Some notes about chords estimation from audio

Christophe Hauser - kereoz@kereoz.org

September 6, 2009

Abstract

Here are some notes about my work on chord detection from audio at the University of Tokyo this summer. An acoustic model is first made from audio. It is based on chroma vectors extraction. Language models are then used along with it in order to get the most likely sequence of chords. I have previously been working on probabilistic modeling of chord sequences using the n-gram model, and the present work is an attempt to improve the detection using such language models.

Acknowledgements

I would like to address special thanks to my supervisors here in Tokyo, professor Sagayama and Ono for their advices, for their great patience in helping me and for their guidance all along this period. I am also grateful to all the people in the Team for their valuable advices and sympathy.

1 Introduction

In chord detection, acoustic models are used to detect chords frame by frame in audio files. No dependency is expressed from one frame to another, and inconsistent chords are kept as is. Using language models with supervised learning can be used to express this dependency, leading to more consistent chord sequences (in a probabilistic sense).

In the following, acoustic models are represented as Matrix. Language models, over sequences of symbols, represent the dependency between those symbols in the time.

Using a language model brings transition probabilities to the acoustic model, resulting in a lattice. The most likely sequence can then be decoded using the Viterbi algorithm.

2 Acoustic Model

The acoustic model we are using is based on chroma vector extraction from audio. This chroma extraction is achieved by a model from Hélène Papadopoulos [4] using a sliding window. These are 12 dimensional vectors, corresponding to all 12 possible semitones in music notes : (C Db ... B). The used frame size is 0.48s. Also, the considered harmonics in the fourier transform are in the range 60-1020 Hz, to avoid high frequency noise.

In order to get beat synchronized vectors, we use the beat tracker from Dann Ellis [1], and compute the average of the chroma vectors between two beats.

We consider only major / minor chords, leading to a dictionary of 24 chords. eg. Cmaj = C - E - G

The acoustic model is in the form of a N x 24 matrix, with N being the number of beats in the song. Each column in the matrix correspond to one of the 24 chords in the dictionary. The matrix contain the probabilities, for each chroma vector, that it corresponds to a given chord.

This probability (pseudo probability) is obtained by calculating the distance (scalar product) between chroma vectors and theoretical chord templates. Chord templates are calculated the following way :

- Give a weight of 1 to each composing note of the chord (C-E-G for Cmaj)
• Consider the weight of the harmonics of each note follows a mathematic sequence of power 0.6, as in Emila Gomez phd [2]
• Normalize the final vector so it sums to 1

The resulting acoustic model is a Matrix, directly usable as an input for 

3 Language model

The language model we use is trained on Harte transcriptions of the Beatles [3]. We transformed those transcriptions in order to use the “Quaero dictionary”, as in the acoustic model. The “Quaero dictionary” has 24 chords, in the form root + type with type being either maj or min, eg Cmaj Amin Dmin etc. We also transform each song into all possible tonalities to have all the transition probabilities in the model.

In the transcriptions from Harte, information about chords and their duration if provided. In the acoustic model, each chroma corresponds to one beat. In order to match the acoustic model in terms of time scaling, we use a notion of “atomic time” in the training set. For each song, we determine the shortest chord, and express all the other chords according to this atomic time using repetitions. For example, a chord repeated twice means its length is two times the atomic duration.

Each song is then converted to all possible tonalities, and an n-gram model is learnt over it. Kneser-Ney smoothing is used here, and the order of the model has been tested with different values from N=2 to N=8.

4 Decoding

The decoding part is identical to one common language processing problem: decoding the most likely sequence of tags over words, in the model. The words are the names we attribute to each chroma vector (Chroma + beat number, eg Chroma1 Chroma2...ChromaT), and the tags are the 24 possible chords.

“Tagger-irisa” by Guillaume Gravier allows to solve such problem, and we use it here. It has been patched to support setting a so called fudge factor $\lambda$, weight of the language ($\lambda$) and acoustic model ($1-\lambda$).

5 Results

Optimal results are observed with a really low fudge factor (language model weight), about 0.1. In most cases, attributing higher weight to the language models leads to decreasing the accuracy. The reason for this is:

• time scaling might not be accurate enough
• the training set includes many transitions (all the tonalities for each song)
• Unrelated chords can have high probabilities in the acoustic model because they are close to the actual chord in term of notes (but this doesn’t make sense for the language model)

6 Improvements

To address this problem, the dataset should be tonality independent, and have accurate beat information (eg one repetition means one beat). In order to get a tonality independent acoustic model, there are two possible ways:

• Measuring the distance between the root of each chord and the key of the tonality
• Finding the function of this chord in the harmonization of the major scale (see notes on chords)

In both cases, we would need to:

• Compute the acoustic model (Matrix)
• Decode the most likely sequence using the acoustic probabilities only
• Detect the tonality (and eventual modulations), resulting in a modulation vector
• Modify the acoustic model in order to obtain a 12xN matrix, where each column is one of the possible root independent chord

• Convert all the training set to a root independent notation

• Decode the most likely sequence in a root independent fashion

7 Notes on chords

There are two ways to look at chord detection

• exact transcription : we want to know exactly what chord is being played, which is the bass note and which is the inversion, which is highly related to the classical music approach

• what is the function of the chord being played, which notes can or cannot exist at a given time in the song (including all the instruments together), which reflects the jazz approach.

7.1 Chords and tonality

If we look at what can be done out of the notes in a Tonality, we can see that harmonizing the major scale of this tonality leads to seven modes, that is, seven scales and seven chord types. These are IM7, IIm7, IIIm7, IVM7, V7, VIIm7 and VIIIm7b5

For example, there is no possible D7 in tonality of C, it has to be a Dm7. If we consider only minor/major chords, we would consider it as being Dm. If we consider more notes, we can say it is a Dm9 (as it’s 9th is in the tonality). In the case where we only consider 24 chords, there are many ambiguous situation : Dm could be either Dm7 (mode 2 or 3) or Dm7b5 (mode 7). Having information on the 7th is really helpful, and in most cases we can disambiguate chords functions within a context.

Therefore, considering a dictionary of 7 root independent chord functions is enough to represent all existing chords in all the tonalities. It is some sort of clustering where we would use seven chord classes, and learn transition probabilities between those classes.

8 Practical aspects

Xml output files with final chord sequences are located in the xml/ directory.

Generated files are located in the gen subdirectory.

• .gen files are acoustic models

• .est files are most likely sequences according to the acoustic model only

• .lmest files are obtained by decoding the most likely sequence using both the acoustic and the language model

• .qo files are Harte transcriptions transformed using the dictionary used in the Quaero evaluation (and in the acoustic model)

The lib folder contains Ruby classes and modules, which are documented in the html format in doc/ The utils folder contains various executables scripts that can be called directly to perform operations such as evaluating the accuracy of a transcription file given the Harte transcription, generate acoustic models from wav files, generate decoded sequences according to a language model and an acoustic model etc...

Note : an improved acoustic model, taking 7th notes into consideration, is available as matlab code in the mcode directory. It is possible to use it by replacing some lines in run.m (those are commented, see comments). Some code has also been made in order to detect the tonality of a sequence of chord, to detect their mode, and to retrieve modal information from minor/major scale using probabilistic models. It is written as Ruby code, in the lib/tonality.rb module.

References

[1] Beat tracket (matlab code)  
