Recommending Songs to Music Learners based on Chord Content

Johan Pauwels [1] **György Fazekas** [1] **Mark B. Sandler** [1]

Abstract

Music learners looking for practice material to play along with are not served well by the current search interfaces for large music collections. While it is easy to find specific songs using metadata or audio fingerprinting, discovering new music based on musical content is hard. In this paper, we'll look at the challenges that arise when creating a search interface that allows to query for songs based on chord content. Specifically, we'll discuss different ways of fulfilling queries and how imperfect chord transcriptions resulting from the automatic estimation process are handled.

1. Introduction

In order to improve as a musician, regular practice cannot be avoided. In a typical educational context, music students are presented with new material in class and are then expected to rehearse on their own by repeating a small number of pieces. The reason for this limited number of pieces is purely practical. It is hard to find suitable practice material that is representative for a particular technique or musical concept without resorting to composing pieces especially for that purpose. However, repeating the same pieces over and over again can make practising monotonous and disengaging. At the same time, presenting the technique under study in more and varied contexts is likely to improve the effectiveness of the rehearsal.

For these reasons, we are investigating how songs can be recommended to music learners based on their choice of a set of chords as input. Chords are chosen for querying because they are musically speaking on a high enough level of abstraction such that multiple songs can match a query and because they make multiple practice scenarios possible. Users can for instance play along with the chords to practice transitioning between them or improve their improvisation skills. We believe this to be a novel type of song recommendation, as it is traditionally based on lower-level features or collaborative filtering (Celma, 2010). Meanwhile, other applications of chord recognition are song-centric, meaning that you first decide which song you want to learn and then retrieve its chords, without chord-based recommendation (de Haas et al., 2014).

[1]Centre for Digital Music, Queen Mary University of London, UK. Correspondence to: Johan Pauwels <j.pauwels@qmul.ac.uk>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

2. Creating a Chord Transcriptions Database

To create a sufficiently large database, 99 960 songs from Jamendo, a Creative Commons music platform, were analysed by the algorithm presented by Pauwels et al. (2017). More specifically, of the nearly 200K tracks offered through their API, we processed those that are deemed of sufficient quality to be actively promoted by Jamendo Licensing. This curation is based purely on recording quality, not artistic merit, in order to remove tracks with clipping or bad mixes from the radio stations offered to stores.

The size of the chord vocabulary is 60, obtained by combining 12 possible roots with 5 chord types (maj–min–7–maj7–min7). In addition to the estimated chord sequences, the algorithm also returns a confidence value. The latter ranges between 0 and 1 with a mean of 0.70 ± 0.12. This will be used to deal with imperfect chord transcriptions. The algorithmic output is used to populate a MongoDB, a type of document-oriented database that allows nested structures. We add a supplementary data field *chordRatio* that contains the proportion of each distinct chord per file, which will be used to facilitate query matching. The final document structure then looks like the one below.

```
{_id: 3,
 chordRatio: {Cmaj: 0.45, Gmaj: 0.55},
 chordSequence: [
  {start: 0.00, end: 2.75, label: 'Gmaj'},
  {start: 2.75, end: 5.00, label: 'Cmaj'}],
 confidence: 0.85,
 duration: 5.00}
```

3. Querying a Chord Transcriptions Database

We will recommend songs from the database based on the input, which is a bag or set of chords where order doesn't matter. The *chordRatio* structure is all that is needed to match a query, although it's the chord sequence that will be returned to the users such that they can play along with it.

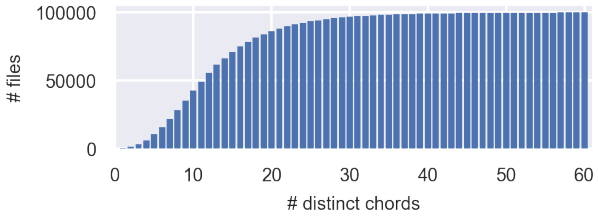

Figure 1. Cumulative number of files per number of distinct chords

Fulfilling a chord query can be interpreted in many possible ways though. Since a chord only appears in the *chordRatio* dictionary when it is recognised somewhere in the audio file, a first naive approach would be to check *chordRatio* for any or preferably all chords in the query. However, this would also return songs in which the requested chords account only for a small part of the total duration.

Therefore we first sum for the proportion of time each of the chords in the query is being played (retrieved from the *chordRatio* dictionary) to get the *coverage* of the query per track. We then impose a minimum coverage, where we can trade off the number of returned music pieces for the presence of unrequested chords in the transcription. Finally, the results are sorted in decreasing order of confidence and a confidence threshold can optionally be applied to trade off the quantity of the results for quality.

A minimum coverage of 100% is likely to give the best user experience, as it avoids returning unrequested chords to the user entirely. Depending on the practice scenario, this might even be required. For instance, if beginners specify all chords they know as input, it would be unacceptable to return pieces containing any other chord, as they don't know how to play it. The drawback of such a strict requirement is that it can drastically reduce the number of files under consideration when the number of chords in the query is low, as is evident from figure 1 where the cumulative number of files is shown as a function of the number of distinct chords.

An unintended side-effect of sorting the results by confidence, is that the top suggestion is often a short, simple song, more of a jingle, that contains only a single chord. The reason is that the longer a song, the more likely an error is to happen at some point during the chord recognition, which will consequently lower the chord confidence. Furthermore, the required minimum coverage might as well be achieved by just a subset of the chords in the query, with some of the requested chords not appearing in the returned files at all. For our intended use-case, we consider this more acceptable than presenting a user with unrequested chords, but it is nonetheless undesirable. Therefore we additionally calculate the number of query chords that are present in the results, and order them first according to this number before ordering and filtering the results by confidence. Single chord songs thus get demoted to a lower rank.

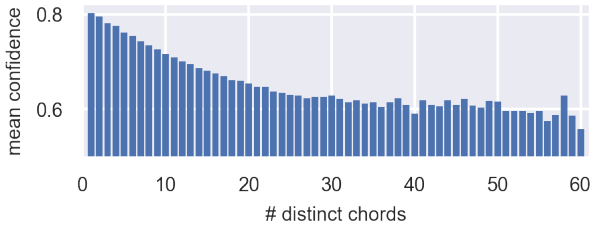

Figure 2. Average confidence per number of distinct chords

4. The Self-Regularising Effect of Queries

One interesting phenomenon we noticed while experimenting, is that even when the results are not sorted or filtered by confidence, typical queries tend to return pieces for which the chord confidence is relatively high. We call this the self-regularising effect of queries with respect to confidence. An explanation can be found in the nature of chord recognition errors and in the number of chords in a typical query. Although errors made by an automatic chord recognition system are not entirely random, but often harmonically related, it is still common that erroneously recognised chords do not legitimately appear anywhere else in the piece. Consequently, the number of distinct chords increases with the number of errors. Because the confidence is designed to be inversely proportional to the quality of a transcription, this effect is also noticeable in the average confidence per number of distinct chords in a piece, as shown in figure 2.

During a small experiment where we invited 7 participants to test the system, the number of chords in a query ranged from 1 to 24, with a mean of 6.81 ± 5.74, which is significantly lower than the average number of distinct chords per file in the database, 13.11 ± 7.44. Since the query resolution promotes files where the number of distinct chords equals the number of chords in the query and the highest quality transcriptions are concentrated around lower numbers of distinct chords, queries of a typical size automatically return files with a confidence that is higher than the average.

5. Conclusion and Future Work

In this paper, we explored how to recommend songs to music learners by querying a large database of automatically generated chord transcriptions. We discussed different ways of fulfilling queries and the reasoning behind them. Incorrectly recognised chords are handled by a confidence measure, while we also noted a tendency of self-regularisation for the most common queries. In the future, we'd like to investigate alternative use-cases and their queries, for instance suggesting the most useful chord to learn next or auto-completing queries based on commonly occurring chord combinations. It would also be interesting to consider adding a controlled amount of randomness such that repeated queries give different results each time.

Acknowledgements

This work has been partly funded by the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/L019981/1 and by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382.

References

Celma, Òscar. *Music Recommendation and Discovery: The Long Tail, Long Fail, and Long Play in the Digital Music Space*. Springer-Verlag Berlin Heidelberg, 2010. doi: 10.1007/978-3-642-13287-2.

de Haas, W. Bas, Magalhães, José Pedro, ten Heggeler, Dion, Bekenkamp, Gijs, and Ruizendaal, Tijmen. Chordify: Chord Transcription for the Masses. In *Proceedings of the 13th International Conference on Music Information Retrieval (ISMIR) Late Breaking and Demo Session*, 2014.

Pauwels, Johan, O'Hanlon, Ken, Fazekas, György, and Sandler, Mark B. Confidence Measures and Their Applications in Music Labelling Systems Based on Hidden Markov Models. In *Proceedings of the 18th Conference of the International Society for Music Information Retrieval (ISMIR)*, pp. 279–285, 2017.