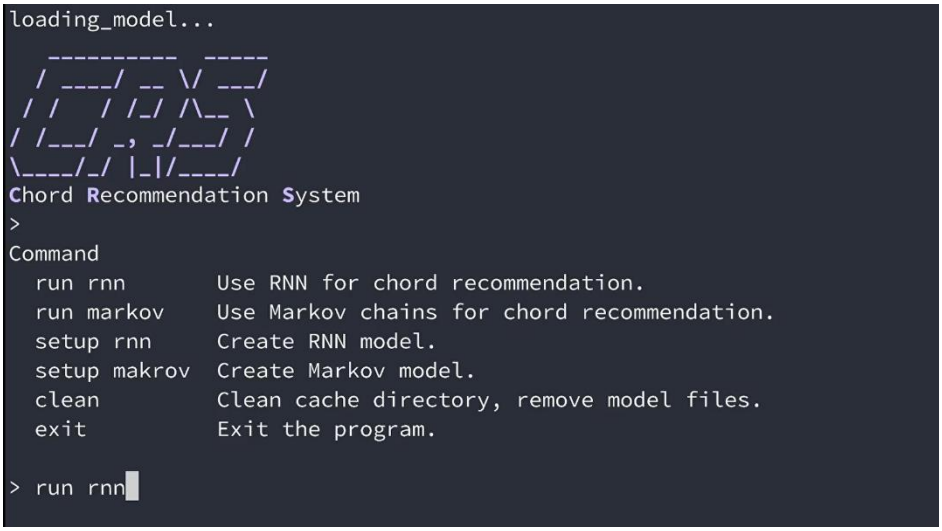
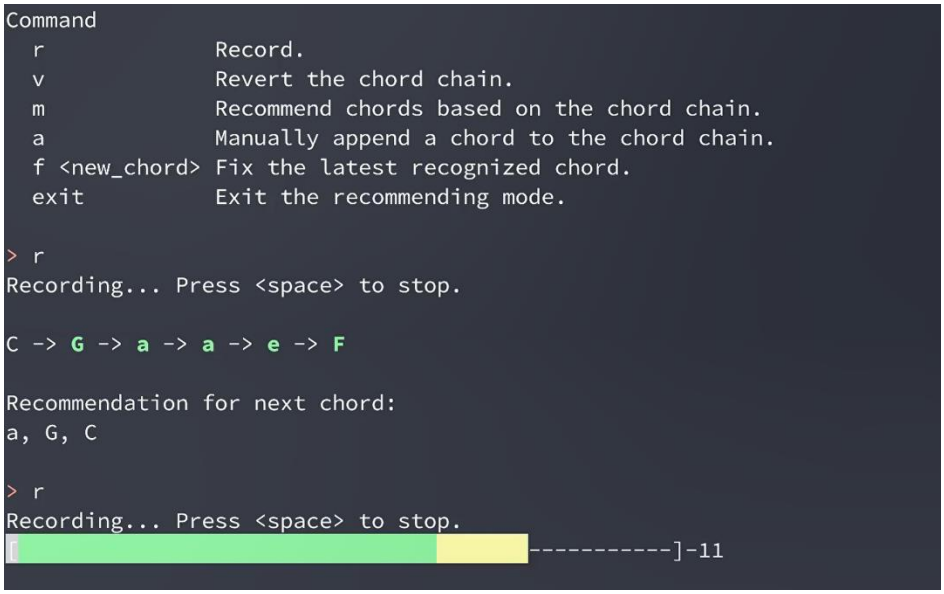
Automatic Chord Recognition and Recommendation | Computational Music Analysis Project 2019

My team and I designed and built a python-based functional tool that can be used to fulfill guitarists' need. This tool can record multiple chords played by guitar player, recognize the played chords, and give recommendations to what chords to be played next that are more musically favored. We also provided functions of reverting the chord chain, appending a chord to the existed chain manually, and change the latest recognized chord manually, to handle different situations like the tool gave incorrect chord recognition results.



I was in charge of the chord recognition part.

Onset Detection



We wanted to detect multiple chords played, so onset detection was an essential part to slice the raw input audio file into several pieces. In order to detect the onset, namely the starting time, of each chord, I first computed the short-time root mean square (RMS) of the input raw audio file. This step can be comprehended as using an RMS energy to represent all samples in a certain frame. Then, I took dB scale to the short-time RMS and lowpass filtered it. The last step was to calculate the difference of filtered RMS between neighboring frames. When the difference was larger than a certain threshold, we could say that there was an energy jump between these neighboring frames and thus there was an onset.

Chromagram

Chromagram was the audio feature I used for chord recognition. Chromagram can be regarded as the frequency spectrum on specific frequencies that correspond to the 12

semitones in an octave. I used constant Q transform (CQT) to compute chromagram but not short-time Fourier transform (STFT) as in most DSP applications. This is because CQT can be interpreted as computing a DFT only for specific, logarithmically spaced, frequency bins. One positive side effect of CQT is increasing the frequency resolution in low frequency range. The Q parameter in CQT can control frequency resolution per octave.

Dataset

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	label	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
2	C	2.218343	2.642597	2.471548	2.783176	2.193303	2.013497	3.214463	3.791406	2.281082	1.240255	1.24672	1.862589
3	C	2.354802	2.206073	1.952043	2.202493	2.314739	1.628388	2.884665	3.989051	2.404552	0.833431	1.037413	1.954233
4	C	2.582823	1.784622	1.477702	1.780898	2.269034	1.372235	2.44286	4	2.229018	0.555382	0.863198	2.059141
5	C	3.059818	1.764225	1.302711	1.630367	2.313995	1.390274	2.37493	4	2.099988	0.531781	0.817918	2.320837
6	C	3.454078	1.927023	1.29295	1.611368	2.261184	1.402035	2.385669	4	2.084381	0.477726	0.75023	2.402391
7	C	3.683336	1.920104	1.218697	1.533408	2.25927	1.338339	2.341696	4	1.998519	0.382807	0.639746	2.400822
8	C	3.80981	2.019646	1.200648	1.543479	2.23649	1.300544	2.372389	4	2.011384	0.340413	0.544469	2.444114
9	C	3.819641	2.072399	1.187036	1.498919	2.131972	1.243022	2.395632	4	2.056899	0.31869	0.486711	2.415515
10	C	3.862051	2.04697	1.099854	1.398794	2.03757	1.155226	2.351942	4	2.037531	0.303268	0.438751	2.426028
11	C	3.900168	2.06092	1.067383	1.347403	1.969566	1.12662	2.386304	4	2.055947	0.307975	0.409557	2.480453
12	C	3.899667	2.164921	1.084631	1.16212	1.884625	1.074291	2.405637	4	2.044552	0.342278	0.453522	2.518943
13	C	3.767158	2.217273	1.133847	1.047824	1.793613	1.065822	2.387197	4	2.019389	0.421335	0.548815	2.498861
14	C	3.586001	2.299274	1.256396	1.015884	1.798078	1.179859	2.508577	4	1.999786	0.528894	0.761453	2.573414
15	C	3.539978	2.639796	1.421297	1.020757	1.74343	1.349561	2.698978	4	2.102315	0.730963	1.082269	2.542759
16	C	4	1.700554	1.179972	2.070397	3.760025	2.170605	1.709906	1.253077	1.055613	0.874697	0.988873	3.36389
17	C	3.922146	1.806172	1.146785	2.318329	3.878881	2.284975	1.964534	1.638808	1.305361	0.981118	0.915541	3.227481
18	C	3.740562	2.049274	1.313772	2.418694	3.83975	2.419965	2.389204	2.286777	1.786582	1.12104	0.935353	3.082492
19	C	3.311192	2.075589	1.403941	2.252708	3.520431	2.205494	2.688688	2.944276	2.158517	1.114489	0.975559	2.695515
20	C	2.812258	2.003339	1.514902	2.033657	3.119388	2.003746	2.839943	3.597674	2.424084	1.091784	1.03322	2.23822
21	C	2.57543	1.874948	1.5652	1.783788	2.629463	1.86255	2.8293	4	2.459684	0.972822	1.044911	2.030885
22	C	2.623566	1.83393	1.398061	1.528541	2.267756	1.523372	2.601284	4	2.239923	0.768133	0.914342	1.908596
23	C	2.919352	1.853722	1.286994	1.464668	2.176393	1.404801	2.503648	4	2.125	0.699817	0.824859	1.997744
24	C	3.25261	1.819681	1.13467	1.428111	2.031569	1.25464	2.443222	4	2.040087	0.589084	0.711459	2.079364
25	C	3.418696	1.911959	1.098397	1.42919	1.979981	1.182576	2.389327	4	1.999174	0.452914	0.585126	2.081746
26	C	3.478458	1.924993	1.063263	1.42407	1.923147	1.110827	2.323324	4	2.006456	0.407436	0.559348	2.07403
27	C	3.495988	1.934532	1.06083	1.382765	1.850908	0.994977	2.237254	4	2.018659	0.401874	0.600897	1.981175
28	C	3.546738	2.135498	1.210568	1.422204	1.829719	1.019898	2.192783	4	2.188819	0.52443	0.777279	2.030916

With a program that could compute chromagram feature, we could build a dataset by ourselves. My team member and I played each major chord and minor chord for more than 40 times with guitar, extracted their chromagram, then formed our dataset with chord names as labels and chromagram as features. I trained and tested three classifiers including kNN, SVM and Gaussian Naive Bayes with our dataset, all had more than 92% accuracy. With a trained classifier model, the whole process of chord recognition would be: 1. record chord sequences, 2. slice the audio file with onset detection, 3. extract chromagram for each slice, 4. classify to chord names with classifier.

	kNN	Gaussian NB	SVM
Accuracy	0.9819	0.9294	0.9864
Time spent	0.3084	0.0215	0.0290

Chord recommendation

The chord progression could be modeled as a Markov process, where the transitions from state to state are based on the observations of limited numbers of previous states. We built a Markov chain model by extracting the chord-to-chord transitions from the McGill Billboard dataset and obtain the transition probabilities represented by a multi-dimensional matrix.