Melodic Analysis using Hierarchical Hidden Markov Models

Ruari Kerr

Master of Science

School of Informatics

University of Edinburgh

2011

Abstract

This dissertation attempts to show that hierarchical hidden Markov models (HHMMs) can be used effectively to model mid-level musical structures. Using data taken from chorales composed by Johann Sebastian Bach, a number of models were designed and trained to reflect elements of phrase-level and bar-level structure. These models were evaluated based on their performance in two different tasks; the prediction of phrase boundaries in unannotated data, and the generation of new compositions.

Acknowledgements

Thanks to my supervisor, Alan Smaill, for his advice throughout the course of this project.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Ruari Kerr*)

Table of Contents

Chapter 1

Introduction

This dissertation investigates the use of hierarchical hidden Markov models to model mid-level musical structures. Conventional hidden Markov models have been used to model music for some time, but their focus on local dependencies means they fail to capture elements of the higher-level structure found in music. One approach to remedying this is using hierarchical hidden Markov models, which explicitly model this higher-level structure. This dissertation aims to produce hierarchical models capable of capturing the structure found in chorales composed by Johann Sebastian Bach, with the ultimate aim of producing new compositions which display this structure.

Chapter 2 explores the background to this work, describing previous work with hidden Markov models in music, attempts to model musical structure, and the background of hierarchical hidden Markov models and their use in this area. Chapter 3 explains various decisions that were made with regards to the design of the project, including the selection of data to be used, the design of the various models used, and the selection of an appropriate evaluation metric. Chapter 4 describes issues relating to the implementation of the project, in particular the use of a 'flattening' algorithm to deal with hierarchical models in a more efficient manner, and the methods used to generate compositions.

Chapter 5 gives the results of evaluating the different trained models on the task of parsing unannotated data. Chapter 6 then seeks to analyse this data and put it into context with other systems, as well as analysing some compositions produced by the models. Finally, Chapter 7 draws an overall conclusion about the success of the project, and suggests some areas for improvement in the models, and some areas for further work.

Chapter 2

Background

This chapter details previous work that forms a background to this dissertation. This combines work in three different areas; the application of hidden Markov models to music, the modelling of musical structure, and the use of hierarchical hidden Markov models.

2.1 Hidden Markov models in music

Music has many features in common with natural languages, and frequently the same techniques are used to model both. Hidden Markov models (HMMs) are widely used in modelling language; they have a relatively low computational cost, but can be remarkably efficient at capturing local properties. A number of approaches have been taken to using HMMs for music.

Allan and Williams [2005] uses HMMs to approach the problem of four-part harmonisation in Bach chorales. In this case, the melody line is taken as the observed state sequence, while the hidden states represent the underlying harmony. This model was trained on chorales and used to generate new harmonisations for melody lines; a second HMM was used to add ornamentation. Allan and Williams note that this system produces 'reasonable' harmonisations, exhibiting both parallel and contrary motion and concluding on appropriate cadences. However, the model does produce an undesirable number of large jumps in pitch. Allan and Williams do not compare the performance of their system directly to any competing approaches, although they do note that rule-based systems have much higher storage and processing time requirements.

A similar system has also been used outside of the classical context to generate

accompaniments to a melody evoking a particular mood [Chan and Ventura, 2008]. The melody is sampled at regular intervals to provide the emissions, which are used to infer the chord sequence; the full accompaniment is then generated from the chord sequence according to the desired mood. Chan and Ventura acknowledge the difficulty in quantifying the success of a compositional system; they suggest evaluating the system in three areas: skill (musical knowledge and intelligent decision-making), appreciation (consistency and adaptability) and imagination (originality), based on human feedback.

Eigenfeldt and Pasquier [2010] uses a HMM to generate chord progressions in a style learned from source data. The selection of states is constrained according to a user-specified 'phrase vector' which specifies the general direction of the musical progression and values for the complexity of chords used and the intervals between successive chords. These factors can be assigned different weights; the selection of states is then made based on a balance between satisfying these constraints and remaining faithful to the training corpus.

Work with HMMs has often focused on chord progressions, as effective chord progressions can be created with a minimal use of context, and there are particular two-chord transitions corresponding to cadences, which are very prevalent, particularly in common-practice classical music. When the models are instead applied to generating melodies, the limitations of the form can be more apparent. While the transitions between notes will still correspond to those in the training data, the melody viewed as a whole may appear somewhat directionless; notes that are more than a few time steps apart are only very weakly correlated. Musical compositions generally have aspects of higher-level structure which cannot be captured by the local focus of a hidden Markov model.

2.2 Modelling musical structure

Musical structure can occur at a number of different levels. At the highest level, there may be an overall structure to a piece, as in the sonata form, or the verse and chorus structure of a pop song. However, a more common focus is mid-level structures such as metrical and phrase structure. Lerdahl and Jackendoff [1996] takes a psychology- and linguistics-based approach, creating formal rules for musical structure based on Gestalt principles. In particular, musical phrases, and higher-level groupings, are identified based primarily on proximity (in time) and similarity, which can be based on a number of factors, such as pitch, note length or dynamics.

Temperley [2004] expands on this approach, considering a wider variety of applications and musical styles, and implementing a system which uses dynamic programming to assign phrase boundaries to a piece. Temperley tests this system on a corpus drawn from the Essen folksong collection and compares the predicted phrase boundaries to the correct groupings; the system achieves a precision of 74.6% and recall 75.5%.

A different approach to the problem of identifying phrase boundaries is the use of probabilistic grammars [Bod, 2001]. Bod trained a probabilistic grammar on a section of the Essen collection annotated with phrase boundaries and used this in a parser to assign phrase boundaries to test data, achieving a precision of 76.6% and recall 85.9% for the best system. Bod asserts that the Gestalt principles do not translate straightforwardly to perception of music, and points to a prevalence of 'jump-phrases' (phrases containing, rather than separated by, large jumps in pitch) in the corpus, which would be misidentified by Gestalt-based theories. He suggests instead that musical phrases tend to have a rising and falling 'arch' pitch contour.

In Eigenfeldt and Pasquier [2010], the system avoids modelling any structure by following a user-defined bassline; any structure in the generated pieces ultimately comes from the user rather than the model itself. Another approach to integrating structure with a HMM, in which the structure is part of the model, makes use of hierarchical hidden Markov models.

2.3 Hierarchical hidden Markov models

Hierarchical hidden Markov models (HHMMs), first proposed in Fine et al. [1998], are an extension of the concept of hidden Markov models motivated by the presence of multi-level structure in natural language and other domains. A HHMM can be thought of as a HMM which can contain, in addition to the normal production states, 'internal states', which instead of a single symbol emit strings of symbols produced from a sub-HMM. These sub-HMMs may in turn also be hierarchical, allowing for multiple layers of structure.

These models were applied in this paper to the analysis of English text and handwriting. In both cases they were able to learn structures at various different levels in the training data. However, the original algorithms used were very complicated and computationally inefficient. Wierstra [2004] provides an alternative method which converts a HHMM to an equivalent non-hierarchical HMM. This model produces ex-

actly the same outputs as the original model with the same probabilities, but is much simpler to perform computations on, and the hierarchical form can be restored to maintain the higher structure. Fine et al. [1998] noted this equivalence in passing, but they did not consider using it to improve their algorithms.

Another approach to simplifying the HHMM algorithms is by equivalence to dynamic Bayes networks [Murphy and Paskin, 2001]. This technique performs inference in $O(T)$ time, as opposed to $O(T^3)$ for the original algorithms, where T is the length of the emission sequence; the authors also assert that it maintains hierarchical structure better than the flattening technique. However, the algorithm is quite difficult to implement.

Weiland et al. [2005] uses HHMMs to learn from the melody lines of Bach chorales, using knowledge of the general structure of the chorales to incorporate musical phrase boundaries into the structure. The results were promising in terms of capturing phase structure, but there remains room for expansion; for example, they did not include note durations or dynamics in their data. Incorporating these aspects could lead to a more powerful system.

Chapter 3

Experiment design

This chapter covers a number of areas relating to the design of the experiments, including the choice and manipulation of data, the selection of an appropriate evaluation metric, and the design of the various models used.

3.1 Data

Similarly to Weiland et al. [2005], I took data from Bach's chorales. These were an attractive choice for a number of reasons; there are over 350 of them (a fairly large corpus for music, although dwarfed by the size of natural language corpora), they are annotated with phrase boundaries in the form of fermatas[1], and they are readily available in an easily-processed format online. The data was processed to isolate the soprano lines and remove extraneous information, leaving only note durations, pitches and phrase ending marks.

It is standard practice to transpose compositions into the same key [Weiland et al., 2005, Allan and Williams, 2005], as combining data from different keys will result in weakening the overall tonality, and training separate models for each key suffers from sparse data. This approach was taken, but rather than expressing the pitches in the transposed key, they are represented relative to the degrees of the relative major scale, as the number of the scale degree, possibly with the symbols '#' or '-' to represent sharps and flats, respectively. For example, if a piece is in G major, the pitch C# would be represented as '4#', while F is represented as '7-'. The symbols '^' and 'v' represent notes an octave higher or lower, respectively.

[1] A fermata, or pause, indicates that a note is to be extended beyond its notated duration; it was used in chorales by Bach and other Baroque composers at phrase endings, where a breath should be taken in performance.

While Weiland et al. [2005] had phrases enclosed in parentheses, with '(' and ')' as separate symbols to indicate the start and end of each phrase, I chose to have phrase endings indicated by ';' attached to the last emission of the phrase. The phrase-ending emissions are produced by different states to the corresponding non-ending emissions. This has the disadvantage of requiring up to twice as many states compared to the other approach, but it does have a number of advantages.

The primary motivation in taking this approach was parsing unannotated data to add phrase endings. If the phrase beginnings and endings are produced on their own, then this model cannot parse data without phrase endings directly, as there are no emissions from the corresponding states. One would have to generate all possible phrase segmentations of the data and find the segmentation which parsed with the highest probability. By contrast, with the alternative approach the emissions of the phrase-end states can be changed to remove the phrase ending, and then the unannotated data can be parsed directly and the state sequence with highest probability can be found. The predicted phrase endings then correspond to the phrase-end states in the best state sequence.

Another factor to consider is context. Consider the snippet in Figure 3.1. In Weiland et al. [2005] this would be rendered as 'd, c,), (, a, e'; under my approach it would be rendered as '2, 1;, 6, 3'. Since the probability of a transition is based only on the previous state, the first model loses some potential context; it can only learn the probability of a note being the end of a phrase, while the second can learn transitions that commonly lead to a phrase ending. This is useful since phrases often end with a cadence, i.e. a particularly strong transition. Also note that the second model can learn the transition probabilities between the last note of one phrase and the first note of the next, which is frequently a larger jump than transitions within phrases.

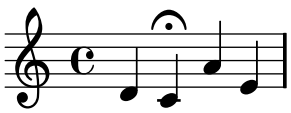

Figure 3.1

The chorales were separated into major and minor pieces, as major and minor keys have very different notes and progressions. This is also standard practice [Weiland et al., 2005, Allan and Williams, 2005].

3.2 Evaluation metric

Since evaluation of composition is somewhat subjective, and time-consuming, another method of evaluating the success of a model was required in order to compare models during the training process. One method used in Bod [2001] and Temperley [2004] was the prediction of phrase boundaries in unannotated data. Since the purpose of the model is to capture phrase-level structure, it should be able to infer phrase boundaries fairly accurately.

As such, 50 major chorales were set apart from the training data to be used for testing. The phrase endings were stripped out and saved separately. To evaluate a model, the training chorales were parsed and the model's predicted phrase boundaries were compared against the actual boundaries. Since the final phrase boundary is always the end of the chorale, and is always predicted correctly, it was disregarded. The model is then scored by its precision and recall over the whole test set.

3.3 Initial model

Examining the structure of the major chorales to look for useful patterns, the first observation was that the chorales almost always start with a note from the tonic triad (ie degrees I, III, V) and end on the tonic, while other phrases are more varied in their starting and ending notes. This suggests that the start and end phrases should be produced by different internal states to the main body of the chorale.

Considering the total number of phrases in each chorale, 6 was both the mode and the mean value; an even number of phrases was generally more common than an odd number. With this in mind, I decided to use a three-state model at the hierarchical level, with states for start, body and end phrases, as was done in Weiland et al. [2005]. The start state transitions to the body with probability 1 and the body transitions to the end state with probability 0.2, and to itself with probability 0.8. Note that when the model is converted to a minimally self-referential form, the self-transition probability will be absorbed into the production states. The value of 0.2 was chosen based on the mean of the negative binomial distribution.

The next step was designing the production states for each internal state. One question was what each production state should represent; would it be possible to have states that could emit one of several notes, e.g. those in a particular chord? But when looking at the data, it was clear that voice leading led to the majority of transitions

being between adjacent or at least close notes. This cannot be maintained if each state has multiple possible emissions, since the probability of an emission from a state is independent of the emission from the previous state. For example, if we have a state which can produce the notes C, E, G and a second state which can produce D, F, A, the transition probability between these states will be high, because C → D, E → D, &c. are transitions between adjacent notes, but if this model is used for generation it will also produce transitions like C → A and G → D, which should be less likely. As such, it was necessary to have a state for each note, at least for models intended to be used for composition.

From examining the data, the notes involved covered just over two octaves, from '4v' to '6^', plus a state for a rest 'r'. I decided to allow all notes to be produced in each internal state, even when a note did not appear in the data, to allow for unseen data. Also, all notes (but not the rest) were allowed to be phrase-endings, again to allow for unseen data, but also because it would make converting to the parsing model simpler. This gave a total of 57 production states for each internal state.

The transition matrices for each sub-HMM were designed by hand based on the transitions in a small subset of the data, with a small amount added to non-appearing transitions to allow for unseen data. This completed the initial model, which could then be trained. See Figure 3.2 for a simplified diagram of the model.

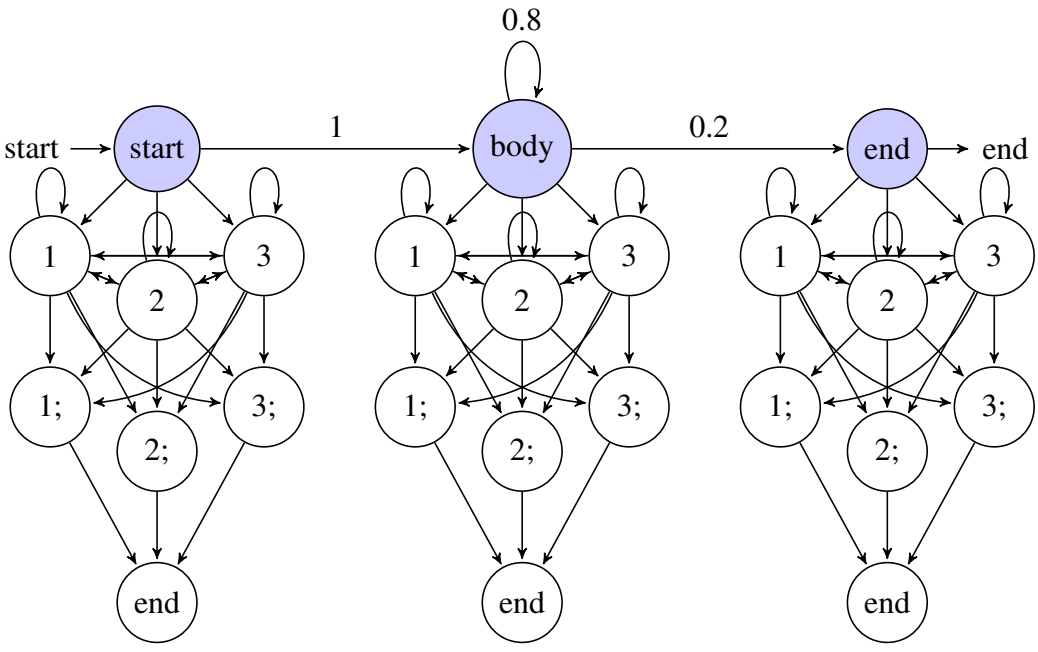

Figure 3.2: A simplified version of the initial model, using only the notes 1, 2 and 3; the full model has 27 notes plus a rest

3.4 Alternative models

There were a number of alternative model structures that were tested to compare against the initial model. The first was a model with four hierarchical states rather than three. Having noted that chorales with an even number of states were more common than those with an odd number, I replaced the single body state with two states which alternate, and gave the second state a higher transition probability to the end phrase state (see Figure 3.3).

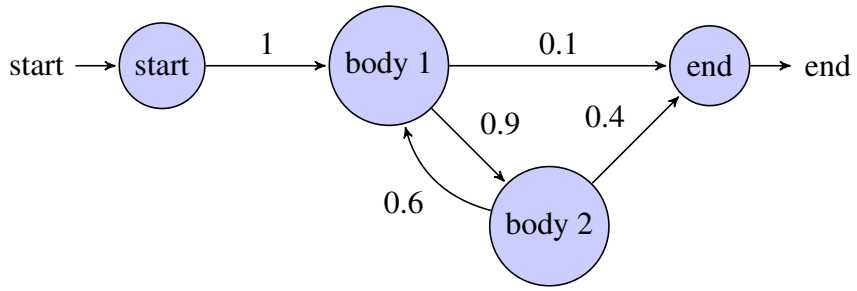

Figure 3.3: The 'four state' model, hierarchical level only

The second model adds a second level of hierarchical structure to the model, mirroring the start → body → end structure within the phrase. Since the end states were already separate states, this only required the addition of 'start states', so the number of states does not become too excessive. The reasoning behind this was that the other models were tending to produce too many short phrases; the intention was that by adding this extra level of structure, the model would produce more long phrases. There is also a certain amount of merit to the idea of structure within a phrase.

The third model has the same states as the original model, but alters the transition matrix to allow a greater variety of transitions at the hierarchical level, allowing the start and end states to occur more than once or out of sequence, although the first and last phrases are always start and end phrases, respectively. The justification for this came from looking at the data and noting that, frequently, phrases in the main body of the chorale would resemble start or end phrases. This model allows these phrases to be parsed using the start or end sub-models. I was curious to see how performance would be affected by ambiguity at the hierarchical level; note that in all other models, an annotated chorale has only one possible parse at the hierarchical level. If the training assigns phrases to the start and end states sensibly, it could result in all three sub-models being improved, as the start and end sub-models get more data to train from, and the body sub-model gets a stronger identity (of 'phrases that do not resemble start

or end states'). On the other hand, if the assignation is bad, the start and end sub-models will be weakened by the inclusion of phrases in the wrong form.

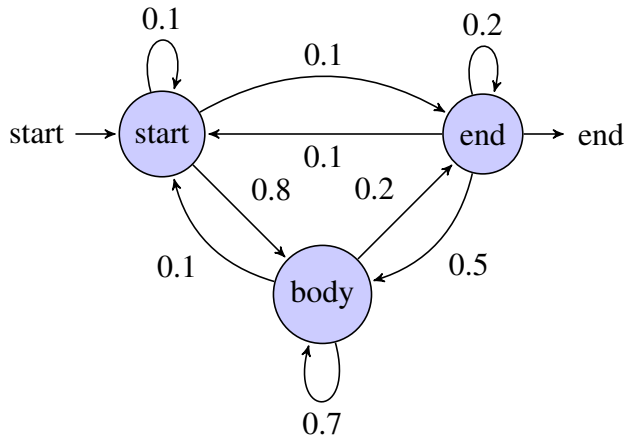

Figure 3.4: The 'flexible' model, hierarchical level only

3.5 Alternative viewpoints

A musical piece can be thought of as a kind of surface which can be viewed from a number of different viewpoints; each musical event has a number of different characteristics which can be observed, such as pitch, duration, dynamics, &c. [Conklin and Witten, 1995]. My initial model was focused on pitch, but I was keen to experiment with other viewpoints and compare the results. In particular, it seemed that note duration might offer a good predictor for phrase boundaries, as phrase endings are often found on longer notes. I also considered making use of pitch contour, the difference between the pitches of successive notes.

The first note duration-based model was based on the original pitch-based model, with three hierarchical states corresponding to the phrase level. However, for the next model I tried a model based on bar-level structure instead. This was primarily motivated by compositional concerns; a model that generated note durations based on the phrase level would be unlikely to produce output that confirmed to a regular bar structure, decreasing the musicality of the output significantly. The aim of this model was then to produce output that would conform to a 4/4 bar structure.

In the model, each bar is divided into segments of 1, 2, 3 or 4 beats in length (Figure 3.5). There are separate sub-models, such as the one in Figure 3.6, for each length, which resemble a tree structure; a number of fixed paths can be taken to produce

a sequence of note durations adding up to the total duration. Phrase endings are also included, although initially I did not expect this model to perform particularly well on the parsing task. Since many of the chorales in the corpus start with an anacrusis (a single beat before the first barline), the model can start either on beat 1 or 4; it can also finish after beat 3 or beat 4, since music theory requires that when an anacrusis is present the last bar be shortened accordingly. This leads to a minor problem when generating output with this model; the model may generate output with an anacrusis but without the shortened final bar, or vice versa. This sort of long-range dependency cannot be captured in a hidden Markov model.

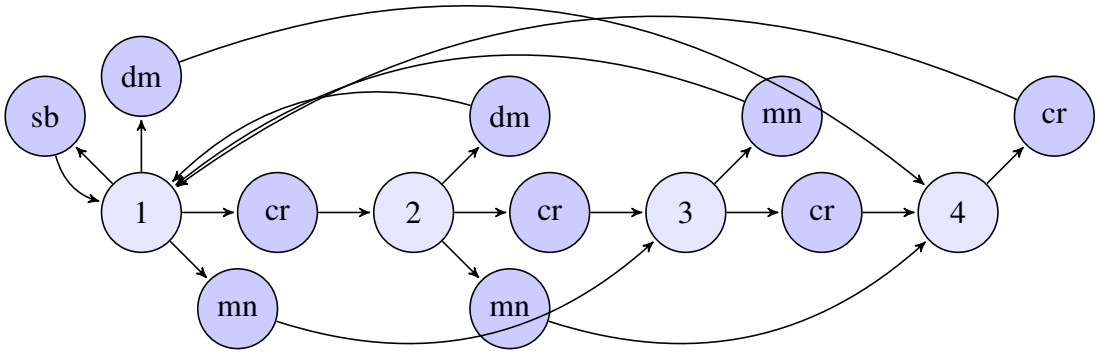

Figure 3.5: The bar-level duration-based model, hierarchical level. Note that the beats 1, 2, 3, 4 are not real hierarchical states but are included for clarity

For training, the data was further restricted to major chorales which were in 4/4 time throughout; fortunately, this comprised the majority of the corpus, so the amount of training and test data was not significantly reduced. If it had been, note durations from minor chorales could have been included in the training data, since the durations should be much the same even if the notes are not. Note that this model is far more restricted than the phrase-based model; each state has far fewer transition options than in the other models. This means that there are fewer transition probabilities to be learned, so sparse data should be less of a problem than with other models.

Since there are considerably fewer different durations than pitches in the data, and so the duration sub-models have fewer states, it is possible to have models with larger hierarchical levels without the number of states becoming unworkable. Following the logic that led to the 'two-level' pitch model gave a model designed to produce longer phrases (Figure 3.7). Both hierarchical levels are structured similarly; a series of states that progress in a linear fashion, with an increasing chance to branch off to the end

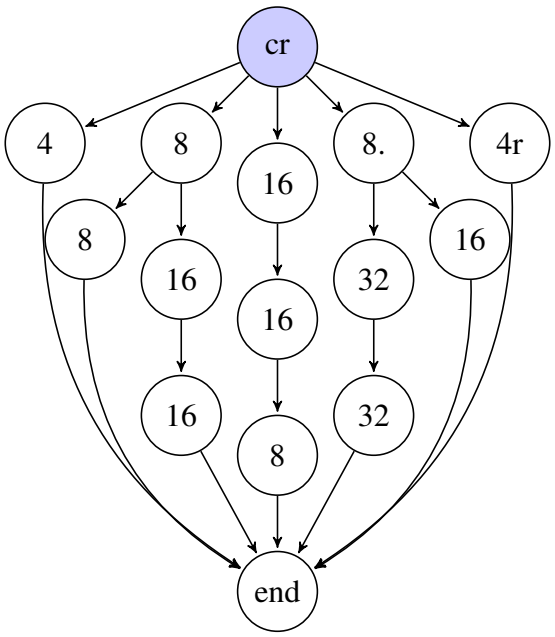

Figure 3.6: The crotchet-length sub-model. Other sub-models are much simpler

state, with a loop after several states. The primary difference between the two levels is that the top level loops between two states, as an even number of phrases is more common, while the within-phrase level loops on a single state, as there is not such a difference for the number of notes in a phrase. The sub-model level produces only one note at a time. A variation of the model makes the within-phrase level the production state level instead, allowing each state to produce any of the durations.

While this model should produce phrases with more consistent lengths, it may suffer from sparse data, since each sub-model (apart from the loops) is only run at most once per chorale, and many of the later ones will only appear in a small number of them.

3.6 Combining viewpoints

Frequently, a model can be improved by combining more than one viewpoint. In particular, a model which combined the pitch and duration viewpoints could perform better than either individual model. However, such a model would have to have a very large number of states; each state represents a pair of duration and pitch, so the number of states is proportionate to the number of possible durations multiplied by the number of possible pitches. I attempted to create a combined model by combining the transition matrices from the trained phrase-based pitch and duration models; the transition be-

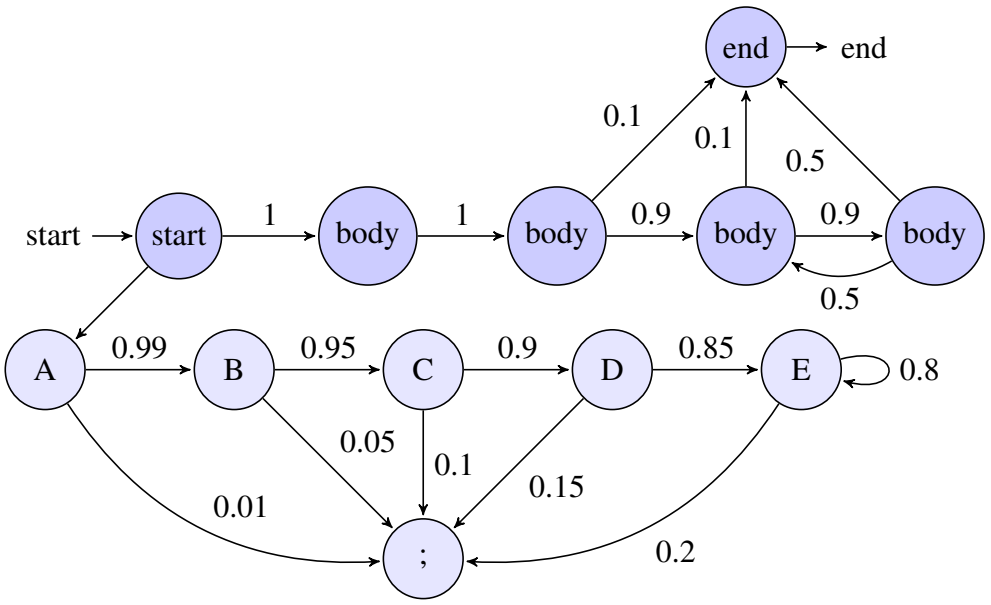

Figure 3.7: The extended duration-based model, hierarchical levels only. The second level is the same for all top-level states

tween note/duration pairs (a,i) and (b,j) is equal to the transition probabilities $a \rightarrow b$ and $i \rightarrow j$ multiplied together. This model had over a thousand states, which made training it further implausible.

Another approach, when combining the models for two separate viewpoints is infeasible, is to instead combine the predictions of the models [Conklin and Witten, 1995]. This is considerably less computationally intensive than combining models, and has the additional advantage that dissimilar models can be used; a phrase-based model cannot be combined with a bar-based model, but their predictions can be combined.

To combine the predictions of two models, the first model was used to parse the data normally; the predicted phrase boundaries were then added to the input data for the second model. The second model was then run on this 'semi-annotated' data to add its own predictions.

Chapter 4

Implementation

This chapter explores some issues relating to the implementation of the experiment, such as the use of a 'flattening' algorithm to work with HHMMs more efficiently, and the use of models for composition.

4.1 Flattening HHMMs

Wierstra [2004] describes a method for converting a hierarchical HMM to a non-hierarchical form which produces the same output. First, the hierarchical form must be minimally self-referential; that is, the internal states do not transition to themselves. Any self-referential HHMM can be converted to an equivalent (i.e. producing the same strings with the same probabilities) minimally self-referential form by redistributing probabilities. Doing this ensures that any two production states have exactly one route to transition between them. For two states i, j in a self-referential internal state a, the transition probability is augmented with the probability of going from i to the end state, remaining in a, and activating j:

$$T'^{a}_{i,j} = T^{a}_{i,j} + T^{a}_{i,E}.T_{a,a}.T^{a}_{S,j} \tag{4.1}$$

The transition probability to the end state is reduced to reflect the probability of a state going to the end state but re-entering the same internal state:

$$T'^{a}_{i,E} = T'^{a}_{i,E}(1 - T_{a,a}) \tag{4.2}$$

Finally, the self-referential transition probabilities $T_{a,a}$ are set to zero, and the probabilities re-normalised.

Once the HHMM is in minimally self-referential form, the end states and internal states are removed, and transition probabilities between states i, j in different internal states a, b with start and end states S, E are assigned using the formula

$$T'_{i,j} = T^a_{i,E}.T_{a,b}.T^b_{S,j} \tag{4.3}$$

To reverse this process, $T^a_{i,E}$, $T^b_{S,j}$ and $T_{a,b}$ must be recalculated

$$T^a_{i,E} = \sum_{j \notin a} T'_{i,j} \tag{4.4}$$

$$T^a_{S,i} = \frac{\sum_{j \notin a} T'_{j,i}}{\sum_{j \notin a} \sum_{k \in a} T'_{j,k}} \tag{4.5}$$

$$T_{a,b} = \frac{\sum_{i \in a} \sum_{j \notin b} T'_{i,j}}{\sum_{i \in a} \sum_{j \notin a} T'_{i,j}} \tag{4.6}$$

It can be seen that these reverse the initial transformation by substituting in equation 4.3. However, the flattening process does not map to all possible flat transition matrices; the models produced will satisfy certain constraints. For example, consider a model with states $i, j \in a$, $k \in b$, $l \in c$ (Figure 4.1).

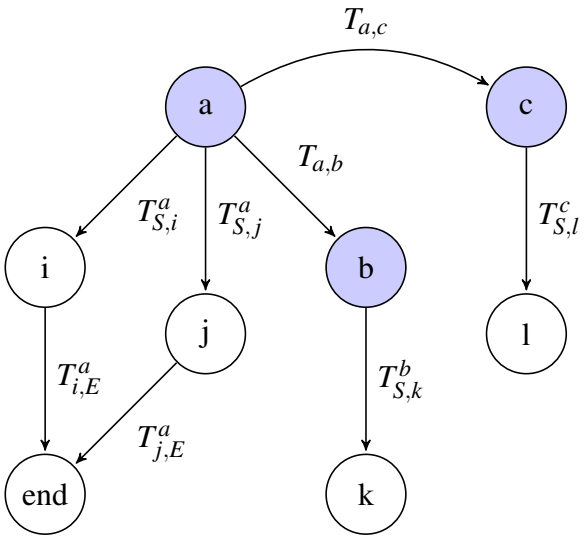

Figure 4.1

Then the flattened probabilities are:

$$T'_{i,k} = T^a_{i,E}.T_{a,b}.T^b_{S,k} \tag{4.7}$$

$$T'_{i,l} = T^a_{i,E}.T_{a,c}.T^c_{S,l} \tag{4.8}$$

$$T'_{j,k} = T^a_{j,E}.T_{a,b}.T^b_{S,k} \tag{4.9}$$

$$T'_{j,l} = T^a_{j,E}.T_{a,c}.T^c_{S,l} \tag{4.10}$$

It then follows that:

$$\frac{T'_{i,k}}{T'_{j,k}} = \frac{T'_{i,l}}{T'_{j,l}} \tag{4.11}$$

$$\frac{T'_{i,k}}{T'_{i,l}} = \frac{T'_{j,k}}{T'_{j,l}} \tag{4.12}$$

These constraints will in general not be maintained when training the flat model on data. If such a trained flat model M_f is converted to hierarchical form M_h using the previous equations, the two models will not be equivalent; running the flattening algorithm on M_h will not give M_f. For example, in the model described above, if the training data was such that state i was always followed by k, and j was always followed by l, the trained model would have transition probabilities

$$\begin{aligned} T'_{i,k} &= 1 \\ T'_{i,l} &= 0 \\ T'_{j,k} &= 0 \\ T'_{j,l} &= 1 \end{aligned}$$

Converting this flat model into a hierarchical form M'_h gives $T_{a,b} = T_{a,c} = 0.5$. These two models are not equivalent; the hierarchical model allows transitions which are impossible under the flat model, i.e. $i \to l$ and $j \to k$, and flattening M'_h will give a different flat model in which $T'_{i,k} = T'_{i,l}$.

The conclusion that can be drawn from this is that a hierarchical HMM cannot be trained in flattened form without losing the one-to-one correspondence with a hierarchical form. Wierstra [2004] offers an algorithm which optimises the hierarchical parameters of the model based on the Viterbi decoding of a sequence in the flattened form; however, extending this to an iterated training process based on multiple sequences proved infeasible for this project. The question is then whether it is necessary to maintain the hierarchical form once it has been flattened; do the additional constraints on the model improve it or not?

If a trained flat model is converted to a hierarchical form and that form is then flattened, a process I call 'reflattening', the result will be a slightly different model that satisfies the hierarchical constraints. Within an internal state, the state transition probabilities will be unaffected; the difference will be in the transitions between states within different internal states. The total probability of a state transitioning to a different internal state will remain the same, but it will be distributed among the different destination

states according to the internal state transition probability and the destination state's activation probability.

In the reflattened form, the probability of the first state of an internal state is unaffected by the last production state of the previous internal state; it is only affected by the internal state. By contrast, the trained flat model allows the transition probabilities across internal state boundaries to be independent, modelling context across the boundary. In a musical context, where the internal states correspond to phrases, there is more independence between phrases than within them, but they are not entirely independent; for example, if one phrase ends with a high note, you would expect a low probability of the next phrase starting with a low note. Training the flat model without regard to the hierarchical structure enables this additional context to be taken into account.

However, in a model with a large number of states within each internal state, the transitions between internal states may suffer from sparse data, as there may only be a few internal state transitions in each training instance. As such, combining the data, as is effectively being done by reflattening, may give a better approximation. It is unclear what are the relative strengths of these effects, but it might be expected that reflattening would be better on models with larger sub-models, while the trained flat model would perform better for smaller models.

4.2 Algorithms

The models were implemented in MATLAB using the Statistics Toolbox; the models were trained using the 'hmmtrain' function implementing the Baum-Welch algorithm. This function allows for smoothing of counts to account for unseen data by use of a 'pseudotransition' matrix; I simply used a matrix with value 0.0001 for all transitions with non-zero probability in the original transition matrix. A more fine-tuned matrix could possibly be used to better effect.

For parsing, the function 'hmmviterbi' was used to find the most likely state sequence using the Viterbi algorithm. Since the models were trained with phrase-ending notes as different emissions, for parsing a different emission matrix was used in which the end states emitted the unannotated notes; the end states can still be inferred from the state sequence. For the second stage of the two-stage parser, in which the predictions of the first model were added as phrase endings to the unannotated data, a third emission matrix was required; the parser needed to be able to acknowledge these predictions while still being able to make new predictions.

The initial version had non-end states emitting only unannotated notes, while end states emitted unannotated notes with probability 0.99 and phrase-ending notes with probability 0.01. This meant that the model was forced through the end states for the previous predictions, without significantly reducing the probability of predicting an end state at an unannotated note. The second version was changed to allow non-end states to emit phrase endings with probability 0.01, while the end states' probability was changed to 0.05. This meant that notes that had been predicted as phrase ends by the first model could be parsed as non-end states by the second model, effectively removing the first prediction, although maintaining the prediction was more favoured.

I coded the HHMM flattening and unflattening algorithms myself, based on the descriptions in Wierstra [2004] and Weiland et al. [2005].

4.3 Composition

Since it was impractical to produce a single model which could produce output of both note duration and pitches, conforming to both bar and phrase structure, it was necessary to find a way of creating output using more than one model. In particular, the phrase boundaries should coincide. The first approach tried was to first generate a melody using the pitch-based model. The phrase endings from this melody were then given to the duration-based model to find the most likely state sequence giving those phrase endings; this gave the note durations. However, since there is no random element in the selection of durations, this gave very predictable output; almost all durations selected were crotchets.

The second approach introduced an element of randomness. Again, a sequence of notes was generated by the pitch-based model, and the resulting phrase endings were fed to the duration-based model. However, instead of producing the most likely state sequence, MATLAB's 'hmmdecode' function was used to produce a matrix of the state occupation probabilities at each time step. At each step, if the state occupation probability is non-zero, that state has non-zero backwards probability; ie there is a valid path through that state that could generate the rest of the phrase endings. The algorithm takes the following steps:

- Extract the row of the transition matrix corresponding to the current state as a vector $\mathbf{t}$.
- For each state s, set $t_s = 0$ if the state occupation probability of s at the current

time step is zero.

- Normalise $\mathbf{t}$ so that $\sum_s \hat{t}_s = 1$.
- Generate a random number $0 < p < 1$ and find the first state x such that $\sum_{s=1}^{x} \hat{t}_s > p$.
- Add the output for state x, set current state to x, increment time step, and repeat.

This method introduces a random element while ensuring that each selection still conforms to the phrase boundaries, and will not lead to a 'dead end'. An alternative would have been to multiply each transition probability by the corresponding state occupation probability, but I felt that this would favour the most likely path too much and might lead to the output being too predictable.

Chapter 5

Results

This chapter details the results of evaluating the various models I had designed and trained. A complete table of results can be found in Section 5.2.

5.1 Observations

The initial phrase-level pitch-based model gave a precision of 25.1% and recall 41.3%. Applying the reflattening process improved both measures slightly, but not significantly. Examining some of the predictions suggests that the model favours short phrases too much; it predicts many short phrases, resulting in a very low precision. All pitch-based models gave similar levels of precision and recall, with the 'flexible' model giving the best recall.

By contrast, the phrase-level duration-based models often predicted too few phrase boundaries, giving a low recall. However, the precision was better; the predictions being made were fairly accurate. This suggests that the pitch-based and duration-based models are somewhat complementary, and there may be a way of combining the two to create a model with both high precision and high recall. The final duration-based model, using the extended hierarchical level, gives a good balance of precision and recall; the variant in which all durations are produced by one state rather than one state per duration performs similarly well despite only having 38 states in total.

The performance of the bar-level model was somewhat surprising, since it was designed to model bar structure rather than phrase structure; however, it produced a significantly better precision than the phrase-based duration models, and good recall. Perhaps phrase endings are particularly common at certain points within the bar; it may also be because the bar model is not forced to make a certain number of predictions

in each chorale. The phrase-based models must predict at least two phrase endings to reach the end state; without this restriction, the bar-level model can avoid predicting any phrase boundaries when the choice is unclear.

In general, the duration-based models are not improved by reflattening. This is presumably because they have fewer states in each sub-model, so sparse data is less of an issue and the primary issue is the loss of context across internal state boundaries. In particular, the extended model loses a large amount of precision and recall from reflattening; since each internal state lasts for only one timestep, reflattening removes almost all context from the model. The most interesting effect comes from reflattening the bar model; it changes from high-precision, relatively low-recall to a lower precision and high recall. It appears that removing context from phrase endings causes the model to predict phrase endings at most long durations; this will capture most phrase endings but result in many false positives.

The combined pitch and duration model was created from the best-performing pitch and duration models that were 'compatible'; the reflattened flexible models. The resulting model gave a higher precision than either of its 'parents' and comparable recall, but it is still quite conservative with its predictions. It may be possible to combine the models in a way which favours end states more heavily.

The first combined predictions method used the same models as the combined model, and gave a lower precision but higher recall. In the first experiment, the predictions of the duration model had to be retained by the pitch model; they could not be removed. A second experiment instead made the first model's predictions favoured, but removable, in the second model; this gave an improvement in both precision and recall, so this method was used for the subsequent experiments. Changing to the bar-level duration model gave further improvements, resulting in the model with the best overall F-score.

The final experiment was something of a curiosity; given that the bar-based model had the best precision, and its reflattened version had the best recall, would combining their predictions be effective? The resulting predictions did give the best recall out of all experiments, although with a fairly poor precision.

5.2 Table of results

Model	Precision	Recall	F-score
Pitch-based models			
Original model	25.1	41.3	31.2
- reflattened	26.8	41.7	32.6
Four-state	23.7	42.5	30.4
- reflattened	20.5	41.7	27.5
Two-level	26.2	44.8	33.1
- reflattened	26.7	44.4	33.4
Flexible	23.7	47.9	31.7
- reflattened	22.0	50.2	30.6
Duration-based models			
Original model	61.9	37.1	46.4
- reflattened	61.0	36.3	45.5
Flexible	71.3	43.2	53.8
- reflattened	65.1	37.5	47.6
Extended phrase-level	57.3	56.0	56.6
- reflattened	36.1	31.9	33.9
- multiple emissions variant	55.1	52.1	53.6
Bar-level	85.3	52.6	65.1
- reflattened	43.9	77.6	56.1
Combined pitch & duration model	75.7	43.2	55.0
Combined predictions			
Phrase-level durations, pitches	64.4	54.2	58.9
- allowing for errors	68.4	59.0	63.4
Bar-level durations, pitches	73.4	65.5	69.2
Bar-level with reflattened bar-level	46.5	86.6	60.5

Chapter 6

Analysis

This section contains analysis of the results of the parsing experiments, and of compositions produced from the models.

6.1 Parsing analysis

One thing that is clear from the results is that note duration-based models performed generally better than pitch-based models. There are a number of factors that could contribute to this. For one, there are considerably more possible pitches than possible durations in the data, which means the pitch-based models have more states and are likely to suffer more from sparse data, which reduces the performance of the model. Another issue is the number of phrase-end states. In the duration-based models, phrase endings are only found on notes of duration 1, 2, 2. or 4, whereas in the pitch-based model almost any pitch can potentially be a phrase ending. As a result, the duration-based model can eliminate any possible phrase endings at shorter notes; the remaining notes will have a higher concentration of correct phrase endings, so the predictions are likely to be better. This could also explain why the duration-based models make fewer predictions; even if the 'prediction rate' is the same, the duration-based models will make fewer predictions because they have fewer candidates to choose from.

The simplest explanation, however, is that duration is a better predictor of phrase endings than pitch, at least for this data. The basis for predicting phrase boundaries based on pitch is that they often coincide with larger jumps in pitch; however, it seems that in this data these jumps are less common than phrase boundaries simply coinciding with longer notes. Nevertheless, the improvements that are gained by combining the duration and pitch models suggests that there is some validity to the pitch-based

models.

This improvement can be ascribed to two main factors. The model properties complement each other; the first model makes few predictions but has a high precision, while the second model has lower precision but makes more predictions. When the predictions are combined, adding in the predictions of the second model makes a more appropriate number of predictions, while having the first model's predictions as a guideline should improve the accuracy of the second model. In addition, when the two models are based in different viewpoints there is a benefit from the independence of the two observations; a phrase ending may be more obvious in one viewpoint than in another. The effect of this independence can be seen from the relatively modest increase in performance when the two models were both duration-based; since the observations were not independent, this did not give as great an improvement as the other prediction combination systems. The fact that there was still an increase in recall shows that the complementary strengths of the models do have an effect.

6.2 Comparison to other results

To put these results into context, it is important to compare the performance with that of other systems. In Temperley [2004], a rule-based system gave a precision of 74.6% and recall 75.5%, while in Bod [2001], the best probabilistic grammar-based system gave 76.6% precision and 85.9% recall. These are better results than in any of my experiments, but the difference is not so large as to be offputting; I believe that with more work on model structure a HHMM-based system could surpass Temperley's system, and possibly also Bod's. Note that both these systems were tested (and trained, in Bod's case) on the Essen folksong collection. This corpus is considerably larger than the set of Bach chorales; Bod was able to use a training set of 5251 songs and a test set of 1000. A HHMM-based system might perform better on this corpus, due to the larger training set making sparse data less of an issue. It is also unclear whether these folksongs are intrinsically easier or more difficult to parse than Bach's chorales.

For a more direct comparison, I performed some experiments on the same corpus, using non-hierarchical HMMs. The pitch-based model gave a precision of 23.4% and recall 46.0%, in a similar range to the hierarchical pitch-based models. It is perhaps unsurprising that the results are similar, given that the original hierarchical model remains in the same sub-model for every phrase except the first and last; it might be interesting to see if the hierarchical model is better at predicting the first phrase boundary in each

chorale.

The non-hierarchical duration-based model gave a precision of 80.2% and recall 31.3%. The recall is lower than any of the hierarchical models; with the high precision this suggests that this model is even more conservative with its predictions. Since there is no hierarchical structure to force it to predict a certain number of phrase endings before the end, it is free to make predictions only in 'safe' cases.

Finally, combining the predictions of these two non-hierarchical models using the two-stage prediction process gave a precision of 77.4% and recall 54.7%. Interestingly, this gives an F-score of 64.1%, better than the combination of the equivalent hierarchical models, even though the pitch model gave roughly the same results and the duration model was arguably worse. Perhaps the shared hierarchical structure reduced the independence of the two hierarchical models, so the information gain from combining them was reduced. However, the combination of the bar-level model with the hierarchical pitch model was still the best performance.

For the final experiment, I combined the bar-level model's predictions with those of the non-hierarchical pitch model. This gave a precision of 78.5% and recall 69.4%, an improvement in both measures. This leads to the somewhat counter-intuitive result of the best phrase boundary predictions being made by a combination of two models which do not model phrase structure at all. Evidently the phrase-level models still have a lot of room for improvement.

6.3 Composition analysis

For the evaluation of compositions produced by my models, I had considered using human evaluation; asking a group of outside observers to evaluate the compositions in areas such as the well-formedness of the phrase structure, the resemblance to real chorales and the general quality of the composition. However, on looking over the compositions produced, it was apparent that the flaws of the compositions were fairly obvious, and my own observations would suffice to evaluate them.

Figure 6.1 shows an example soprano line from one of Bach's chorales, taken from the training data. The typical properties of the Bach chorales can be seen in this; e.g. note the multiple sequences of consecutive ascending or descending notes. Phrase endings are typically found on long notes, before large jumps in pitch, or at the end of ascending or descending sequences, as seen with the second and fourth fermatas.

Figure 6.2 shows a composition from the first system, which generated a pitch

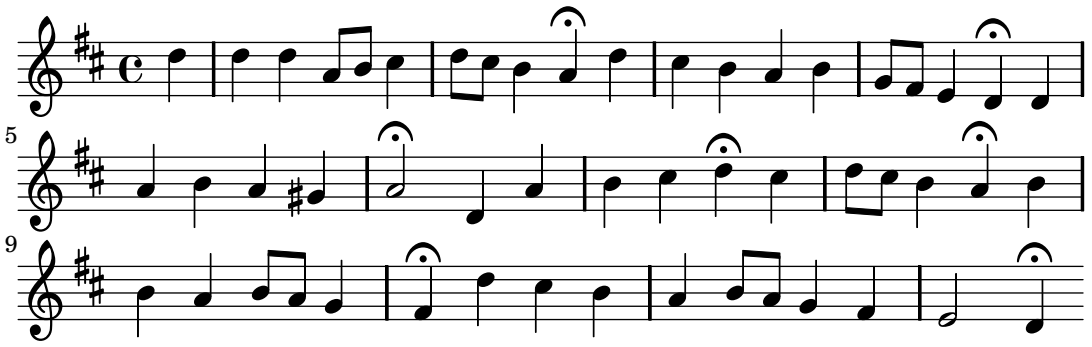

Figure 6.1: Bach chorale number 20, *'Ein' feste Burg ist unser Gott'*, soprano line only

sequence using the Markov model and then used Viterbi decoding to find the most likely sequence of durations with the same phrase boundaries. As a result, the selected durations are very predictable, consisting of mostly crotchets and no shorter notes; the real chorales have much more variety in note durations. Figure 6.3 is the result of changing the system to attempt to remedy this; the durations are instead generated using Markov selection, with constraints to ensure that it chooses states compatible with the phrase boundaries. The resulting composition does have more variety in note durations; however, both systems share several other problems.

Note that both compositions have a large variance in phrase lengths, containing extremely short phrases only two or three notes long as well as considerably longer phrases; the authentic Bach has more consistent phrase lengths. Many of the phrase endings are also illogically placed; e.g. the first phrase ending in Figure 6.2 does not correspond with a long note, a jump in pitch or the end of an ascending or descending sequence. There is also more of a tendency towards repeated notes and going back and forth between two notes, rather than the 'scale-like' sequences in the authentic Bach.

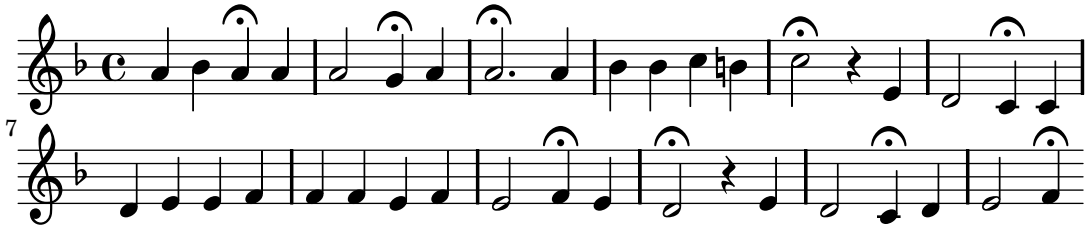

Figure 6.2: A composition produced by generating a pitch sequence and using the Viterbi algorithm to find the most likely sequence of durations

The next system was an attempt to solve the problem of inconsistent phrase lengths by including a third model in the generation process. The extended phrase-level du-

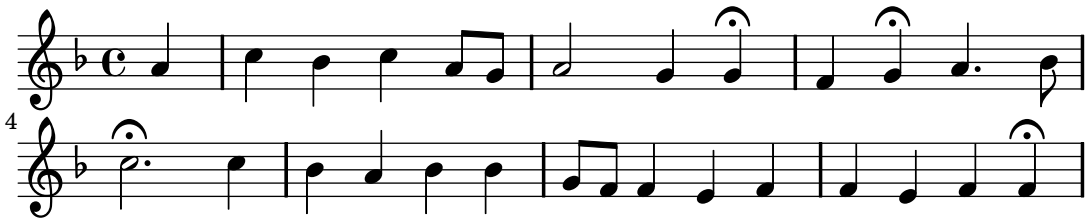

Figure 6.3: A composition produced by generating a pitch sequence and using constrained Markov selection to find a compatible sequence of durations

ration model seemed to give the best phrase lengths, but its sequences did not fit into bar structures; as such, it was used only to generate phrase endings. The pitch and durations were then both filled in using constrained Markov selection. Figure 6.4 shows a composition produced by this method, with additional annotations to indicate the results of parsing it using the two-stage method. A green fermata indicates a phrase ending predicted by both the generator and the parser, a red fermata indicates a generated phrase ending which was not predicted by the parser, and a blue fermata is one predicted by the parser but not generated.

The generated phrase lengths certainly seem more natural than in the previous compositions. Many of the phrase boundaries also seem more logical; this could be because the longer phrases have more 'character'; with more notes in a phrase, it is more likely that an ascending or descending sequence could emerge, which the phrase ending can then be seen as the culmination of. It is interesting to note the differences in phrase boundaries between the generator and the parser; for example, the parser fails to predict the first phrase boundary, despite it being a fairly strong phrase ending, as a long note at the top of an ascending sequence. By contrast, the semibreve in bar 9 was predicted as a phrase boundary by the parser, and as a long note followed by much shorter notes seems a strong candidate, but it was not intended as one by the generator.

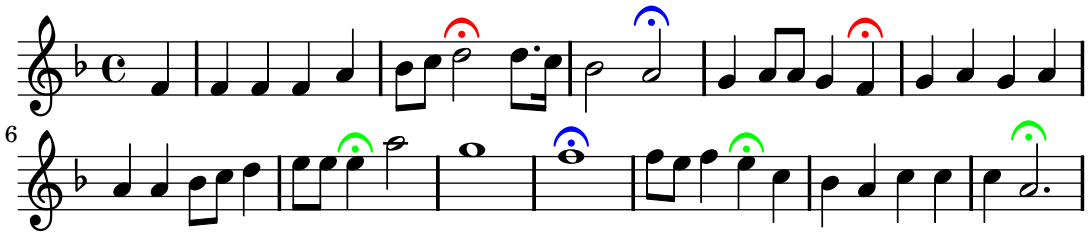

Figure 6.4: A composition produced by generating a sequence of phrase lengths and using constrained Markov selection for both pitch and duration, with fermatas as generated in red and green, and fermatas predicted by the parser in green and blue

To try and improve the generated phrase boundaries further, the next system used the reflattened versions of the pitch and duration models; since reflattening removes the context across phrase boundaries, the start of a phrase will be independent of the last note of the previous phrase, so the phrase boundary should be more clear. The results can be seen in Figure 6.5; for this composition, I annotated the phrase boundaries myself rather than using the parser. The obvious problem with this system is that the lack of context across phrase boundaries means that it is possible to get pitch jumps which are far too large, as seen in bar 3. However, the phrase boundaries do seem better, with only one minor correction, although this is a fairly short composition. Figure 6.6 shows the results of reflattening the duration model but not the pitch model; it avoids including massive jumps in pitch, but still places most of the pitch boundaries sensibly, usually on long notes.

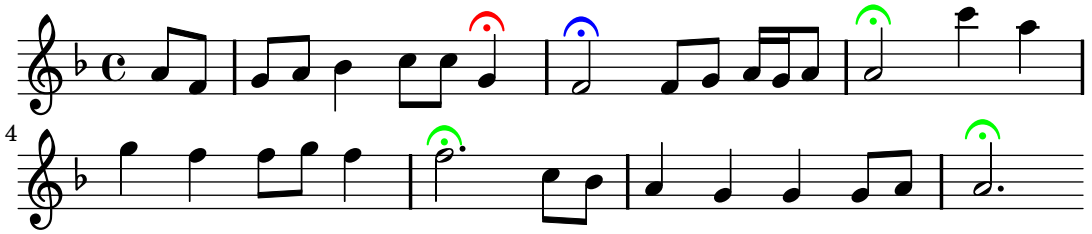

Figure 6.5: A composition produced using the reflattened models for both pitch and duration, with my own annotations to the fermatas

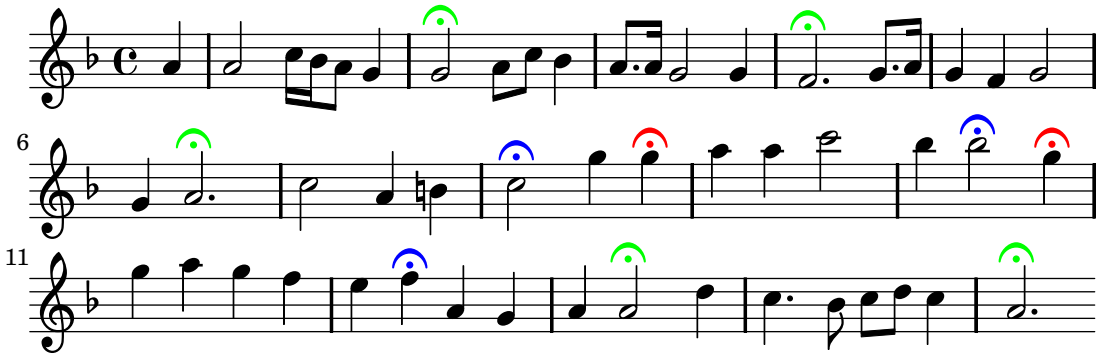

Figure 6.6: A composition produced using the reflattened model for duration but not for pitch, with my own annotations to the fermatas

These final systems have solved many of the obvious issues with the earlier attempts, but they still do not seem quite authentic. The reason for this is most likely due to the phrase contours. Note that in Figure 6.6 the difference in pitch between the first and last notes of a phrase is usually fairly small, and the notes in the phrase

go up and down in a roughly equal ratio. Contrast with Figure 6.1, where phrases often start and end at fairly distant pitches, and phrases are often dominated by either upwards or downwards movement, particularly with ‘scale-like’ sequences of consecutive notes. To capture this property, the pitch model would have to be redesigned with more consideration for structure within a phrase.

Chapter 7

Conclusion

This chapter gives an overall conclusion about the success of the parsing and composition experiments, and suggests some directions for future work.

7.1 Parsing

It cannot be said that the pitch-based hierarchical models were particularly successful at parsing; they performed no better than a non-hierarchical model. This suggests that the hierarchical structures that were used were not particularly effective. On the other hand, the results using duration-based models were more encouraging, with both the phrase-level and bar-level models showing an improvement in parsing performance over the non-hierarchical models.

There was also a significant increase in performance from combining the predictions of the two types of models; the best combined system produced results that were inferior, but at least comparable, to those in Temperley [2004] and Bod [2001]. There are a number of factors which could account for some of this difference, however. The Essen folksong collection used by Temperley and Bod is a considerably larger corpus than the Bach chorales; the larger volume of potential training material could improve a HHMM-based model significantly. It may also be that the use of the flattening algorithm is not the best way to train a HHMM; using a method that is more faithful to the hierarchical structure may also improve performance.

Overall, the use of HHMMs for parsing shows promise; the performance is not far off that of existing systems, and there are a number of potential changes which could bring the performance in line or above these alternatives. In particular, there is definite room for improvement in the hierarchical structures of the pitch-based models.

7.2 Composition

While the products of the composition systems show a definite improvement over the course of the experiments, they cannot be said to have fully captured the style of Bach. In particular, the phrase contours are rather flat and directionless compared to the authentic Bach. This again indicates the weakness of the pitch-based models. The duration-based bar-level model has a somewhat simpler task, but seems to produce fairly authentic output, and tend to assign longer notes to phrase endings. This model demonstrates one of the strengths of HHMMs; the design of the model is such that the structure guarantees that output conforming to the bar structure will be produced, without the need to impose external rules onto the output. With more work on the pitch-based model to improve the phrase contours, the system could produce strong musical output.

7.3 Future work

It would be informative to re-run the experiments using the Essen folksong collection for training and test data, as in Bod [2001]. This would allow for a more direct comparison with the results of other systems, and could show the effect of a larger training set on performance. It would also allow for comparison between the models after training on two different data sets; to see how the learned transition probabilities differed, and to compare the compositions that were produced.

One area in particular that future work could focus is in improving the hierarchical models used, particularly the pitch-based models, and investigating the effect of different hierarchical structures. As mentioned above, one problem with the pitch sequences that were generated was that the overall pitch contours of the phrases were too flat. With this in mind, one approach could be to put more focus on the internal structure of a phrase, and have different internal states corresponding to different phrase contours; one state in which upwards transitions were favoured, another which favours downward transitions, another which switches between the two partway through, producing an ‘arc’ contour, &c.

Another possible approach would be incorporating contour into the states, so each pitch would have three states corresponding to a higher, lower or equal pitch to the previous state. This would effectively incorporate a little more context into the model, and should give the harmonic movement some ‘momentum’; if the previous transition was

ascending, the next would be more likely to be ascending also. This should produce more 'scale-like' sequences in the output, making it appear more authentic. Alternatively, a model based solely on pitch contour could be used; this would require fewer different emissions and so could make use of higher-order context without increasing the number of states too drastically.

There are also improvements that could be made aimed at improving the parsing performance. Originally, I intended to use the parsing task primarily as a means of finding the best models for use in composition, but through the course of the project it became clear that the two tasks have somewhat different requirements. Using models designed with parsing in mind rather than composition could result in better parsing performance. For example, due to compositional concerns, I dismissed the possibility of having states which could emit more than one different pitch, but in a model intended solely for parsing, this could be useful. A hidden-layer representation based on chords, similar to Allan and Williams [2005], would require fewer states and might prove more effective at capturing the cadences found at the end of phrases.

There are fewer obvious areas for improvement in the duration-based models, but one is in creating a phrase-level model more suited to working with durations. As the phrase-level models were based on the pitch-based models, they assume that the start and end phrases are dissimilar from the body phrases; but this seems to be less the case with durations. A more appropriate phrase-level structure might differentiate between 'slow phrases' in which transitions to long durations are favoured, and 'fast phrases', for example; this could prevent situations where a phrase is generated with several long notes and the phrase ending on a shorter note.

A more general suggestion, which applies particularly to the bar-level, relates to shared sub-models. The bar-level model contains four instances of the crotchet sub-model; however, during flattening these are mapped to different states, and so are trained separately. They could be recombined when the model is reflattened, but it might be better if they could be trained as one model. This is apparently possible using the dynamic Bayes network approach [Murphy and Paskin, 2001], which was beyond the scope of this project. It might be interesting to see if training models using this approach gave better results than using the flattening approach used here. Alternatively, the original HHMM algorithms from Fine et al. [1998] could be used; although they take longer than using the flattened form, maintaining the hierarchical form throughout might prove to work better.

Appendix A

Additional compositions

This appendix contains some further examples of typical compositions produced by the systems discussed in the text.

Figures A.1 and A.2 were produced using the three-stage generator, using the original (i.e. not reflattened) models.

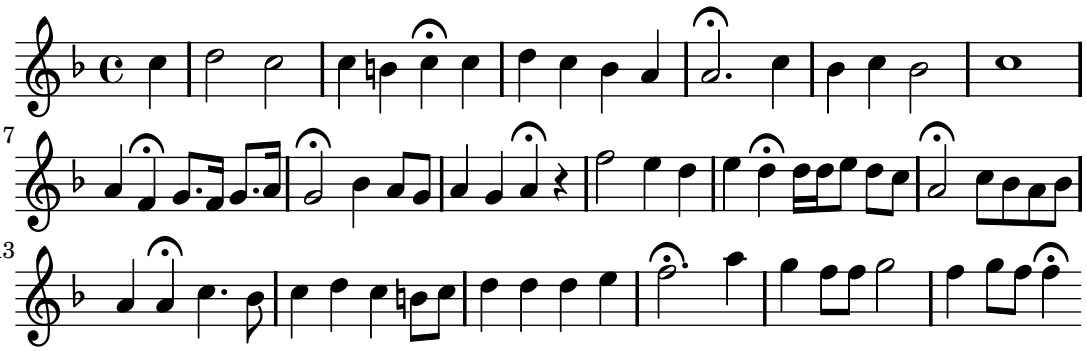

Figure A.1

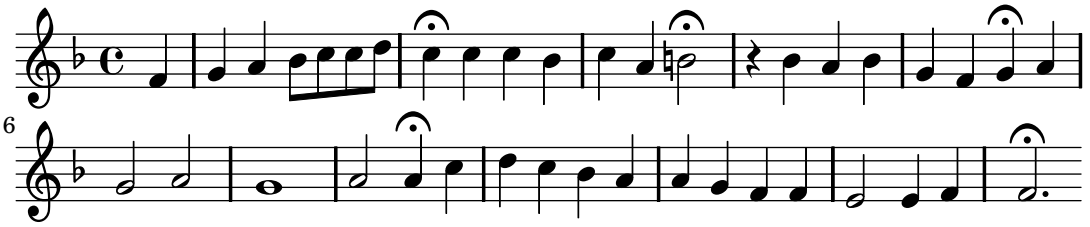

Figure A.2

Figures A.3 and A.4 were produced using the reflattened duration model and the original pitch model.

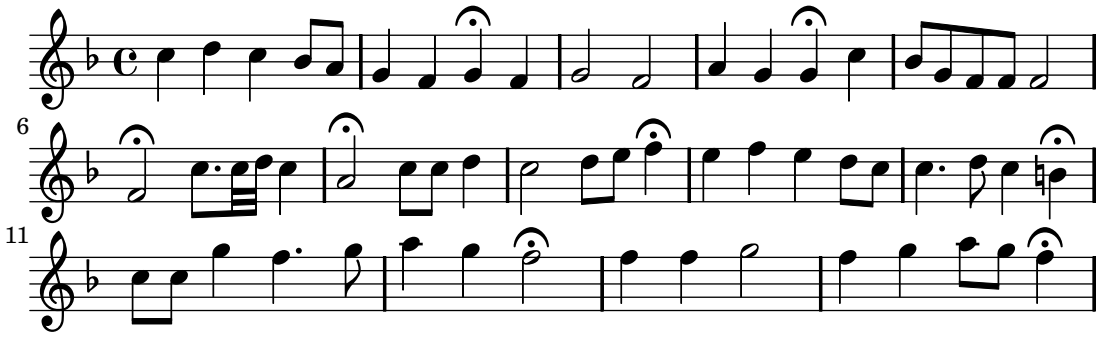

Figure A.3

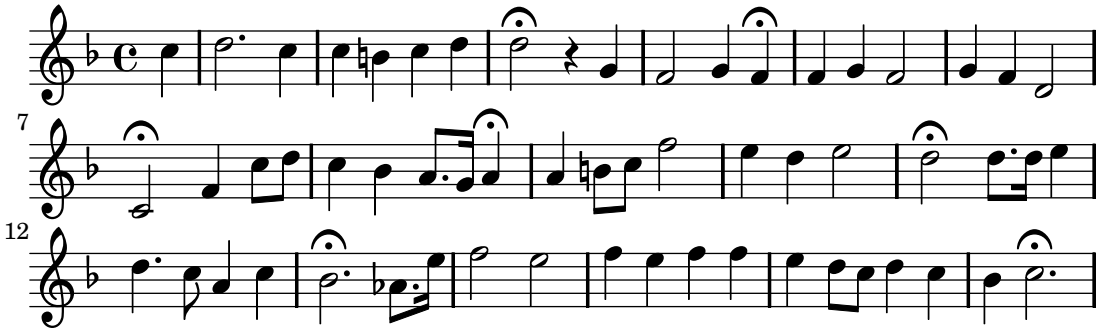

Figure A.4

Bibliography

M. Allan and C.K.I. Williams. Harmonising chorales by probabilistic inference. In *Advances in Neural Information Processing Systems 17: Proceedings Of The 2004 Conference*, 2005.

R. Bod. Probabilistic grammars for music. In *Belgian-Dutch Conference on Artificial Intelligence (BNAIC)*. Citeseer, 2001.

H. Chan and D. Ventura. Automatic composition of themed mood pieces. *on Computational Creativity 2008*, page 109, 2008.

D. Conklin and I.H. Witten. Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73, 1995.

A. Eigenfeldt and P. Pasquier. Realtime generation of harmonic progressions using controlled Markov selection. In *Proceedings of ICCC-X-Computational Creativity Conference*, 2010.

S. Fine, Y. Singer, and N. Tishby. The hierarchical hidden Markov model: Analysis and applications. *Machine learning*, 32(1):41–62, 1998. ISSN 0885-6125.

F. Lerdahl and R. Jackendoff. *A generative theory of tonal music*. The MIT Press, 1996. ISBN 026262107X.

K.P. Murphy and M.A. Paskin. Linear time inference in hierarchical HMMs. In *Advances in neural information processing systems 14: proceedings of the 2001 conference*, page 833, 2001.

D. Temperley. *The cognition of basic musical structures*. The MIT Press, 2004. ISBN 0262701057.

M. Weiland, A. Smaill, and P. Nelson. Learning musical pitch structures with hierarchical hidden Markov models. Technical report, University of Edinburgh, 2005.

D. Wierstra. A new implementation of hierarchical hidden Markov models. Master's thesis, IDSIA (Istituto Dalle Molle di Studi sullIntelligenza Artificiale), 2004.