Learning harmonic changes for musical style modeling

Jean-Noël Rivasseau
Department of Computer Science
University of British Columbia
Vancouver, Canada
jnriva@cs.ubc.ca

Abstract

In this project I created a system capable of performing statistical learning of the harmonic changes present in musical pieces. For that, musical pieces must be pre-processed and normalized, so that the obtained harmonies are key-independent. Once learning has been done, new harmonic sequences can be generated and analyzed, to better understand what exactly the system learns (and how). The results of several experiments show that the system is able to correctly normalize harmonies and produce realistic new sequences, but I believe additional work would really help in improving its correctness and capabilities. Nevertheless, our current system has achieved its goals of converting "raw music scores" onto harmonic sequences, and capturing some of the structure of these sequences. It could be used as a solid basis for any future work dealing with NLP techniques applied to musical composition.

1 Introduction

Computational linguistics techniques are usually applied to natural language processing. A variety of techniques have been developed over the years, in order to attain "high-level" goals such as summarization of documents. However, such techniques can also be applied to other fields. Music has in fact many parallels with natural language; it has its own symbols (not words, but notes), but also its own structure and characteristics. As words are not chosen randomly in order to form a book, so notes are not in a piece: the order of notes and harmonies obviously obey some formal rules, much like the grammar rules for English or any other language. Being able to understand and formulate those rules would be a first step towards a generation of a musical work by a computer, a task that several researchers have already attempted using various methods (see [1, 2]).

Several problems arise when trying to apply NLP techniques to music. The first and biggest one is the availability of huge corpora. NLP techniques are usually based on statistical analysis, so large corpora of text information have been built and processed ("tagged", generally) in order to provide a statistical data bank for many purposes. There is no such available musical corpus, so researchers in this field must create and build their own corpus. A second problem is the choice of symbols that are to be used for modeling music. Symbols are important because essentially, a given system will try to learn something based on these symbols. If we wanted to learn grammar rules, such as “a verb is usually followed by a noun”, we would use a corpus of symbols representing different grammatical categories (pronouns, verbs, and so on...). We would not care about the actual words. On the other hand, if we wanted to build a system for summarization, obviously the sense of each word would have to be modeled. The same analogy applies to music; should individual notes be used as symbols? or sequence of chords ? It obviously depends on what one is trying to learn and infer.

This project’s goal is to build a system capable of recognizing the sequences of harmonies in a given piece or corpus (collection of pieces). This is motivated by the fact that I believe harmony can be a good indicator of the structure of a musical work. Some of my previous work in the area of automated music composition dealt with the clustering of several music pieces into various “styles”. In order to achieve this, the data extracted from the musical works consisted only in “transitions” from one note to another, and the order didn’t matter. While such a system achieved some results, and the clustering was overall good, the generated music coming out of the algorithm didn’t make much sense. That was, in my opinion, in no small part due to the fact that relying only on note transitions isn’t constrained enough. While it may capture some patterns used by a particular composer, overall it will produce music that is not bound to any rules, which is not the case for almost any human-composed work, since usually music obeys some general rules of harmony. To give a very rough analogy, considering only notes is like considering only letters (and not words) in English. It may be true that “r” is more likely to be followed by “a” than by “z”; however generating English sentences by learning only such probabilities will almost always produce gibberish and not actual words.

That is why I decided to go one step beyond this model and try to use harmony to learn the structure of musical works. The symbols that we are going to use are thus harmonic symbols. A harmony is an abstract set of notes (whose exact pitch, or even arrangement, is not defined). Original music pieces have to be converted to a sequence of harmonies, to obtain a corpus that can be analyzed by standard NLP techniques. Note that one of the advantage of learning harmony is that the set of symbols is in fact very limited, it will usually be in the range of tens. We won’t have to deal with thousands of symbols as can be the case with English words.

Our system should be capable of learning, from an original corpus, the basic harmonic movement rules, and generating a new harmonic sequence according to these probabilistic rules. This paper will describe the process of building such a system. It will be organized as follows. The next section will provide an overview of the system, stating precisely what are our objectives and requirements, and what methods (NLP techniques, or others) we will use. After that, Section 3 describes the process of converting a musical work into a list of exploitable symbols. Section 4 describes how we do parameter estimation. Section 5 presents some results achieved with our system, and Section 6 briefly provides an evaluation of these results. The emphasis on this report is on the description of the mechanisms governing our experimental algorithm, not on the evaluation of preliminary results. It is our hope that the system we built can be used as a basis for the creation of a large corpus, which would allow a more precise evaluation. Future work could improve many areas of the system and allow for better results.

2 Overview of the system

2.1 Goals and specifications

Our general objectives are:

- given a particular corpus of harmonic symbols, learn the probabilistic grammar associated with that corpus
- use that probabilistic grammar to generate a new harmonic sequence, hopefully somewhat similar to the succession of harmonies present in the original corpus
- be able to generate an exploitable corpus from a given set of music data

The most important thing that we want to achieve is generality and extensibility. That means we wish to produce a system anyone can reasonably use with any kind of music data. So we decided to take as an input, for the creation of a corpus, MIDI files. MIDI files are widely available on the Internet and they contain sufficient "abstract data" to be able to produce a sequence of harmonies out of them. Still, they are far from perfect, and other music formats (Finale format for example) are closer to the standard written (notated) music used by Western composers. However files in these formats are much less available. Choosing MIDI was a reasonable compromise as MIDI files are easily generated, and we believe it is the only current choice that can allow quick creation of arbitrary corpora.

It is important to understand that for any work dealing with music composition or analysis, dealing with notated music is almost mandatory since notation simplifies the actual "acoustic" music it represents. Without that simplification, it is simply too hard to deal with music; for example, even a single musical work can sound different, depending on the player's interpretation. As written English is a simplification of speech (and the complexity of the associated sounds!), notated music is a very useful simplification of the actual music.

2.2 Methods chosen

From a general point of view, this whole project is based on the work of Ponsford et al., described in [6]. Readers are referred to that paper for some of the technical details of the algorithms described here. Ponsford et al. used an unique corpus consisting of 84 seventeenth-century French *sarabandes*; they also started with MIDI data, that they converted to Prolog first. The techniques used in this project and their work are the same, both for the preparation of harmony symbols and the NLP method used to learn a statistical grammar. The implementation is a little bit different, however, since I focused on being easily able to create a corpus directly from MIDI files with as little possible human interaction as possible. As a result I used GUIDO rather than simple Prolog data as an intermediate format (see Section 3).

In order to perform statistical learning over harmonic symbols, several choices are available, such as hidden Markov Models or context-free grammars. We will use n-grams, more precisely, trigrams. N-grams allow to capture a certain amount of context when analyzing a sequence of symbols. For trigrams more specifically, the main assumption is that every new symbol in a sequence depends on the two last symbols. So for every two given symbols appearing in the corpus, we can get a statistical distribution over the choices for the next symbol. In our case, this would mean the evolution of harmony only depends on the last two harmonies. Of course, this is a simplification, if we were to use trigrams with natural language, it is obvious that a word in a book is not only dependent on the two last words (in fact, it is probably dependent of any previous word, that is, the whole book).

We have in fact a trade-off: with longer n-grams, we can capture more context, which makes the system more realistic. However as n becomes larger, the data becomes sparser; when n is equal to the length of the piece, there is only one available n-gram consisting of the whole piece. The derived grammar obviously consists of this unique symbol and is uninteresting. Trigrams seem a good choice, given the fact that we are modeling harmonies. Such harmonies are already derived from several notes.

Parameter estimation using trigrams is described in more details in section 4.

3 Converting notated music to a sequence of harmonies: preparing a corpus from MIDI files

Building the corpus is the first thing our system should be able to do, in order to be trained on this corpus later. This first phase can be itself divided in 3 separate steps.

1. Convert from MIDI to a list of notes and their durations
2. Sample harmony at fixed-length intervals from the sequence of notes
3. Arrange harmonies, resolve dissonances

3.1 Conversion from MIDI

This step is straightforward. It involves extracting the data encapsulated in a MIDI file and convert it to a sequence of notes. Each note must also have an associated length (duration). In MIDI, each note is in fact a keystroke on a piano instrument, so the length is encoded in milliseconds. We would like that length to be converted to a standard duration for notated music (ie, quarter of a bar, etc...). In our case, this conversion is done with the help of a commercial program, Sibelius, that can import MIDI files and re-export them in GUIDO format. GUIDO is a standard format for score level music representation, more suited than MIDI for our system, and that can be easily parsed (see [5]). In fact, there are converters directly from MIDI to GUIDO, but I favored the use of Sibelius as an extra step in the conversion chain, since Sibelius MIDI import does a much better job than most other MIDI converters available.

From then, the GUIDO data is parsed (with a C++ program) and we obtain a sequence of notes and durations. We are then able to proceed to the next step.

3.2 Sampling Harmony

For this step we need to precise a "window" size, called the sampling rate. We will use that window size as the length of a time interval, and all notes inside such an interval will be regrouped together in order to derive a harmony, as shown in Figure 1. "Doubled" notes are removed.

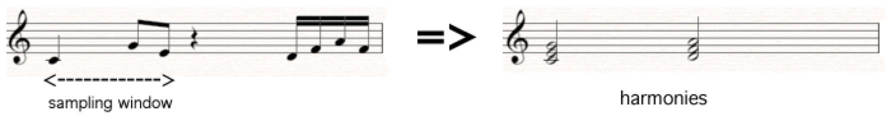

Figure 1: Sampling step: regrouping notes to form harmonies (here the sampling rate is equal to one half of a full note)

It should be noted that the choice of a sampling rate is very important. The optimal sampling rate should probably be the rate at which the smallest harmonic changes occur in the considered piece or corpus. As of now, the specification of the sampling rate requires human interaction. An algorithm guessing a plausible value for the sampling rate would be a valuable future work.

3.3 Arrangement of harmonies

An harmony is an abstract set of notes. Thus we could skip this step, leaving all the various steps of notes we obtained in the previous step as our symbols for parameter estimation. However, here we will favor "standard" harmonies over others, corresponding to the main chords on a given key (scale). They consist of triads (for the key of C, these are C-E-G, D-F-A, E-G-B, etc...). So we will try to convert most of the sets we obtained to these triads. For example, if only a single note is encountered in a given sampling time interval, two notes will be added in order to obtain a triad (if C is found, it will be completed to C-E-G). Similarly, we will resolve dissonances and seventh, removing additional notes. The algorithm used for this step is the same as in [6], and the reader is referred to that paper for details. Figure 2 describes this process.

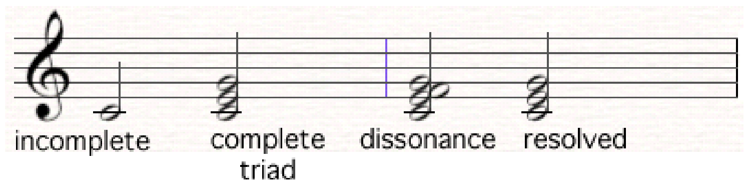

Figure 2: Normalization of harmonies

All this simplifying work has the result of diminishing the number of total "harmony symbols". This allows for a better modelization of the harmonic structure of a piece, as such normalization eliminates most uncommon harmonies and concentrate all the data into a few essential symbols. Some "strange harmonies" still remain after this step, however, they become statistically insignificant and are not a problem.

It should be noted at this point, that every piece of music considered has of course a key and a mode (major or minor). To be able to perform learning over pieces of different keys in the same corpus, every piece is converted so that the harmonies obtained are neutral with respect to key and mode. This implies that the system should be provided with a key and a mode for each piece considered.

4 Parameter estimation, grammar learning, generation

Once we are at this step, the system has nicely converted all music data into a list of different symbols corresponding to various harmonies. So in essence, these symbols are abstract and we will use standard NLP methods just as we would if we were dealing with English words. As a side note, all the algorithms described in this section were implemented in Perl, contrary to the previous step when everything was written in C++.

4.1 Parameter estimation

Parameter estimation is straightforward. The set of trigram sequences in the considered corpus is found, and counts for all these trigrams are computed. Then probabilities of a next symbol, given a context of two previous symbols, must be learned. For that, it is sufficient to get the number of times a symbol occurs after the given context (that is, in fact, the counts of trigrams), and that number is divided by the number of times the considered context appears in the corpus, to obtain the required probability.

$$Pr(symbol_a|symbol_b, symbolol_c) = \frac{\#\ occurences\ trigram_{a,b,c}}{\#\ occurences\ context_{b,c}}$$

To facilitate generation, tables of the next symbol's probabilities for all the possible contexts can be stored.

4.2 Grammar Learning

Once we have those probabilities, grammar learning cab be done. The incentive behind grammar learning is to attempt to derive rules that govern the harmonic movement of a corpus. However, in the current case, obtaining these rules doesn't guarantee we will get precious information about the music structure we are trying to learn. This is because a typical grammar is much too big and complex to understand, and the justification of relations between formal grammar rules and harmonic evolution is not self-evident. I believe it is more useful to generate new pieces in order to understand what rules govern the composition of pieces, and what is (or isn't) correctly learned by the system. So for the current project, I decided not to focus on grammar learning. The reader interested in the process of grammar generation can read Paragraph 9 of [6]. It involves converting n-grams parameters to a finite state transition network.

4.3 Generation of a new piece

To generate a new sequence of harmonies one simply has to start with two symbols forming a context appearing on the original corpus. From then, the next symbol is picked up according to the transition probabilities tables we computed (we sample a random number and choose the next symbol accordingly). This will give us another context, and this sequence is repeated as much as we want (we choose the length of the new piece).

This process is rather simple, however, and can be refined. One obvious, but essential change is to add an “end” symbol at the end of each piece in the corpus. Then we don’t have to specify a predetermined length for the generated piece, but the generation process ends when we encounter (ie, sample) an “end” symbol. This allows pieces of flexible length, and also allows the end of the generated piece to be more like the end of the pieces in the original corpus. The end of a piece is called the cadence, and is generally characteristic of a given style, so this is important (however, remember that trigrams allow us to capture a context of two symbols only: this is obviously a problem for the cadence...). Following the same principle, “start” symbols could be considered, which would remove the need for a choice of an initial context (we would just begin with two “start” symbols, this even helps capturing the structure of harmony at the beginning of a piece). Many other variations are possible for better and more realistic generation; adding symbols for the beginning or end of a musical phrase would probably add much to the depth of the generated pieces, but this would require being able to determine automatically the phrasing of musical works (some work has already been done on that are: see for example [3, 4]). Due to time constraints, advanced generation variants were not implemented and the system sticked with the standard generation method.

5 Experiments and results

Once the system was implemented, I ran several experiments with different musical pieces, thus creating various corpora. This section presents the results of one experiment dealing with a particular corpus, and the settings for that experiment. I decided that the best way to present actual results from my experiments was to reproduce, in conventional music notation, the harmonic scores for one of the generated piece (Figure 5).

The corpus was based on several songs composed by the progressive rock band Marillion. I chose to build my corpus on such musical works mainly because I like that band and not because it would be “well-suited” for my system, or “interesting” to study. In fact, the main incentive for this project was to implement a system that could generate harmonies for any music style, in order for myself (and maybe other people) to play with it and try different styles of music and just “look at the results”. I chose a sampling rate of half a full note, and the length of the generated pieces was fixed to 50. It should be noted that actually the corpus is based on very few songs, but several sequences of notes for each song were used, since each song contained scores for many different instruments (strings, piano, bass, acoustic guitar, electric guitar, vocals etc...). So in the end we ended with a reasonably large corpus (10 scores), with about 40 distinct harmonic symbols.. The size of our corpus is comparable to the one used by Ponsford et al., (they used a corpus of about 40 major pieces and 40 minor pieces, but the pieces were much shorter than the ones we consider).

Experiments dealing with only one score were also conducted, as it allowed me to test the robustness of the system, debug it, and verify that its outcome made at least some kind of sense. So a generated piece from a corpus of a single score is also given here (Figure 3), along with a score describing the "harmonic translation" of the original piece (Figure 4).

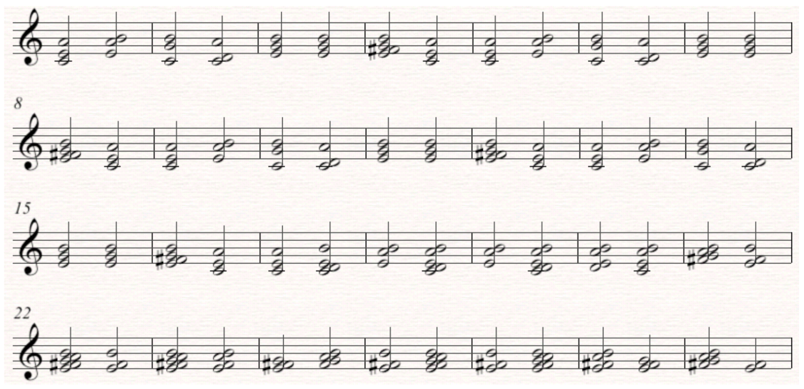

Figure 3: Harmonic translation of an original piece (part only)

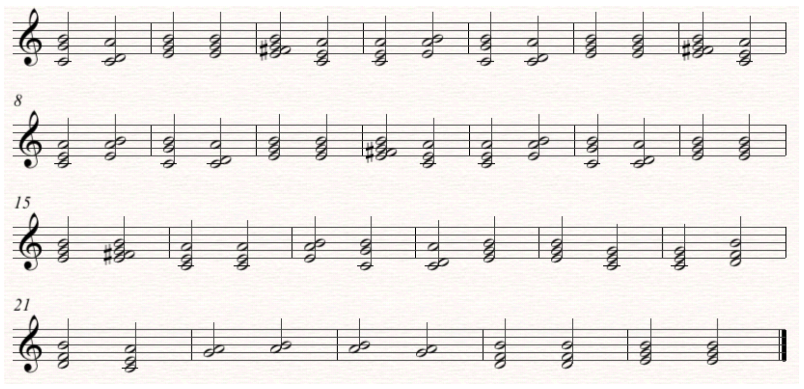

Figure 4: Generated piece, from the original piece given above

6 Evaluation of the project

The troublesome part of this project was the evaluation of the system, and of the results of whatever experiments I would run. It is quite difficult to judge the "quality" of the system I built, because the final goal of such a system is hard to define. I consider that my main objectives were reached, since the final system is able to take music data files (in MIDI format) as an input, derive their harmonies, and generate new sequences of similar harmony transitions. Regarding the generated sequences, it is also hard to objectively tell if the system learns intelligently, and to which point. I listened to a lot of produced sequences, and I can say some were quite interesting, with some nice modulations (it happened also, of course, that some of the generated pieces sounded just dull). Without doubt, the sequences obtained are somehow similar to the original ones, but I must admit that this is probably due in no small part to the fact that the same harmonic symbols are used. A better evaluation could be done by asking people their

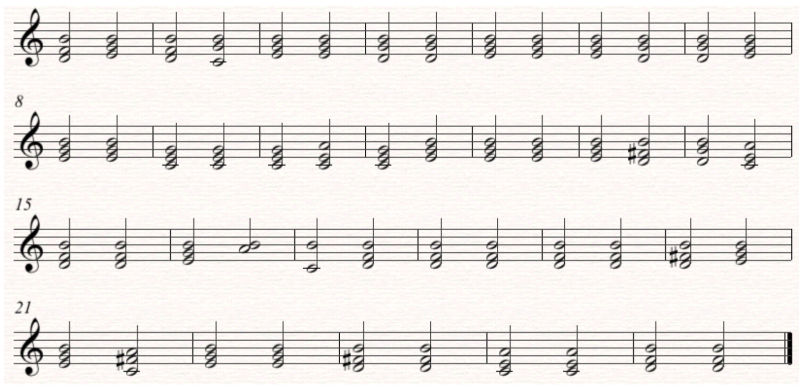

Figure 5: Generated piece, from a corpus of songs

opinions about various generated harmony sequences, derived from musical works they would choose themselves. This would be relatively easy to perform, and could point to several weaknesses of our system (that could be fixed, or not). However, we must remember that harmony is only a part of music, and by simplifying and normalizing a lot of the data present in the original scores, we make music easier to model and learn, but we also lose a lot of precious information regarding the style of the composer.

Anyway, running these experiments allowed me to learn some lessons, and get some ideas of the directions that are worthy to explore in the future. Generally speaking, I know the harmony normalization part needs to be more polished in several ways. I think this is a weak point of my system, but that could be easily fixed with more time. For example, with respect to the "key-neutrality" concept that harmonies have to respect, a score can have different parts in different keys (ie, uses more than one key). The model should allow for that possibility, since with some of the pieces I entered as input, I noticed this caused some problems. Typically, a whole part (generally the middle) of the sequence of harmonies for a piece would be wrong, full of uncommon and not normalized symbols, since at that point the piece occurred a change of key but the system didn't know about that. There are many other examples of changes (minor and major) that should be applied to the conversion and normalization steps of my system, to make it "more polished".

Much more difficult are the issues already raised in Section 4, the introduction of new symbols to enhance the learning of phrases or cadences. It is also hard to predict how much a radical change in the core structure of our model (moving from an n-gram based model to a hidden Markov model, for example) would benefit our statistical learning of harmonic changes.

7 Conclusions and further work

Basing our work on Ponsford et al. ([6]), and using the same algorithms and techniques, we implemented a system capable of learning, to a certain extent, the harmonic structure of a musical work. The results I obtained are encouraging, which is a conclusion similar to the one reached by Ponsford et al. On the short-term, a valuable addition to the current system would be to allow users to directly input MIDI files and get a generated harmonic sequence. Right now the system is still very “rough” and needs to be operated by someone who knows in details the algorithms underneath it. On a longer-term scale, I mainly consider harmonic movement learning as a “skeleton” for more in-depth research about the possibility of computer musical composition. Harmony is usually the framework of a musical work, and a composer builds everything on this framework, adding up things as melody and rhythm as the composition process goes on. If we want to achieve plausible musical composition in a totally unsupervised way, we will have to follow the same path.

References

[1] David Cope. *Computers and Musical Style*. A-R Editions, Computer Music and Digital Audio Series, Vol. 6, 1991.

[2] Shlomo Dubnov, Gerard Assayag, Olivier Lartillot, and Gill Bejerano. Using Machine-Learning Methods for Musical Style Modeling. *IEEE Computer Society*, 2003.

[3] Ferrand, Nelson, and Wiggins. A probabilistic model for melody segmentation. In *Electronic Proceedings of the 2nd International Conference on Music and Artificial Intelligence (ICMAI 2002)*, 2002.

[4] Miguel Ferrand, Peter Nelson, and Geraint Wiggins. Memory and melodic density: a model for melodic segmentation, 2003.

[5] Hoos H., Hamel K., Renz K., and Kilian J. The GUIDO Notation Format - a Novel Approach for Adequately Representing Score-Level Music. In *Proceedings ICMC*, pages 451–454, 1998.

[6] Dan Ponsford, Geraint Wiggins, and Chris Mellish. Statistical learning of harmonic movement, 1999.