Harmonic Analysis of Jazz MIDI Files Using Statistical Parsing

Music and Machine Learning Workshop, Edinburgh

Mark Granroth-Wilding and Mark Steedman

30 June 2012

I took part in the *5th International Workshop on Machine Learning and Music*, 2012. This is the talk I gave, now in article form (including the slides from the talk).

Abstract

Harmonic analysis involves identifying hierarchical structure, similar to that found in the syntax and semantics of language, in the harmonic progressions underlying tonal melodies. In previous work, we have used grammar-based parsing, with related machine-learning techniques, for automatic harmonic analysis of jazz chord sequences.

We now turn to the harder task of harmonic analysis of jazz MIDI performances using similar techniques. First, we evaluate a strict pipeline approach: we use an HMM to perform chord recognition and our previous system to analyze the output. Then, we use the chord recognizer to propose a chord lattice and analyze this using an adaptation of the previous system.

Contents

1 Introduction **2**

2 Harmonic Analysis **3**

3 Parsing Chords **6**

4 Parsing MIDI **9**

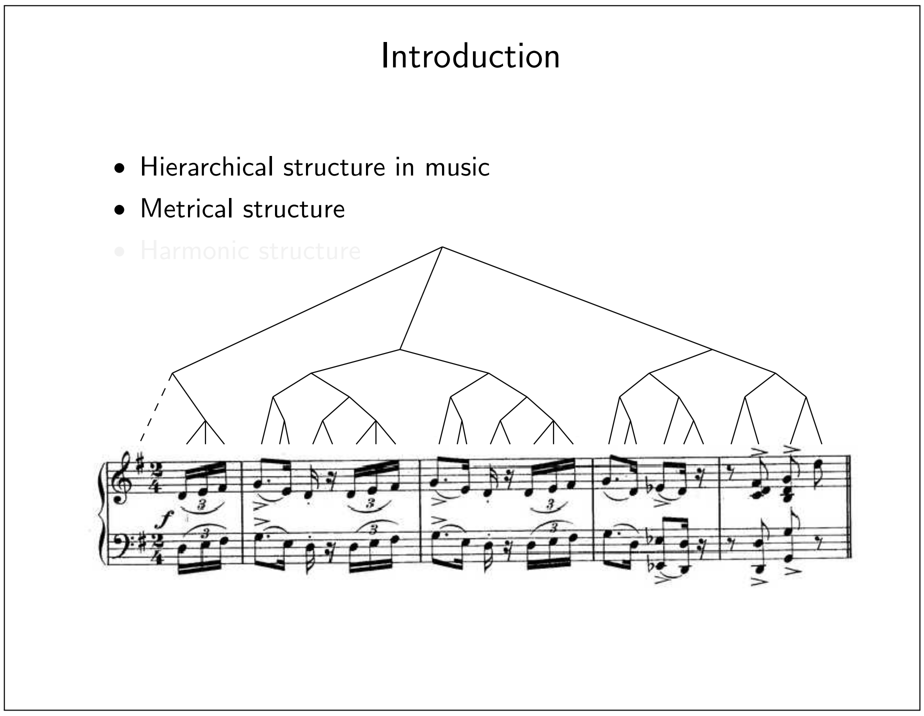
4.1 Pipeline System 12

4.2 Lattice System 15

5 Conclusion **17**

1 Introduction

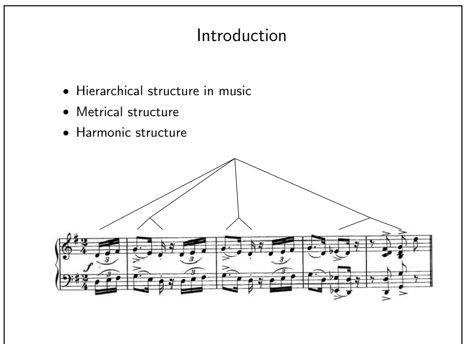
In this article, I present our work on the application of statistical parsing techniques from natural language processing (NLP) to music processing. I first introduce our approach automatic harmonic analysis using a musical grammar and I present some results from a system for harmonic analysis of chord sequences. Then I move on to the harder task of harmonic analysis of note data and show some results from a couple of proof-of-concept systems on this task.



Introduction

- Hierarchical structure in music
- Metrical structure
- Harmonic structure

Several aspects of music are organised hierarchically. One example is the metrical structures underlying rhythmic patterns (such as that shown on top of the score above). Another is the harmonic progressions underlying melodies and polyphony (a simple example can be seen above the score to the right).

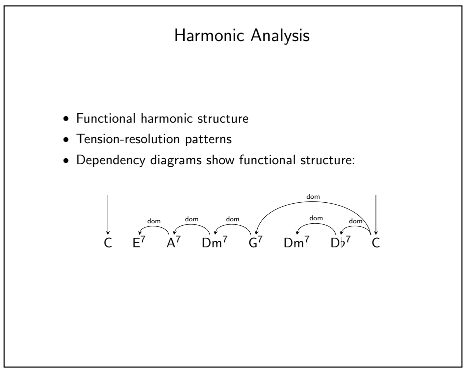


Introduction

- Hierarchical structure in music
- Metrical structure
- Harmonic structure

Identifying these structures is an important part of understanding music for a human listener and automatically identifying them is useful for many music processing tasks. Our work focuses on the automatic analysis of harmonic structure.

2 Harmonic Analysis

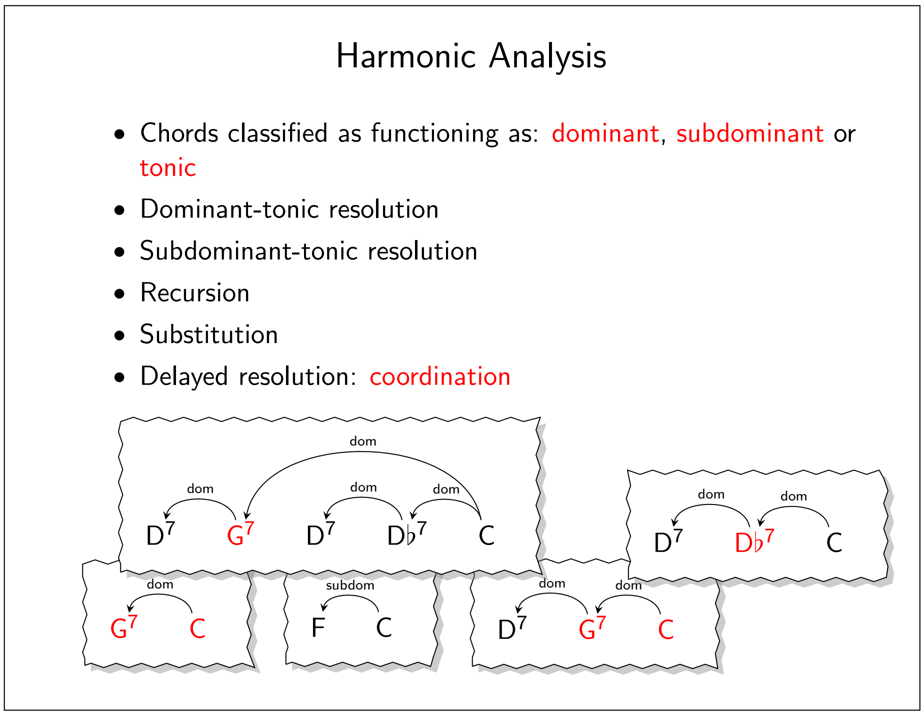


Harmonic Analysis

- Functional harmonic structure
- Tension-resolution patterns
- Dependency diagrams show functional structure:

Harmonic analysis is the task of identifying the functional harmonic structure that underlies a piece of music. This structure consists of patterns of tension and resolution. We can think of these patterns in terms of tension-resolution dependencies between chords: that is, pairs of chords in which the first raises a tension and the second provides the resolution.

We can therefore use a representation similar to that used to represent liguistic dependencies to express a harmonic analysis. An example dependency graph can be seen here and I will come back to explain this in more detail shortly.



Harmonic Analysis

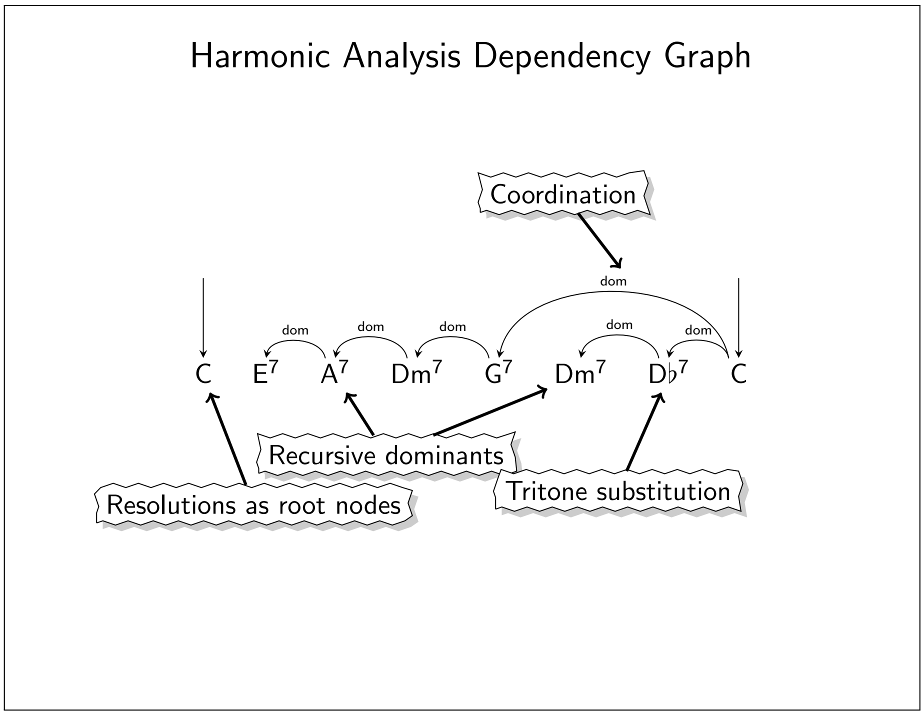
- Chords classified as functioning as: dominant, subdominant or tonic
- Dominant-tonic resolution
- Subdominant-tonic resolution
- Recursion
- Substitution
- Delayed resolution: coordination

In our analyses, each chord is classified as functioning as a dominant, a subdominant or a tonic. The most common type of tension-resolution pattern is the dominant-tonic resolution. The resolution to a dominant-function chord is found a perfect fifth below it (as in the G^7 C example). The subdominant-tonic pattern works in the same way, but with a different resolution (a perfect fourth down, as in the F C example).

These relations are subject to recursion. The resolution of a dominant-function chord can itself be a dominant-function chord, requiring in turn its own resolution (see D^7 G^7 C).

The phenomenon of substitution occurs when a chord, in some particular context, is replaced by another chord that serves the same harmonic function. In the example shown here, the D$\flat^7$ serves as a substitute for a G^7 – the so-called *tritone substitution*, common in jazz.

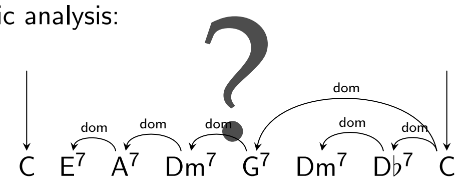
The structures get really interesting when the resolution of a tension chord is delayed. Another sequence of tension-resolutions intervenes and the eventual resolution is shared by the two structures. In the example, the G^7 chord's resolution is delayed and eventually comes in the form of the C chord, which is also the resolution to the D$\flat^7$. We refer to this as *coordination*, by virtue of its similarity to certain types of coordination in language.



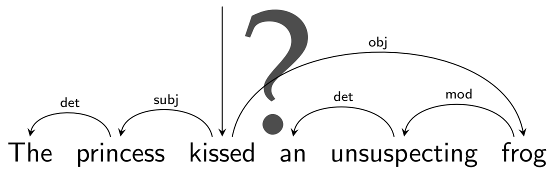
We can now return to the example graph we saw above and make a bit more sense of it. Each final resolution (tonic) is marked as being connected to the root of the graph, denoted by the downward arrows. There are several examples of recursive dominant-function chords, a tritone substitution and a coordination in this example.

Parsing

- Harmonic analysis:



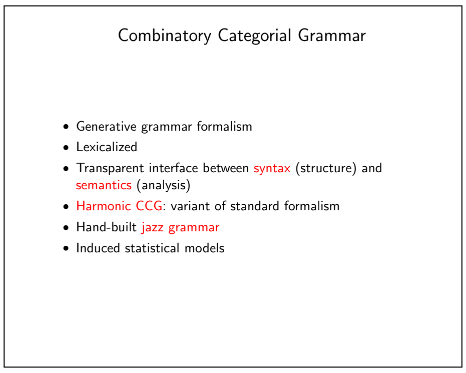
- Parsing language:



To perform harmonic analysis, we are interested in taking an unstructured musical signal – in this case a chord sequence, but it could equally be a note stream or audio signal – and producing a structured interpretation. In NLP, *parsing* is used to produce some structured analysis of an unstructured input, such as the predicate-argument structure of a sentence. The obvious question that arises, therefore, is:

To what extent can we apply parsing techniques from NLP to the task of harmonic analysis of music?

3 Parsing Chords

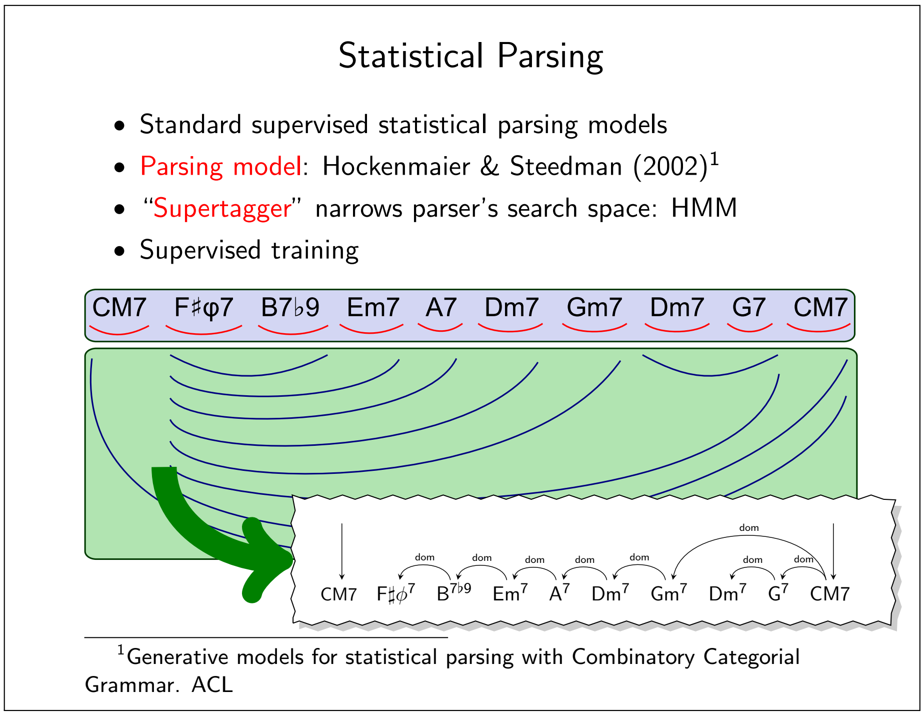


Combinatory Categorial Grammar

- Generative grammar formalism
- Lexicalized
- Transparent interface between syntax (structure) and semantics (analysis)
- Harmonic CCG: variant of standard formalism
- Hand-built jazz grammar
- Induced statistical models

We tackle this problem using the generative grammar formalism Combinatory Categorial Grammar (CCG). CCG is a lexicalized formalism: rich information is encoded in syntactic types that are associated with individual words (or in our case chords) and a small set of grammatical rules is used to combine these into a full interpretation. One of its attractive features is that it provides a transparent interface between the syntax (that is, structure) and the semantics (the actual analysis we wish to produce).

We have developed a variant of the standard CCG formalism to express the syntax of tonal harmony. Using this formalism, we have built by hand a grammar aimed at interpreting the harmony of jazz standards. This grammar may be supplemented by induced statistical models to assist with parsing.



Statistical Parsing

- Standard supervised statistical parsing models
- Parsing model: Hockenmaier & Steedman (2002)[1]
- "Supertagger" narrows parser's search space: HMM
- Supervised training

CM7 F♯φ7 B7♭9 Em7 A7 Dm7 Gm7 Dm7 G7 CM7
dom
dom dom dom dom dom dom dom
CM7 F♯ø7 B7♭9 Em7 A7 Dm7 Gm7 Dm7 G7 CM7

[1]Generative models for statistical parsing with Combinatory Categorial Grammar. ACL

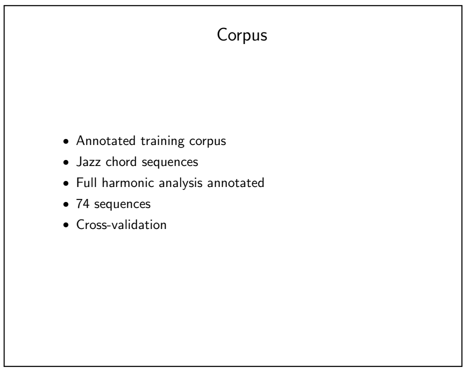
We have applied standard supervised statistical models used for chart parsing with CCG in NLP. Our input is a chord sequence. A parsing model is used to guide the parser and rank the (usually numerous) interpretations it produces. Our parsing model is identical to that of Hockenmaier & Steedman (2002).

Supertagging is a technique commonly used in NLP for statistical parsing with lexicalized grammars. A sequence model is used to reduce the number of lexical categories (interpretations of individual words) on the basis of a word's local context. The parsing model is then used to decide between the remaining categories using higher-order statistics over the grammatical derivation. We use an HMM as a model for a supertagger component.

The result of parsing is a harmonic analysis of the input of the sort that we have seen above.

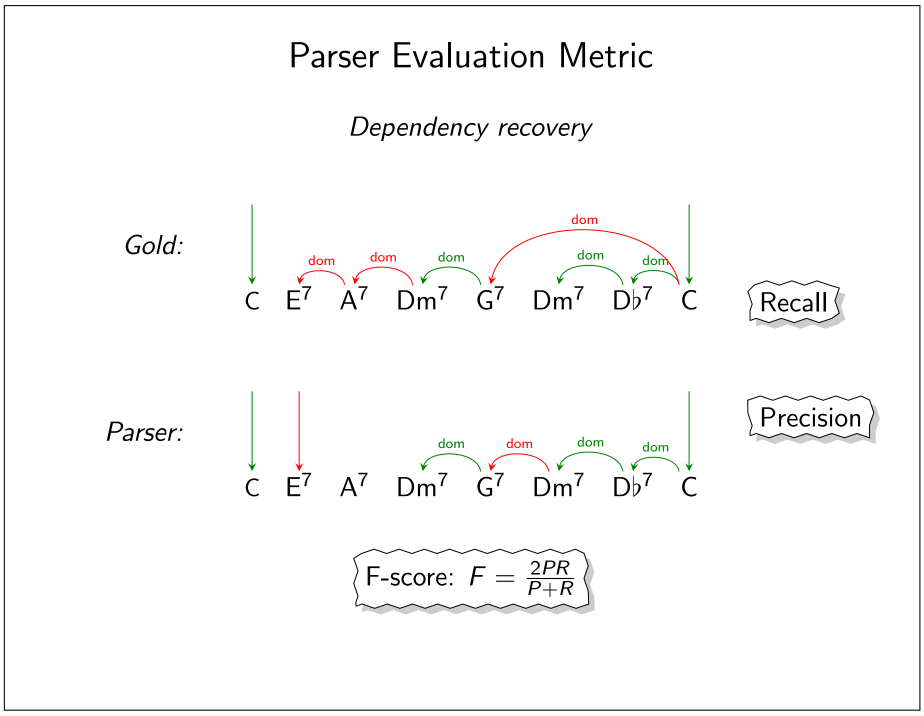
The parsing and supertagging models are trained in a supervised fashion.

We have constructed a small corpus of annotated data for training the models. The corpus consists of chrod sequences of jazz standards, each annotated with a gold-standard harmonic analysis by a single annotator. It contains 74 sequences. Since it is so small, we have not kept a held-out test set. Instead, the evaluation I will present below is performed using 10-fold cross-validation over the full set.



Corpus

- Annotated training corpus
- Jazz chord sequences
- Full harmonic analysis annotated
- 74 sequences
- Cross-validation

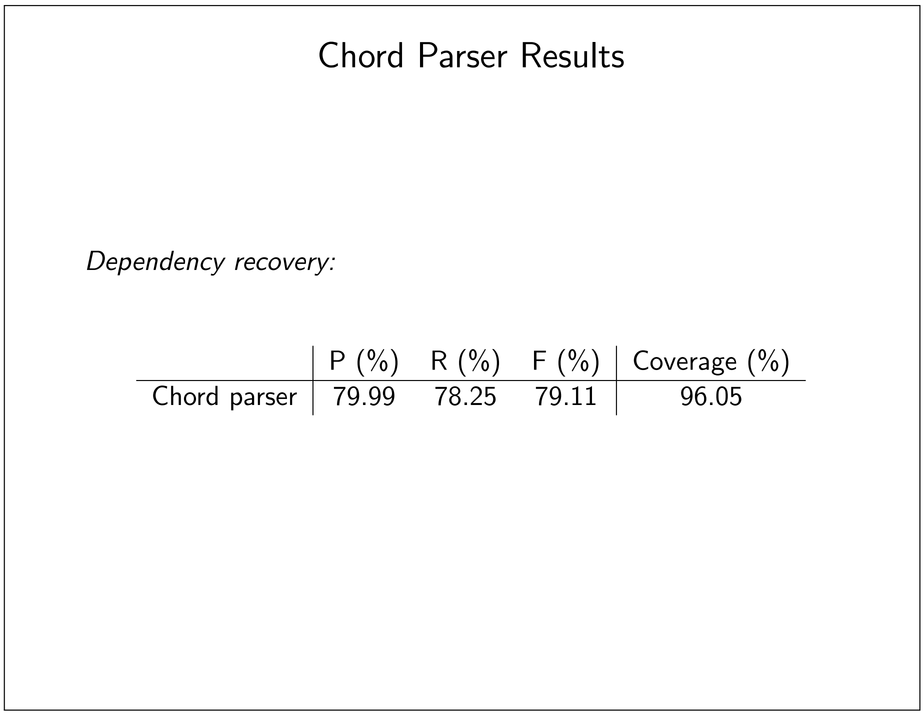


Parser Evaluation Metric

Dependency recovery

Gold:
dom
dom
dom
dom
dom
dom
C E^7 A^7 Dm^7 G^7 Dm^7 D♭^{7} C
Recall
Parser:
dom
dom
dom
dom
C E^7 A^7 Dm^7 G^7 Dm^7 D♭^{7} C
Precision
F-score: $F = \frac{2PR}{P+R}$

We evaluate the parser's performance using a standard metric used to evaluate dependency graphs in NLP: *dependency recovery*. Given a gold-standard dependency graph and that output by the parser, we count up how many of the annotated dependencies were recovered to compute recall and how many of the recovered dependencies were in the annotations to compute precision. The f-score is the harmonic mean of these two figures.



Chord Parser Results

Dependency recovery:

	P (%)	R (%)	F (%)	Coverage (%)
Chord parser	79.99	78.25	79.11	96.05

The results obtained by the chord-input parser are shown on this slide. I will not discuss this table at length here, but we will return to these figures later as a comparison for other systems. Worth noting, however, is that I report a figure for the system's coverage. This is because it is possible that for some input the parser fails completely to find any analysis. We can see from the table that this happens only rarely in this experiment.

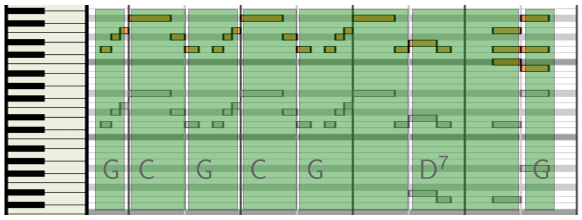
4 Parsing MIDI

Parsing Notes

- We can parse chord sequences

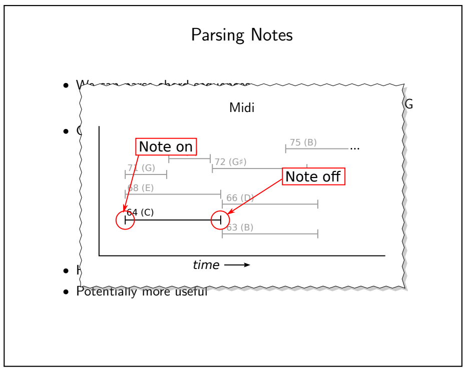
G C G C G D^7 G

- Can we parse music?



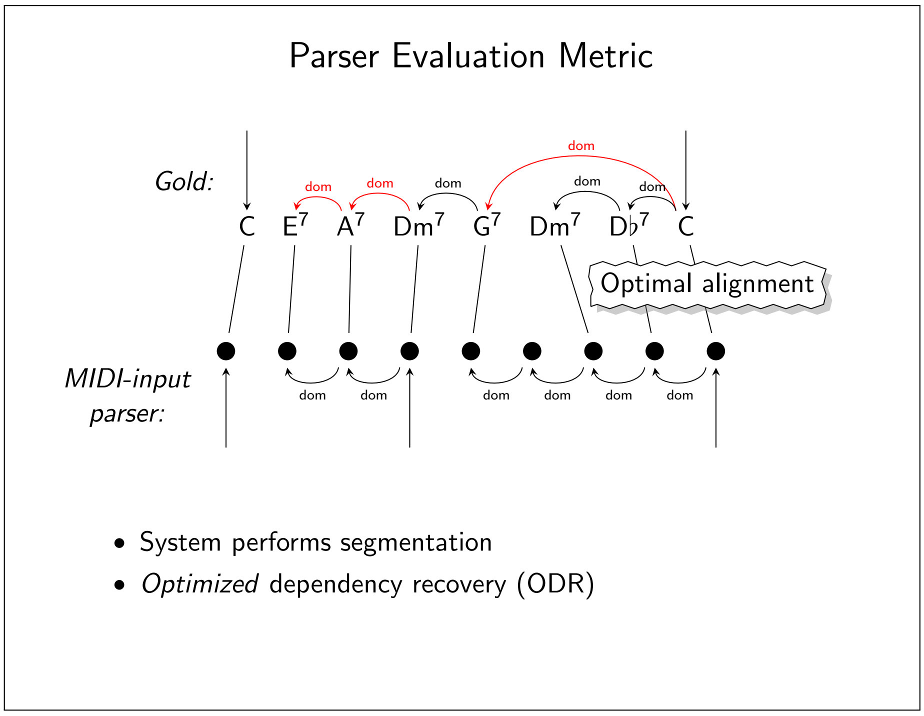
- Harder task
- Potentially more useful

So far, I have demonstrated a method for harmonic analysis of a chord sequence input. Harmonic analysis of a musical performance is harder. In some sense, it is a more interesting task, since it is closer to the task performed by a human listener on hearing a piece of tonal music. It is also likely to be more useful in practical music processing applications.



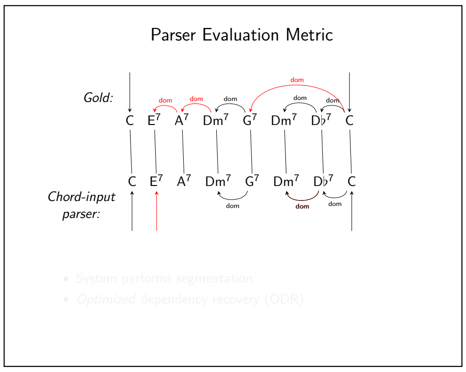
One of the reasons this is a harder task is that it must involve segmenting the input into the primitive units of the analysis – chords. It is worth noting, however, that the system may never assign explicity chord labels to these segments.

We shall represent our input data in the form of MIDI. As far as we are concerned for the time being, this data encodes a stream of notes, each with a pitch (as a keyboard note number), a start time and an end time.



Parser Evaluation Metric

- System performs segmentation
- *Optimized* dependency recovery (ODR)



Parser Evaluation Metric

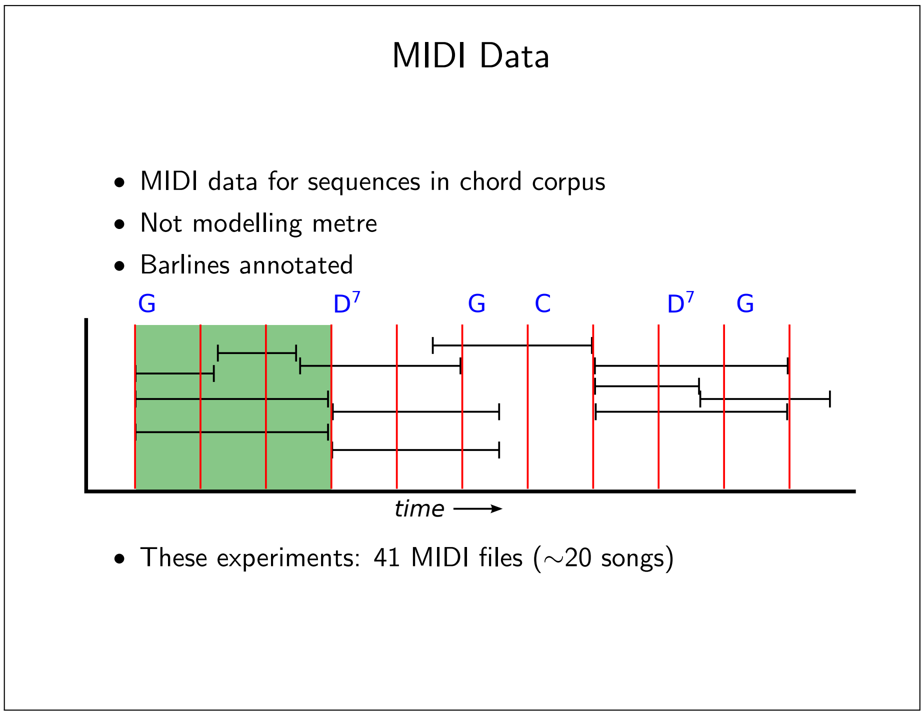
Gold:
dom
dom
dom
dom
dom
dom
C E^7 A^7 Dm^7 G^7 Dm^7 D♭^{7} C
Chord-input parser:
C E^7 A^7 Dm^7 G^7 Dm^7 D♭^{7} C
dom
dom
dom

- System performs segmentation
- *Optimized* dependency recovery (ODR)

Let us return to the issue of an evaluation metric. Unfortunately, we cannot simply apply the dependency recovery evaluation metric that we used before to this task. Recall that previously we simply checked which dependencies were correctly recovered to compute recall, precision and f-score of the parser's output (see right). Now that we are working with systems that perform segmentation of their input, we do not know how the segments over which the parser's analysis is defined correspond to the segments in the gold-standard annotations, and so we don't know which dependencies have been recovered.

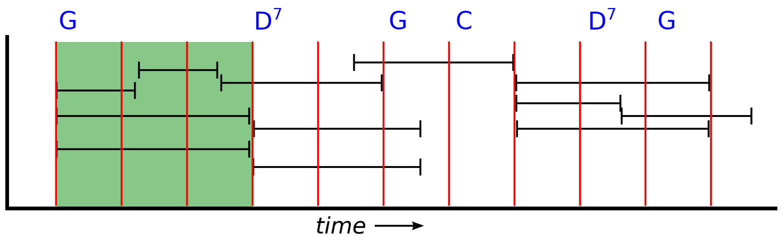
Our solution is to find the optimal alignment between the two dependency graphs – that which gives the most recovered dependencies – and compute the dependency recovery on the basis of that. I shall refer to these figures as *optimized dependency recovery*.

Using this metric also means that we can evaluate a MIDI-input parser on a particular input by comparing its dependency graph to that annotated in the corpus for the same song.



MIDI Data

- MIDI data for sequences in chord corpus
- Not modelling metre
- Barlines annotated



- These experiments: 41 MIDI files (∼20 songs)

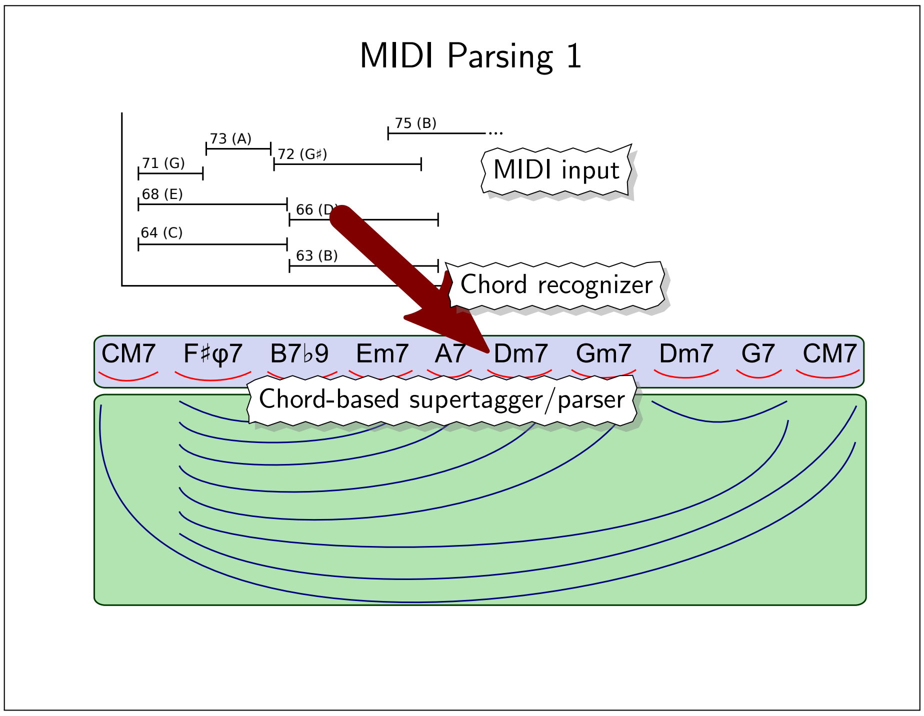
We have collected a set of MIDI-encoded performances available on the web and added them to the corpus described above. The set covers most of the songs in the corpus, with in general several MIDI files per song.

The models I will describe below do not incorporate a model of metre. Consequently, we need to make an extra assumption about the form of our input data. Namely, we assume that we have an alignment of the MIDI data to some high-level metrical unit (say, half-bars). The task of segmentation now becomes somewhat easier: it now boils down to decided at which of these units a new chord begins.

The experiments below are evaluated on 41 MIDI files which we currently have this information for, covering a total of about 20 songs from the chord corpus.

I will now present two simple ways of tackling this harder problem. Both extend the chord-input parser described above. Both systems are really just proofs of concept to give an indication of how the parser might be extended to the harder task.

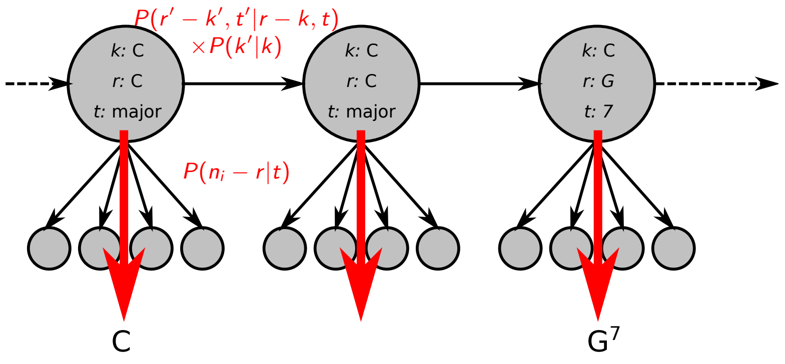
4.1 Pipeline System



The first system is a strict pipeline. One model assigns a single sequence of chord labels to the MIDI data (deciding on the segmentation). This chord sequence can then be passed on to the old chord-input parser exactly as it is.

Chord Recognition Model

- Adaptation of audio chord recognition model: Ni et al. (2011)[2]
- HMM
- Train using EM



[2]Harmony Progression analyzer for Mirex 2011

To perform the initial labelling of the MIDI data, we use an adaptation of the audio-based chord recognition model of Ni et al. (2011). The model is an HMM. Each state encodes three pieces of information: the current key, the root of the current chord and the type of the current chord.

The transition and emission distributions are constrained in certain ways. In the **transition distribution**, the chord root is taken relative to the key $(r - k)$. Then the chord root and type are conditioned only on the previous chord root and type, whilst key transitions are modelled independently $(P(k'|k))$.

The **emission distribution** treats the MIDI notes as a *bag of notes*: each is emitted independently, conditional on the state. The pitches of the notes are taken relative to the current chord's root $(n_i - r)$ and the emission distribution for each note is conditioned only on the chord type.

The model is trained using expectation-maximization, biased by its initialization by giving the notes each chord a high probability and others a lower probability, and transition probabilities initialized to those estimated from the chord corpus.

We can produce a single best chord sequence for a MIDI file using the Viterbi algorithm.

MIDI Parsing 1: Results

Optimized dependency recovery

	P (%)	R (%)	F (%)	Cov (%)
Chord parser (DR)	79.99	78.25	79.11	96.05
Chord parser (ODR)	83.46	81.63	82.53	96.05
MIDI pipeline	63.80	50.33	56.27	78.05

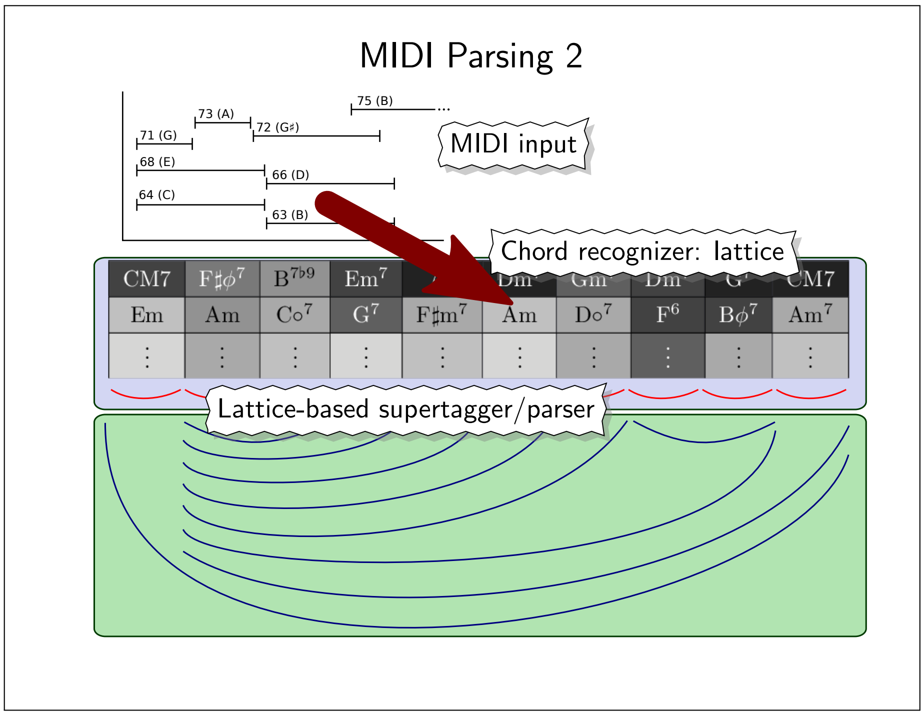
- Previous results with new metric for comparison
- Ceiling for MIDI-input parsers
- Pipeline: low coverage

These are the results for the pipeline system, evaluated using ODR. The first row of the table (grayed out) shows the results we saw before for the chord-input parser. The second shows the same results, this time evaluated using ODR for comparison – this gives slightly higher figures.

It is worth noting that the chord-input parser is not a baseline for this task. In fact, it is performing an easier task, so it makes more sense to think of it as a ceiling.

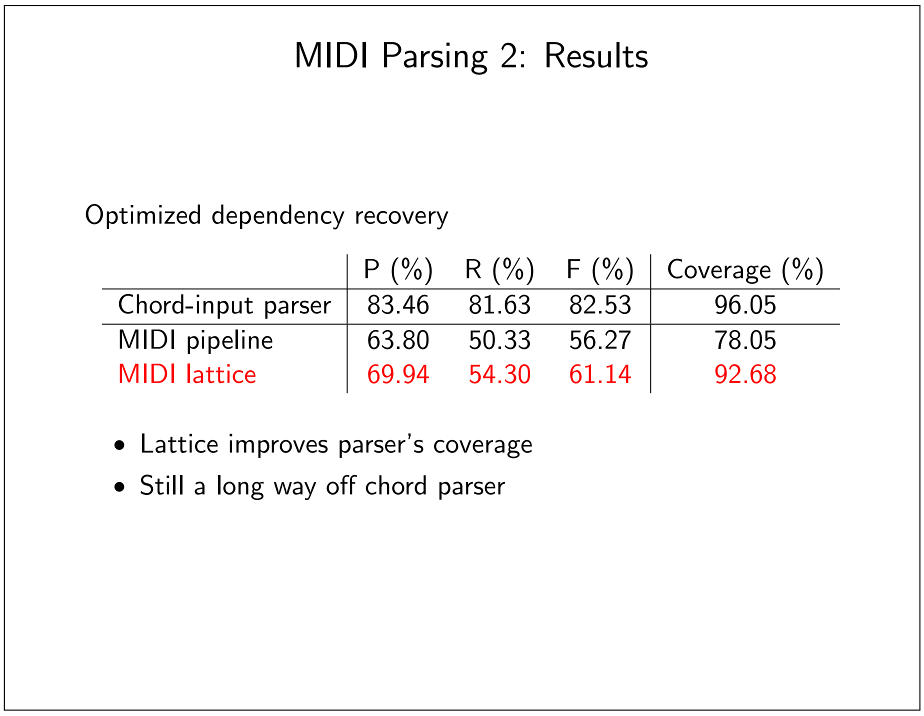
The pipeline system scores rather badly. Note that this is partly due to its low coverage. Recall that for certain chord-sequence inputs, the parser may fail to find any analysis. This has happened in a much higher proportion of cases this time. It is easy to see why. The chord recognizer has a very weak notion of coherence (especially with a small amount of training data). In many cases, it will produce a chord sequence that the parser cannot assign any categories to the combine into a full parse.

4.2 Lattice System



This second system attempts to improve on the first by reducing the problem of over-commitment of the chord recognition model. The same model is used again, but this time, instead of producing a single best chord sequence, it outputs a *weighted lattice*, with multiple possible chord labels for each segment. The supertagger component of the parser is adapted to take input in the form of a lattice and the parser proceeds as before.

This technique effectively allows the parser to take advantage of some of the chord recognizer's uncertainty and make use of labels that the recognizer considered reasonably probable, but that were not on the most probable sequence.



MIDI Parsing 2: Results

Optimized dependency recovery

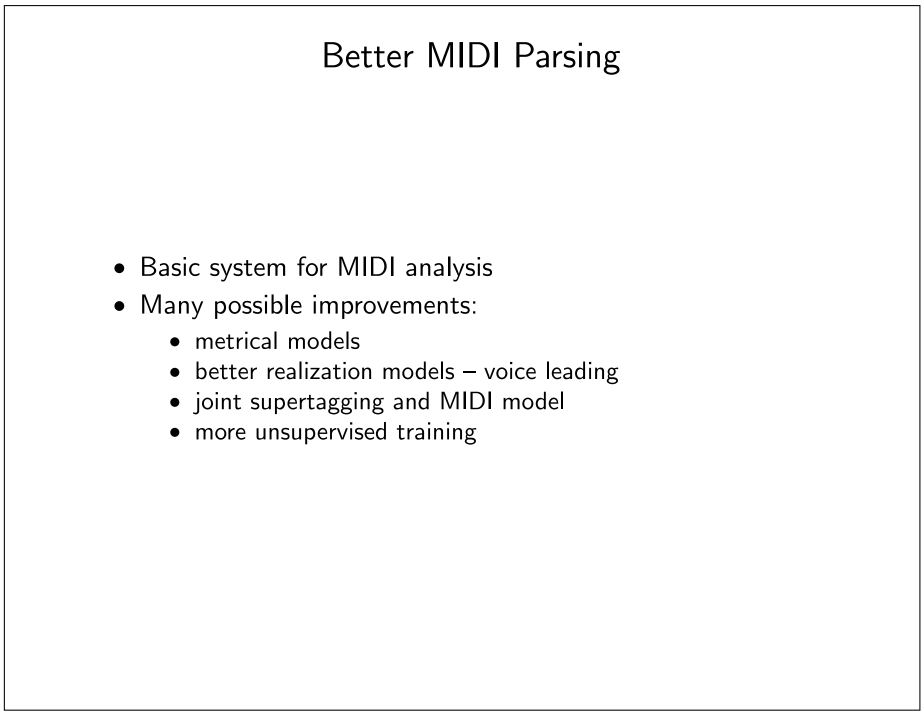
	P (%)	R (%)	F (%)	Coverage (%)
Chord-input parser	83.46	81.63	82.53	96.05
MIDI pipeline	63.80	50.33	56.27	78.05
MIDI lattice	69.94	54.30	61.14	92.68

- Lattice improves parser's coverage
- Still a long way off chord parser

The results of the lattice system are a considerable improvement on the pipeline and, in particular, have pushed up the coverage greatly, as we hoped.

These results are still a long way off the chord-input parser, but that is unsurprising and they show that even with rather a naive approach we can set a reasonable baseline.

5 Conclusion



Better MIDI Parsing

- Basic system for MIDI analysis
- Many possible improvements:
  - metrical models
  - better realization models – voice leading
  - joint supertagging and MIDI model
  - more unsupervised training

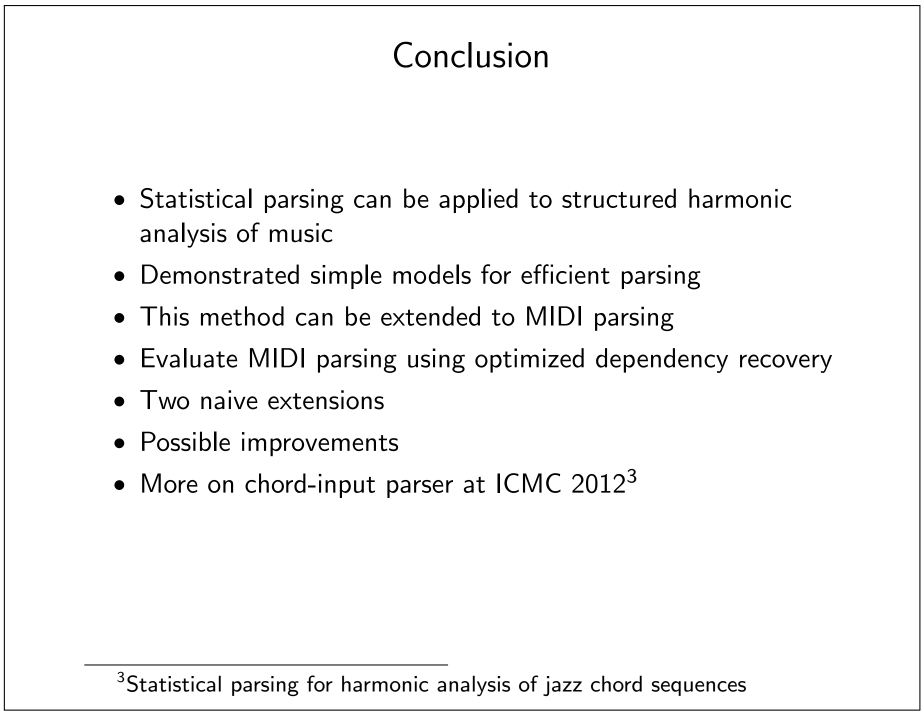
These two systems are a proof of concept for the MIDI parsing task. There are many ways in which they can be directly improved: here are a few.

We expect that incorporating a model of the piece's metre into MIDI models would improve the results and it would furthermore allow us to relax our assumptions about the form of our input data to cover a more general class of MIDI performances.

The system would certainly benefit from better models of realization. Ideally these should encompass notions such as voice-leading.

We have improved over the pipeline system by reducing the bottleneck between the chord recognizer and the supertagger. Better still would be to incorporate these two models into a single joint model, performing segmentation and supertagging in one go.

Finally, the chord recognizer was training using EM: it would be easy to simply feed to more training data. MIDI data is not hard to come by.



Conclusion

- Statistical parsing can be applied to structured harmonic analysis of music
- Demonstrated simple models for efficient parsing
- This method can be extended to MIDI parsing
- Evaluate MIDI parsing using optimized dependency recovery
- Two naive extensions
- Possible improvements
- More on chord-input parser at ICMC 2012[3]

[3]Statistical parsing for harmonic analysis of jazz chord sequences

We have shown that statistical parsing techniques used for natural language parsing can be used for structured harmonic analysis. I have implemented some of these techniques using supervised statistical models to demonstrate that we can use them to perform efficient parsing of music.

The method can be extended to the analysis of MIDI input. I have discussed an evaluation metric suitable for this task. I demonstrated two naive approaches to extending our previous models to this task and suggested some possible improvements.

We will be presenting a paper at ICMC 2012 this September (Granroth-Wilding & Steedman, 2012 (to appear)) covering more details of the first system, the chord-input parser.

References

Granroth-Wilding, M., & Steedman, M. (2012 (to appear)). Statistical parsing for harmonic analysis of jazz chord sequences. In *Proceedings of the International Computer Music Conference*. International Computer Music Association.

Hockenmaier, J., & Steedman, M. (2002). Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Meeting of the ACL*, (pp. 335–342). Philadelphia, PA.

Ni, Y., Mcvicar, M., Santos-rodriguez, R., & Bie, T. D. (2011). Harmony progression analyzer for mirex 2011. In *Proceedings of the 12th International Society for Music Information Retreival (MIREX)*, (pp. 1–4).