Parsing the Billboard Chord Transcriptions

de Haas, W.B.; Burgoyne, J.A.

Publication date
2012
Document Version
Final published version

Parsing the Billboard Chord Transcriptions

W. Bas de Haas and John Ashley Burgoyne

Technical Report UU-CS-2012-018
December 2012

Department of Information and Computing Sciences
Utrecht University, Utrecht, The Netherlands
www.cs.uu.nl

ISSN: 0924-3275

Department of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands

Parsing the Billboard Chord Transcriptions

W. Bas de Haas* and John Ashley Burgoyne†
W.B.deHaas@uu.nl, J.A.Burgoyne@uva.nl

1 Introduction

Over the past few years, research involving chord sequences as a mid-level representation for musical content has become increasingly popular. The classical problem for music information retrieval (MIR) has been the design of algorithms that can automatically extract chord sequences from audio (e.g., Mauch, 2010). Various approaches have also been presented that use chord sequences as a primary representation for solving MIR tasks like similarity estimation or harmonic analysis (e.g., De Haas, 2012). High-quality chord sequence data is necessary for both of these research strands, but until quite recently, the amount of such data was severely limited.

In 2011, John Ashley Burgoyne, Jonathan Wild, and Ichiro Fujinaga presented the McGill Billboard data set, a set over 1000 professional chord transcriptions of popular music randomly selected from *Billboard* magazine's "Hot 100" chart between 1958 and 1991, all time-aligned with audio (Burgoyne et al., 2011). In addition to the chord transcriptions, the data include annotations about musical structure, function (e.g., verse or chorus), and instrumentation. For copyright reasons, the corresponding audio files are not available directly, but several sets of features derived from the audio, including the widely-used EchoNest features,[1] are available to download along with the annotations.

These data represent a six-fold increase in the amount of data available for research on chord sequences, but the very richness of the data set poses some challenges for usability. The transcription format used to represent the chord sequences was explicitly designed for the Billboard dataset, and aims at being both machine- and human-readable. This technical report presents the reasoning behind a new software library for parsing this format in order to make full use of the Billboard set.

*Utrecht University
†University of Amsterdam
[1] `http://developer.echonest.com/docs/v4/track.html`

2 The Billboard file format

A sample annotation in the McGill Billboard format appears in Figure 1. Each annotation begins with a header including the title of the song (prefixed by '`# title:`'), the name of the artist (prefixed by '`# artist:`'), the metre (prefixed by '`# metre:`'), and the tonic pitch class of the opening key (prefixed by '`# tonic:`'). Similar metre and tonic comments may also appear in the main body of the annotations, corresponding to changes of key or metre. In some cases, there is no obviously prevailing key, in which case the tonic pitch class is denoted '?'. The annotators were less consistent in describing the mode (major or minor), and so at the moment, there is no information about mode available. The chord annotations are linked to the matching audio files by an identifier in the filename.

The main body of each annotation consists of a single line for each musical phrase or other sonic element at a comparable level of musical structure. Each line begins with a floating-point number denoting the timestamp of the beginning of the phrase (in seconds) followed by a tab character. There are special lines for silence at the beginning and end of the audio file and a special line for the end of the piece. The other lines continue with a comma-separated list of elements among the following.

- **Capital letters**, possibly followed by an arbitrary number of primes (apostrophes), designate high-level musical structures. They appear at the beginning of each high-level musical segment and are presumed to continue until the next appearance of a capital letter. When two letters match, the two high-level segments are musically similar. Other than denoting similarity, the letters themselves have no intrinsic meaning, but for the letter 'Z'. 'Z' denotes non-musical passages in the audio such as noise or spoken words.
- **Plain text strings** denote more traditional names for musical structures, e.g., *verse*, *chorus*, and *bridge*. The vocabulary was semi-restricted: Annotators used a common vocabulary for frequently appearing structures but had the freedom to use whatever terms they felt were most appropriate for unusual contexts.

```
# title: Just The Way You Are
# artist: Billy Joel
# metre: 4/4
# tonic: D

0.0           silence
0.422267573   A, intro, | D:1 E:hdim7/b7 | D:1 . G:maj/5 D:maj | D:1 E:hdim7/b7 | D:1 . G:maj/5 D:maj |, (keyboard)
7.842154195   B, verse, | D:maj | B:min7 | G:maj7 | B:min7 D:7 |, (vocal
14.962585034  | G:maj7 | G:min7 | D:maj | A:min D:7 |
21.949160997  | G:maj7 | G:min7 | D:maj | B:min7 |
28.992108843  | E:sus4(b7) | E:7 | A:sus4(b7,9) | A:sus4(b7,9) |
35.887301587  | D:maj | B:min7 | G:maj7 | B:min7 D:7 |
42.892721088  | G:maj7 | G:min7 | D:maj/3 | A:min D:7 |
49.795396825  | G:maj7 | G:min7 | D:maj/3 | B:min7 |
56.62478458   | E:min | A:sus4(b7,9) |, voice)
60.129523809  A, interlude, | D:1 E:hdim7/b7 | D:1 . G:maj/5 D:maj | D:maj E:hdim7/b7 | D:1 . G:maj/5 D:maj |, (saxophone)
67.106031746  B, verse, | D:maj | B:min7 | G:maj7 | B:min7 D:7 |, (voice
74.243582766  | G:maj7 | G:min7 | D:maj/3 | A:min D:7 |
81.054217687  | G:maj7 | G:min7 | D:maj/3 | B:min7 |
87.926553287  | E:sus4(b7) | E:7 | A:sus4(b7,9) | A:sus4(b7,9) |
94.748526077  | D:maj | B:min7 | G:maj7 | B:min7 D:7 |
101.543832199 | G:maj7 | G:min7 | D:maj/3 | A:min D:7 |
108.513106575 | G:maj7 | G:min7 | D:maj/3 | B:min7 |
115.351179138 | E:min | A:sus4(b7,9) |, voice)
118.809206349 A, interlude, | D:1 E:hdim7/b7 | D:1 . G:maj/5 D:maj | D:1 E:hdim7/b7 | A:min7 D:7 |, (saxophone)
125.901043083 C, bridge, | G:maj7 | A:7 | F#:min7 | B:sus4(b7) |, (voice
132.871791383 | E:min7 | A:7 | D:maj | D:maj D:7/b7 |
139.776303854 | Bb:maj7 | C:7 | A:min7 | D:7 |
146.563129251 | G:min7 | G:min7/11 | G:maj/9 | G:maj/9 |
153.481836734 B, verse, | D:maj | B:min7 | G:maj7 | B:min7 D:7 |
160.4992517   | G:maj7 | G:min7 | D:maj/3 | A:min D:7 |
167.353605442 | G:maj7 | G:min7 | D:maj/3 | B:min7 |
174.242607709 | E:min | A:sus4(b7,9) |, voice)
177.600181405 A, interlude, | D:1 E:hdim7/b7 | D:1 . G:maj/5 D:maj | D:1 E:hdim7/b7 | A:min7 D:7 |, (saxophone
184.734058956 B, solo, | D:maj | B:min7 | G:maj7 | B:min7 D:7 |
191.658276643 | G:maj7 | G:min7 | D:maj/3 | A:min D:7 |
198.634195011 | G:maj7 | G:min7 | D:maj/3 | B:min7 |
205.497528344 | E:sus4(b7) | E:7 | A:sus4(b7,9) | A:sus4(b7,9) |, saxophone)
212.331269841 B, verse, | D:maj | B:min7 | G:maj7 | B:min7 D:7 |, (voice
219.259229024 | G:maj7 | G:min7 | D:maj/3 | A:min D:7 |
226.146145124 | G:maj7 | G:min7 | D:maj/3 | B:min7 |
233.337800453 | E:min | A:sus4(b7,9) |
236.922675736 C', outro, | Bb:maj | C:maj | A:min7 | D:7 |, voice)
243.918526077 | G:min7 | A:7 |, (saxophone
247.366848072 B, outro, | D:maj | B:min7 | G:maj7 | B:min7 D:7 |
254.31893424  | G:maj7 | G:min7 | D:maj/3 | A:min D:7 |
261.134852607 | G:maj7 | G:min7 | D:maj/3 | B:min7 |
268.044013605 | E:sus4(b7) | E:7 | A:sus4(b7,9) | A:sus4(b7,9) |, fadeout
274.881247165 | D:maj | B:min7 | G:maj7 | B:min7 D:7 |
281.660566893 | G:maj7 | G:min7 | D:maj/3 | A:min D:7 |, saxophone)
289.208888888 silence
290.79510204  end
```

Figure 1: A sample annotation from the McGill Billboard set.

Chord Type	Shorthand	Components
power chord	5	(5)
suspended 2nd	sus2	(2,5)
major 11th	maj11	(3,5,7,9,11)
minor 11th	min11	(b3,5,b7,9,11)
dominant 11th	11	(3,5,b7,9,11)
major 13th	maj13	(3,5,7,9,11,13)
minor 13th	min13	(b3,5,b7,9,11,13)
dominant 13th	13	(3,5,b7,9,11,13)

Table 1: Extensions to the MIREX chord syntax as used for annotations in the Billboard set.

- **Chord annotations** appear as series of bars flanked by pipes ('|'). A phrase may by followed by an 'x' and an integer, which means that the phrase is repeated that number of times. A phrase may also be followed by an arrow ('->'), which is a musicological hint that the phrase is musically elided into the following phrase.

- **Leading instruments** are noted in songs where there is a notable deviation from the norm of a leading vocal throughout the entire song. They appear as text strings preceded by a left parenthesis ('(') in the segment where the instrument comes to prominence and as text strings succeeded by a right parenthesis (')') in the segment where that instrument fades from prominence. If an instrument is prominent for a single segment only, its name appears with both left and right parentheses.

More detail on the structural annotations is available in (Smith et al., 2011). Essentially, these annotations replace the lower level of structural annotations (lowercase letters) from this reference with chord annotations.

Although there are some passages that have chord annotations at the eighth-note level, in general the chord annotations are simplified to the beat level. All chord symbols follow the standard presented at ISMIR 2005 and used in MIREX[2] since (Harte et al., 2005; Downie, 2008), with a few additions to the shorthand to facilitate the richness of these annotations: '5' for power chords, and 'sus2', 'maj11', '11', 'min11', 'maj13', '13', and 'min13' for the corresponding chords in traditional jazz notation Burgoyne (2012). Table 1 provides the theoretical components for these chords in MIREX-style notation, although users should be warned that in practise, jazz musicians frequently omit some of the components of 11th and 13th chords. An additional pseudo-chord shorthand of '1' denotes pure bass notes with no chord on top (without such a shorthand, they could be confused for major chords).

[2] http://www.music-ir.org/mirex/wiki/MIREX_HOME

In order to save space, repeated chords are denoted with a dot instead of the full chord name. To further save space, bars containing a single chord on all beats list the chord symbol only once; likewise, in quadruple metres (4/4 or 12/8), bars with only two chords and the change on the third beat list those two chords with no dots. For brief changes of metre, the metre may appear in parentheses at the beginning of the bar rather than as a full metre comment.

Two non-chord symbols may appear within bars. For passages that were too musically elaborate to merit beat-level chord annotations, annotators sometimes filled the bar with an asterisk ('| * |'). For brief pauses of arbitrary length (often a single beat), annotators added a bar with the special annotation '&pause'.

3 The Billboard parser

We present the billboard-parser program and library.[3] The parser for the Billboard data has been developed in the functional programming language Haskell (Peyton Jones, 2003),[4] and relies on the uu-parsinglib parser combinator library (Swierstra, 2009).[5] A particularly attractive feature of this library is that it features error correction, which, in case of input that does not match the format, allows the parser to finish the parse and output detailed information on where the parsing failed.

The interface of the billboard-parser program is straightforward. Currently, the parser features five modes, parse, mirex, full, test, or result, that can be selected with the mode of operation flag --mode <mode>. With the first mode, parse, one or more pieces are parsed, and the piece is printed to the user in a way that resembles the original. If a piece cannot be parsed correctly, the parser will inform the user which part of the input could not be parsed, and where this occurred exactly in the file. The second mode, mirex, converts billboard pieces into the format typically used in MIREX evaluations: 'onset <space> offset <space> chordlabel'. In MIREX mode, the chord labels are also truncated to the triad shorthands ('maj', 'min', 'dim', 'aug', 'sus2', and 'sus4'); all additional chord extensions are removed. We truncate a chord by expanding it to its corresponding list of intervals, and analyse the components, if any, that are a second, third, fourth, or fifth above the root. If this reduced set of components fails to match any of the truncated shorthands (for instance, in the case of a power chord or a 7(♭5) chord),

[3] http://hackage.haskell.org/package/billboard-parser

[4] www.haskell.org

[5] http://hackage.haskell.org/package/uu-parsinglib

the chord label is replaced by 'X', a special label that the MIREX evaluation routines are designed to ignore.

The parser supports a few other modes. The third mode, `full`, uses the same format as the `mirex` mode, but it prints the full chordlabel as found in the data. The fourth mode, `test`, executes a series of unit tests that flag chords with an unexpected duration. Finally, the mode `result` will output the results presented in Section 5.

The `billboard-parser` can parse single files by using the flag `--file <path>`, or batch process a directory with billboard files by using the `--dir <path>` flag. In the latter case, the user should point to a location where the Billboard collection is stored. The parser will look for a file named `salami_chords.txt` inside a folder containing exactly four digits (which are used as identifiers). We also implemented an alternative way of pointing to a file by providing the path to the Billboard dataset (with `--dir <path>`) and a Billboard identity, i.e., the unique folder number, with the flag `--id <integer>`. If the `mirex` mode is used in combination with the `--dir <path>` flag a file `mirex_chords.txt` is written to the folder that also contains the source file. Optionally, a user can set a specific output folder with the flag `--out <path>`.

Because we use an eighth-note grid, the generated output can be rather verbose. To compress the generated output, we added another flag to our interface to reduce the output chord sequence. When the `--comp reduce` switch is added, the `billboard-parser` will merge all subsequent chords that do not contain additional information, like bar lines, structural annotations, instrumentation etc. If the user adds a flag `--comp expand`, the chord sequence is not reduced, and the all eight note grid positions are printed. By default all output is reduced.

Besides a program, the `billboard-parser` is also a software library. If you are familiar with the Haskell programming language, you can import this library and use the internal representation of a billboard song for further computing. Internally, we use a datatype to bundle the metadata and the chords. The chords are stored as a list of chord datatypes that store information about the chord, the onset and offset, lead instrument labels, and structural segmentation information at every eighth note position.[6]

In the parsing process, all dots ('`.`') are transformed into the chords they represent. If a meter change is encountered, it is parsed on-the-fly, and the result is used in the remainder of the parsing process. Naturally, if the parser comes across a repetition, the involved chords sequence is automatically expanded. The textual structural segmentation and instrumentation annotations are categorised, and the starting and ending positions are stored in the chord datatype. Musical passages not containing harmonic information that can be transcribed at the beat level ('`*`', '`&pause`', 'Z', and '`silence`', etc.) are internally represented as a *non-chord* annotations. Within MIREX, an '`N`' annotation is generally used to denote such passages.

[6] See the Haskell API for details: `http://hackage.haskell.org/package/billboard-parser`

4 Interpolation based audio alignment

Generally, parsing the format described in Section 2 is straightforward, but the alignment of the chords with the audio deserves some further explanation. Because annotating the exact starting and ending positions of the chords would have been very time consuming, it was a deliberate decision to add timestamps only at the beginning of every chord annotation line. This decision implies the assumption that the tempo does not fluctuate a lot within the song, but we believe that this is a justifiable assumption because the dataset consists of pop music only (see Scheirer, 1998). Thus, in order to be able to pinpoint the exact starting and ending positions of individual chords, the timestamps must be interpolated based on the timestamps at the start of each line and the metrical structure of the chords sequence.

In general, chords are annotated at every beat, but due to metre changes, occasionally it is only possible to keep a consistent tempo at the eighth-note level. Hence, we interpolated all chord annotations at the eighth-note level. As a side effect, the MIREX output contains chords and timestamps for every eighth-note position as well, making these files rather long. Hence, we added the chord sequence compression explained in the previous section.

For measuring how well the automatic interpolation does its job, we implemented a unit test that checks whether there are eighth-note time frames that are considerably longer or shorter than the average length of a frame. The *acceptable deviation* is a parameter to the test, which we set to 7.5%: frames with a duration 7.5% longer or shorter than the average eighth note within a piece are reported to the user. We ignore non-chord annotations because non-harmonic sections are not necessarily expected to align with the eighth-note grid.

The acceptable deviation test identified a considerable flaw in our interpolation strategy: at the beginning and endings of some songs, we encountered unexpectedly long frame lengths. At the beginning of songs, the most common reason the test was triggered was the presence of the so-called *slow introduction*, a common stylistic feature whereby the first few phrases of a song are at a markedly slower tempo (Burns, 1987). Occasionally there were also

some problems due to variation in how annotators handled pickup beats. At the end of a song, annotators marked '`silence`' when there was nothing to hear any more. Due to a fade-out or the decaying sound of the last chords, for certain pieces there was a discrepancy between the timestamp of the last chord and the timestamp of the annotated silence, which distorted the interpolation of the last line of chords.

To overcome improper alignments at the beginning or ending of a song, we redesigned the interpolation for the first and last line of chords. For the misalignment of the last line of chords, we used the average beat length of the penultimate line to predict the beat durations of the chords on the last line and filled the remaining gap between the last chord and the silence annotation with additional non-chords. In the case of problems with the pickup beats, the misalignment of the first line of chords was corrected by applying the same function, but with the chord sequence reversed.

5 Evaluating the alignment quality

It was difficult to estimate automatically how well the chords transcriptions were aligned to the audio by the `billboard-parser`. To ensure that most of the alignments were correct, we manually checked the alignment of songs that failed the acceptable deviation test by hand. We also evaluated the MPTrEE chord transcription algorithm (De Haas et al., 2012) on the first tranche of 649 songs, and identified a couple of misalignments. Furthermore, this evaluation showed that for some pieces the tuning was off the mark. Both the misalignments and the tuning problems were corrected as much as possible. Although we were surprised how well the automatic alignment worked for most pieces, it was impossible to get both the alignment and the tuning perfect at all eighth note frames in every piece. Some commercial recordings are notoriously tuned almost exactly between two keys at standard pitch (A4 = 440 Hz), for example, Jimmy Gellis's 'Stand By Me', and many songs have at least one phrase with an abrupt *ritardando* or lengthy pause that no interpolation strategy could solve. Nonetheless, as we listened to all of the pieces that triggered the unit test alongside a 'click track' with the interpolations, it was remarkable how well it worked most of the time.

To give the reader some idea of the amount of deviation in the alignments, we measure the number of eighth note positions in the first tranche of 649 songs that fail our deviation test at five different levels of eighth note deviation: 5%, 7.5%, 10%, 15%, and 25%. These numbers are displayed in Table 2. With our interpolation strategy less then 1 percent of the eighth note positions, averaged over all pieces, deviated more than 10% of the average eighth note length of a song. It might be good to realise that for a piece played at 120 beats per minute an eighth note is only 0.25 seconds long, and a 0.025 second deviation is smaller than frequently used signal processing frame lengths.

Deviation	5%	7.5%	10%	15%	25%
Misalignments	3.58%	1.38%	0.95%	0.62%	0.47%

Table 2: The percentage of eighth note grid positions that deviate more than the above mentioned deviation, averaged over 649 Billboard pieces.

6 Concluding remarks

In this technical report we have presented the `billboard-parser` program and library, which reads and transforms the McGill Billboard chord transcriptions. We described the Billboard file format, explained how the `billboard-parser` parses the chord annotations, how it aligns these annotations to their audio, and how it can be used create MIREX compatible ground-truths.

Every set of ground truth contains errors, and likewise our alignment between audio and chords is not perfect. Nevertheless, we believe that the alignment is very good in general. Most of the remaining mis-alignments are caused by sudden, dramatic changes in tempo that would require laborious manual annotations of the timestamps for every eighth note. Moreover, we expect that the effect of such mis-alignments is marginal in benchmarking competitions like MIREX because all algorithms are affected equally (in contrast to a strategy relying on automatic beat-tracking, which would heavily favour chord-recognition algorithms that used the same beat tracker). Overall, we are convinced the `billboard-parser` is a practical and high quality tool that will aid in fulfilling the full potential of the Billboard dataset.

Acknowledgements

W. Bas de Haas is supported by the Netherlands Organization for Scientific Research, NWO-VIDI grant 276-35-001, and John Ashley Burgoyne is supported the NWO-CATCH grant 'Cognition-Guided Interoperability Between Collections of Musical Heritage (COGITCH)'.

References

Burgoyne, J. A. (2012). *Stochastic Processes and Database-Driven Musicology*. PhD thesis, McGill

University, Montréal, Québec, Canada.

Burgoyne, J. A., Wild, J., and Fujinaga, I. (2011). An expert ground truth set for audio chord recognition and music analysis. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, pages 633–638.

Burns, G. (1987). A typology of 'hooks' in popular records. *Popular Music*, 6(1):1–20.

Downie, J. S. (2008). The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255.

De Haas, W. B. (2012). *Music information retrieval based on tonal harmony*. PhD thesis, Utrecht University.

De Haas, W. B., Magalhães, J. P., and Wiering, F. (2012). Improving audio chord transcription by exploiting harmonic and metric knowledge. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, pages 295–300.

Harte, C., Sandler, M., Abdallah, S., and Gómez, E. (2005). Symbolic representation of musical chords: A proposed syntax for text annotations. In *Proceedings of the 6th International Society for Music Information Retrieval Conference (ISMIR)*, pages 66–71.

Mauch, M. (2010). *Automatic Chord Transcription from Audio Using Computational Models of Musical Context*. PhD thesis, Queen Mary University of London.

Peyton Jones, S. (2003). Haskell 98 language and libraries: the revised report. *Journal of Functional Programming*, 13(1):7–255.

Scheirer, E. (1998). Tempo and beat analysis of acoustic musical signals. *Journal of the Acoustical Society of America*, 103(1):588–601.

Smith, J. B. L., Burgoyne, J. A., Fujinaga, I., De Roure, D., and Downie, J. S. (2011). Design and creation of a large-scale database of structural annotations. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, pages 555–560.

Swierstra, S. D. (2009). Combinator parsers: a short tutorial. In Bove, A., Barbosa, L., Pardo, A., and Sousa Pinto, J., editors, *Language Engineering and Rigorous Software Development*, volume 5520 of *Lecture Notes in Computer Science*, pages 252–300. Spinger.