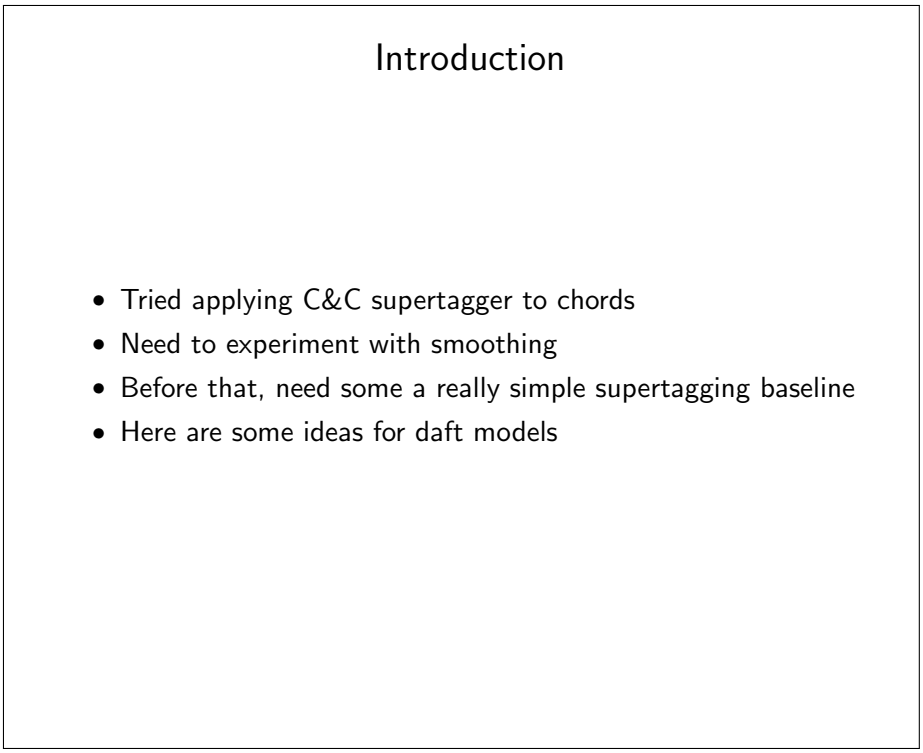
Stupid Baselines for Musical Supertagging

Mark Granroth-Wilding

14th Sept 2010

This is just a short document (in the form of an annotated presentation) with some thoughts on simple baseline approaches to musical supertagging.

1 Introduction



Previously I tried a naive approach to supertagging for chord parsing by using the supertagger component of the C&C parser straight out of the box. I expect that the results I got can be greatly improved on by trying different methods of smoothing unseen data. However, before I go on with trying to get some better results of this or experimenting with other supertagging approaches, I need to establish some baseline results using even more naive supertagging methods.

In this document I present some possibilities for such baselines.

The Input

- Our input is chord sequences: e.g.

  CM7 A7 Dm7 G7 CM7

- Don't want to model this directly

  GM7 E7 Am7 D7 GM7

  $\equiv$

  CM7 A7 Dm7 G7 CM7

- Should receive the same interpretation model

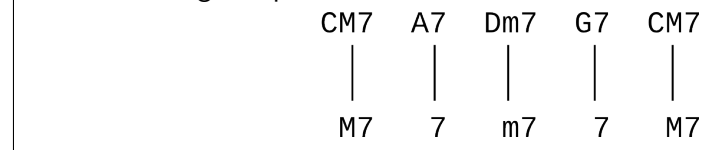
Before I go any further, I should mention the type of input we're handling.

The inputs are chord sequences in which each chord consists of a root (a note) and a chord type. However, it would be a mistake to model interpretations over chord sequences in the form shown above. Our models should be insensitive to absolute pitch. The two sequences printed above should receive exactly the same interpretation. The only difference between them is a transposition of a perfect fifth (seven semitones). The relative pitches and the chord types are all the same and the absolute pitch doesn't affect the way we interpret it the music.

2 Modelling

Stupid Idea

- Important information in chord types
- Just ignore pitches



- But interpretation depends on pitches of surrounding chords

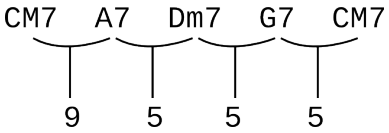
So, the question now is how to formulate the features of the chord sequence that will be used as input to the model – that is, how the input should be preprocessed before it reaches the supertagging model.

The chord types (M7, m7, etc) tell us some useful information for interpretation on their own. One way to construct a representation that abstracts away the absolute pitch would be to ignore the pitch part of the chords altogether and model only the chord types.

Interpretation depends on surrounding chords, though. This is clearly a stupid idea, since we need to know something about the relative pitches of the surrounding chords to know how to interpret a chord. The type of the chord itself isn't enough. For example, a dominant chord will usually resolve down a 5th.

Slightly Less Stupid Idea

- Only look at intervals between roots



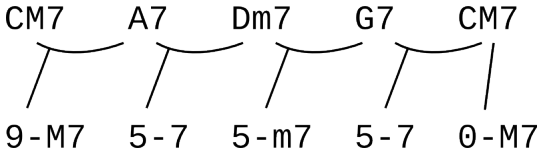
- Number of semitones between chords
- Intervals important to interpretation
- 5 – down a perfect 5th, likely to be dominant-tonic
- 7 – down a perfect 4th, likely to be subdominant-tonic

Another idea is to model just the intervals between the chords. The diagram above shows these as the number of semitones between the roots. This information tells us quite a lot about probably interpretations. A dominant chord will usually resolve a 5th down (a 4th up, an interval of 5 semitones up). A subdominant chord will usually resolve a 4th down (5th up, 7 semitones up).

The interpretation is still uncertain, of course. A dominant chord might not resolve straight away, or it might resolve to a substituted chord. The intervals do give us some important clues about the most common tension-resolution patterns, though.

Both Together

- Could use both for observations
- Observation consists of type of chord and interval to next chord



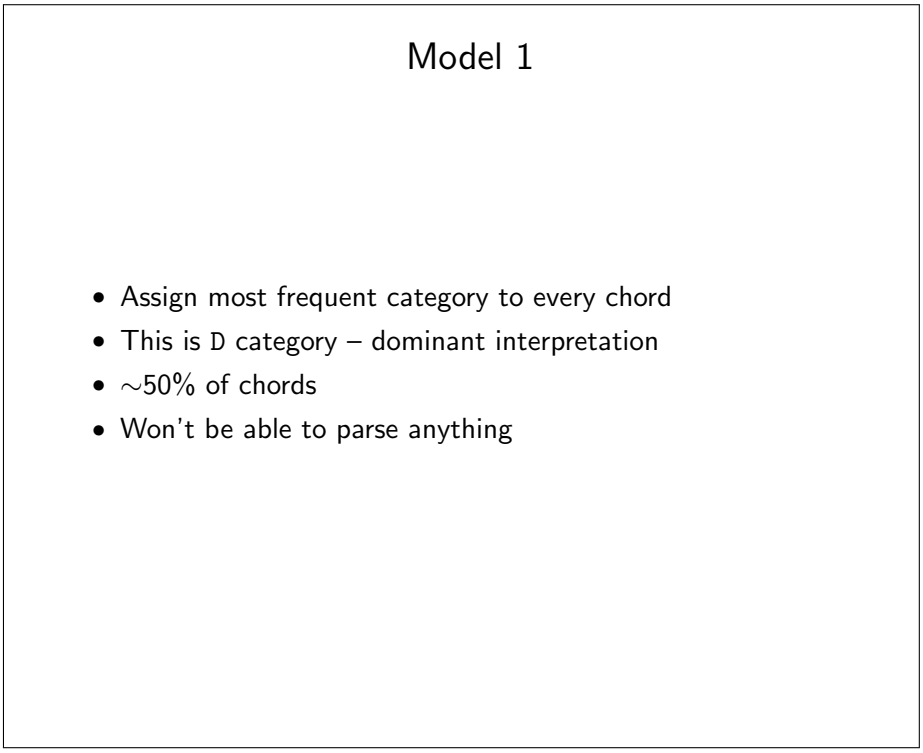
- Used this for C&C input

These two pieces of information – the chord type and the intervals between chords – both give important clues about interpretation. An observation made up of the two features is insensitive to absolute pitch, but otherwise includes all the information that was in the original chords (since we assume the input to be equally tempered and therefore forbid ourselves from using any notational clues such as choices between enharmonically equivalent notes).

This is precisely the format of observation that I've used so far to train the C&C supertagger on.

3 Daft Model Ideas

I now present several proposals for models that are as daft as possible to be used as baselines.



Perhaps the simplest daft model conceivable is one that assigns the same category (and therefore interpretation) to every chord. The obvious choice is the category that appears most frequently across the whole corpus. And indeed it is a clear winner: the category for an unsubstituted dominant chord 'D' handles about half of the chords in the corpus.

Unfortunately, this scheme isn't going to lead to any parses. At best the dominant categories, if we're very lucky with pitches, could all combined into a single complex category spanning a sequence, but without any tonic resolutions we'll never get a fully-resolved atomic category for the span.

This isn't even worth implementing.

Model 2

- Assign tonic category to every chord
- Second most common category
- Should parse everything – every chord in a new key!
- Will get lots of correct roots
- Will get mostly incorrect functions

Instead, we could do the same thing with the second most frequent category – the unsubstituted tonic chord '`T`'. In the gold standard, this covers about a quarter of all chords.

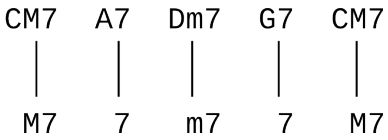
Unlike the last scheme, we can hope to get some parses out of this. In fact, every sequence should be parsed, with every chord interpreted as a tonic in its own key, each modulating to a new key.

We would actually expect this to do quite well in assigning roots (tonal space points) to the chords, since the majority of chords are not substitutions, so interpreting their played root as the actual root will get the correct answer. The majority of functions (tonic, dominant or subdominant), however, will be wrong, since the majority of chords are not tonics.

This perhaps is a useful baseline, since it's sufficiently stupid and we would hope, but cannot assume, that more complex models will do a better job of assigning roots.

Model 3

- Pick highest unigram probability, modelling only chord type
- E.g. all M7 chords get tonic interpretation
- All 7 chord get dominant interpretation
- Will get more correct functions than model 2
- Will get fewer parses



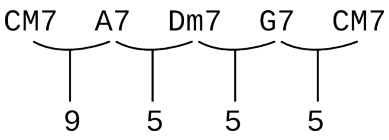
We would hope to get slightly better models once we start using some probabilities computed from the corpus. The simplest thing we can do is to model the frequentist unigram probability of each category given the observation.

This possible baseline model involves modelling the unigram probabilities using only the chord type as an observation. This would probably result, for example, in all M7 chords getting a tonic interpretation and all 7 chords getting a dominant interpretation.

This should do a bit better than the previous model on assigning functions, since the function is fairly heavily dependent on the chord type. However, we're back to the problem we had with model 1 with the parses – the chosen categories won't take account of surrounding intervals, so there's no reason to believe they'll actually be able to combine with the surrounding chords.

Model 4

- Pick highest unigram probability, modelling only intervals
- E.g. 5, perfect fifth down, will get dominant interpretation
- Good for some common cases
- Will get more parses than model 3, but incorrect functions
- Will make a mess of tonics

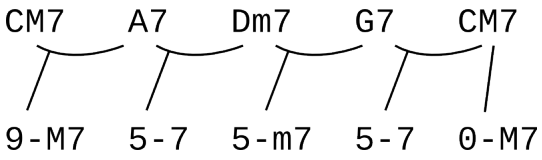


Another unigram model could use just the intervals between chords. This is likely to get more parses, since it's more likely to pick something sensible for forward-facing slash categories, in particular for dominants, substituted and not. It should do well on common cases like these dominants, but since it's only a unigram model and is only using such limited information about the input it will fail to pick up rarer categories altogether. It's also unlikely to do very well on detecting the correct functions, since the chord type information plays a major role in this.

This model will not do well with tonics, since in these cases the interval that follows the chord makes no difference to the interpretation. In the example above the CM7 chord is only represented as '9', but this interval is irrelevant to the interpretation of the chord.

Model 5

- Pick highest unigram probability, modelling intervals and chord types
- Uses all available information
- Probably better than model 3 (types only)
- Getting closer to C&C model
- Might already start suffering from data sparsity



The obvious next model is one that combines the interval and the chord type, using all the information available about the chord. This is probably going to perform better than model 3, since it's more likely to choose categories that will be able to combine.

It's getting close to the C&C model tried previously and the comparison will tell us how valuable the contextual information is when weighed up against the data sparsity problem. It may be that this model already starts to suffer from data sparsity, which would be interesting and rather scary.