Harmonising Chorales in the Style of Johann Sebastian Bach

Moray Allan

Master of Science
School of Informatics
University of Edinburgh
2002

Abstract

This dissertation describes a chorale harmonisation system which uses Hidden Markov Models. We use a standard data set of chorale harmonisations composed by Johann Sebastian Bach. This data set provides a large number of stylistically similar harmonisations, and is freely available in a machine-readable format. We divide the data into training and test sets, and compare the predictive power of various models, as measured by cross-entropy, the negative log likelihood per symbol. Using Hidden Markov Models we create a harmonisation system which learns its harmonic rules by example, without a pre-programmed knowledge base. We assume that we only need to take into account short-term dependencies in the local context. However, we generate globally probable harmonisations, rather than choosing the locally most likely outcome at each decision. The results produced by the system show that pre-programmed harmonic rules are not necessary for automatic harmonisation. Statistical observation of training examples provides the harmonic knowledge needed to generate reasonable chorale harmonisations.

Acknowledgements

I would like to thank my supervisor, Chris Williams, for his ideas and encouragement while I worked on this project. I would also like to thank my parents for proof-reading this dissertation.

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.

(*Moray Allan*)

Table of contents

Chapter 1

Introduction

This dissertation investigates automatic harmonisation of chorales. This task can be described as follows: given a line of music, can we automatically create three further lines of music which will sound pleasant when played simultaneously with the original melody? We will examine how we can use an existing set of chorale harmonisations to get a machine to learn how to harmonise chorales, and how we can use the same data to evaluate the quality of the harmonisations it produces.

Even music students who have no wish to become composers are asked to compose simple pieces of music to show their understanding of the harmonic language of Western classical music. These exercises in composition often include writing harmonisations of chorale melodies. This task is seen as open enough to allow a student's skill to be judged, but constrained enough that it is not free composition. It is not originality that is being looked for, but an understanding of the basic 'rules' of harmonisation, codifications of aesthetic preferences. A machine learning approach to this task attempts to build as good a model as possible from example harmonisations. This model can then be used to predict likely harmonisations given the melodic and harmonic context, and by iterating across many contexts we can create a harmonisation for an entire melody.

In this dissertation we will use chorale harmonisations by Johann Sebastian Bach as our examples. These provide a relatively large set of harmonisations by a single composer; they are freely available, and well understood by music theorists.

After a discussion of various model types, a final system will be described which uses Hidden Markov Models to provide significant enhancements over previous harmonisation models, by taking into account probabilities over an entire chorale rather than making decisions based only on the local context.

Chapter 2 provides the background to the dissertation, including information about the musical background. A brief explanation is given of some music theory necessary to understand the task of harmonisation, and the origins of the chorale form are related. An overview is provided of previous machine-learning research relevant to chorale harmonisation: several approaches have been taken in the past, including constraint-based systems, genetic algorithms, sequence prediction methods, and neural networks.

Chapter 3 describes the data with which we will be working, the surviving chorale harmonisations of Johann Sebastian Bach. Various advantages of this data set are identified to justify its choice, notably that the data set was already freely available in an annotated machine-readable format. An example of a chorale harmonisation in this format is included, and the annotations provided by the edition are explained. Reasons are given for the division of the data into separate sets of training and test data.

Chapters 4 and 5 examine alternative ways in which we can create models of the chorale harmonisation data. In Chapter 4 sequence prediction using Markov models is described and applied to the data, while Chapter 5 describes the use of Hidden Markov Models. In both chapters an explanation is given of the assumptions made by the model type, and the mathematics needed to build the models. The predictive power of the various models is compared.

Chapter 6 describes a harmonisation model which was built taking into account the results of the previous chapters, and discusses the results which this final model produces. Some examples of chorale harmonisations generated by the model are included.

Chapter 7 asks what conclusions we can make from the preceding material. Suggestions for future work are given, including possible enhancements to the model described in Chapter 6.

Chapter 2

Background

This chapter provides an overview of the background to the work described in this dissertation. After a brief explanation of some music history and music theory which motivate the activity of harmonisation, we will explore some of the previous work relating to automatic harmonisation.

2.1 Musical background

2.1.1 Harmonisation

Since the middle ages, much of Western music has been polyphonic, with two or more musical voices being heard at the same time. In church choirs, for example, singers came to make intentional deviations from the set tune, when they found that this improved the overall effect of the music. Later, composers of written music began to write complex polyphonic music which needed careful planning in advance. By the eighteenth century a system of rules had developed, dictating what combinations of notes were allowed to be played at the same time or following each other. These rules are still relevant to much music today: while the forms of music have changed, the underlying system has seen few changes.

This tonal system of music classifies tunes according to a system of keys. In a melody line, a single pitch is the centre, to which the line must always return.

Similarly, a single chord is made the centre of the polyphonic texture, to which the lines must come together.

Notes are seen as equivalent to those of half or double their frequency, and can be used to perform the same functions. The intervening pitches are quantised, with eight notes (including both ends) allowable in any given key as we move from a note to that of double the frequency, so that this interval is known as an 'octave'. These eight notes are placed unevenly, since they are not in origin a sequence but are generated by ratios of frequencies. After the 2:1 ratio of the octave, notes in a 3:2 ratio are viewed as those which go best together – the note whose frequency is in a ratio of 3:2 to the fundamental note is the fifth in the scale. A ratio of 4:3 gives rise to the fourth note in the scale. One possible tuning for the remaining notes can be generated just by multiplying these ratios, or in practical terms by tuning the remaining notes on a keyboard instrument from these first three notes by repeating the ratios up and down the keyboard.

An alternative view sees the eight notes as a subset of thirteen notes in geometric progression, so that each note is in the same ratio to the next, 1: $\sqrt[12]{2}$. This view had not yet taken hold in the eighteenth century (though it is the system generally used today), but it highlights the unevenness of the gaps between the eight notes. Traditionally the larger gaps have been classified as tones and the smaller ones as semitones. In a 'major' key the gaps between the eight notes run tone – tone – semitone – tone – tone – tone – semitone. The seven distinct notes in an octave are named A to G. For historical reasons relating to much older arrangements from which the tonal system developed, the most natural note from which to start a major scale is C, since the scale derived from any other note will stray outside the notes named so far. These in-between notes are referred to indirectly, as the 'flat' (♭) or 'sharp' (♯) version of the note above or below. If we use ratios to tune a scale, rather than the artificial 'equal-temperament' given by working with twelfth roots, then C sharp, for example, is not the same note as D flat, or even C flat as B. 'Minor' keys use the pattern of gaps tone – semitone – tone – tone – semitone – tone – tone, so that the key of A minor, for example, gives the same notes as C major. However, minor keys are more complex than major ones, in that

the sixth and seventh notes may be sharpened in certain contexts.

Once this system of keys is established for monophonic melodies, the idea of keys can be extended to polyphony. If we want to choose notes to fit with a melody note, we will be choosing these from the scale in that note's key. So a basic C major chord is the chord containing the notes C, E, and G, adding the third and the fifth notes in the major scale from C. The different notes in an octave are also given names according to the harmonic functions that they play, so that, for example, the base note may be called the 'tonic', the fifth the 'dominant', and the seventh the 'leading note'.

As is hinted at by the above, a great deal of analysis has been associated with the tonal system. Not only has the system been deeply analysed, and everything given (often several different) names, but individual pieces of music can be analysed in great detail on multiple levels. Harmonic analysis may be performed over the several hour span of an opera, or over a single phrase of music, specifying the exact implications of each note. Indeed, the system itself encourages such analysis, or, equivalently, many composers thought about the harmonic functions of the notes they wrote in its terms.

Into this framework certain rules are introduced when students are taught about harmonisation. These additional rules also clearly represent codifications of aesthetic preferences, rather than natural laws, but some of them pre-date the tonal system. A famous example is the rule which says that 'parallel fifths' are not allowed. That is, if a voice plays the fifth note in the scale from the note which another voice is playing, then when these voices move to their next notes the relationship between them must change.

2.1.2 Chorales

Since the sixteenth century the music of the Lutheran church had been centred on the 'chorale'. Chorales were hymns, poetic words set to music. A famous early example is Martin Luther's chorale 'Ein' feste Burg ist unser Gott'.

Chorales at first had only relatively simple single melodic lines, but soon com-

posers began to arrange more complex music to accompany the original music. This chorale-based music was expressed in various different forms, some intended for performance by a trained choir, going beyond what could be expected from congregational singing.

The music with which we will be concerned here represents generally a rather straightforward treatment of the chorales, but one which still offered many compositional possibilities. The chorale tune is taken unchanged, and three other musical parts are created alongside it, supporting it and each other. To be interesting, the added lines of music should not fit too easily with the melody, but should not clash with it too much either, and should come together with it at the end. These needs can be expressed in terms of consonant and dissonant harmonies: a dissonance will improve the music, if it is resolved into a pleasant consonance.

This type of chorale harmonisation allows us to apply machine learning techniques to the harmonisation problem in a way that is somewhat detached from problems of melodic invention. By dealing with a much narrower field than free composition we can limit the variables we need to deal with, and come much closer to a precisely-defined problem. These reasons explain why chorale harmonisation is so often used to train and to test students, and why it is an appropriate arena in which to investigate machine learning methods.

2.2 Previous work

2.2.1 Constraint-based systems

Even while Bach was still composing chorales, music theorists were catching up with musical practice by writing treatises to explain and to teach harmonisation. Two famous examples, Rameau's *Treatise on Harmony* (Rameau, 1722) and the *Gradus ad Parnassum* by Fux (1725), show how musical style was systematised and formalised into sets of rules. This seemed, and still seems, quite reasonable given their presumed basis in acoustic facts, although clearly the rules remain only explanatory aids, describing a single aesthetic: they are the rules of a style rather

than of all possible musics.

This emphasis on systematisation leads to the idea of automatically-composed music. Once a piece of music has started it can seem that the notes themselves, or rather the rules we impose on them, are in control. The events which follow 'must' come to fulfil the musico-syntactic form that has begun. Can we therefore construct detailed enough rules to spell out this necessity, and draw it out as a mathematical conclusion? We can encode our rules as constraints, and use constraint-satisfaction techniques to search for a solution which satisfies their requirements.

Pachet and Roy (2001) provide a good overview of constraint-based harmonisation systems. As an example, one early system (Schottstaedt, 1989) takes rules from Fux and assigns them penalties of different sizes, according to the seriousness of each rule being broken. This system then conducts a modified best-first search to produce harmonisations. Using standard constraint-satisfaction techniques for harmonisation is problematic because the space and time needs of the solver tend to rise extremely quickly with the length of the piece. For example, the system developed by Tsang and Aitken (1991) uses 70 megabytes of storage to find a harmonisation for a melody 11 notes long. Pachet and Roy (1995) demonstrate that space and time needs can be dramatically reduced by dividing the problem up into simpler subtasks: they propose a two-part system, where individual notes must meet constraints to make chords, while these chords must meet constraints to make the finished harmonisation.

A similar approach to a related problem is described by Löthe (2000). The task proposed here is to compose minuets in the early classical style. This task is divided into simpler subtasks, with the overall structure of the piece planned first, then the harmonic shape and melody of individual phrases, then the bass. Each subtask can then be represented as a search problem. Löthe lists the various sources from which domain knowledge may be acquired: period literature, modern harmonic theories, interviews with experts, manual analysis of examples, computer analysis, computer composition experiments, and interviews with composers. Some of these are included as possible sources of 'implicit knowledge'

which is not included in the relevant literature. The approach to automatic harmonisation that will be proposed below effectively integrates knowledge acquisition into the overall task, by directly using statistics acquired from a data set to construct the model that will be used in finding solutions to harmonisation problems.

2.2.2 Genetic algorithms

Genetic algorithms are a relatively recent method (Holland, 1975) which can be used to find solutions to tasks where conventional constraint satisfaction techniques would take too long or require too much temporary storage space. Several systems have applied genetic programming techniques to harmonisation, for example, McIntyre (1994). However, Phon-Amnuaisuk and Wiggins (1999) are reserved in their assessment of genetic programming as applied to this problem. They perform a direct comparison with an ordinary rule-based system, and conclude that the performance of each system is related to the amount of knowledge encoded in it rather than the particular technique it uses. In their comparison the ordinary rule-based system actually performs much better, and they argue that this is because it possesses implicit control knowledge which the system based on the genetic algorithm lacks.

Towsey et al. (2001) also discuss the use of genetic algorithms in automatic composition. They suggest that a better fitness function would be obtained by measuring statistics about a population against the same statistics calculated for a collection of training data. They propose various potential features, each a simple numerical measure of elements of a piece, in terms of its pitches, tonalities, contours, rhythms, and repetitive patterns. Principal components analysis is suggested as a method for uncovering relevant features, and providing target values. Their analysis here is inconclusive, but this is almost certainly because they use only 36 training melodies, composed over a period of about five hundred years, rather than because this sort of statistical analysis would not be helpful.

2.2.3 Sequence prediction

Conklin and Witten (1995) create new chorale melodies using probabilistic finite-state grammars. They train their system on a sample of 95 Bach chorale melodies. (The chorale melodies were in fact written by many different composers over a long period, but we can perhaps assume that Bach picked melodies to harmonise which appealed to his own compositional sensibility). Conklin and Witten create a 'multiple viewpoint system' to make predictions that can be used to analyse existing melodies or to generate new ones. This system combines different 'viewpoints', each a simple model of a property of the musical sequence. These properties include, for example, the start time of an event, its pitch, duration, key signature, time signature, its position in the bar, its interval from the first event in the bar, and its interval from the first event in the phrase.

Conklin and Witten's system is similar to some of the systems described above in that it divides a complex task into simpler subtasks. The previous systems made a division into a number of subtasks to be performed sequentially, to deal with the complex constraint requirements of harmonisation. In contrast this system deals with the difficult long-term relationships in melodies by constructing a large number of simple models to be used in parallel, rather than trying to get a single model to learn the effects of both the closer and more distant melodic relationships. Multiple viewpoint systems use weighted linear combinations of Markov models of different orders; alternatively we could work with the products of the models which we want to use together. Hinton (1999) describes the use of such 'products of experts'. He explains how individual expert models can be combined using 'logarithmic opinion pools', where we look at the average of appropriately weighted log probabilities (since taking the sum of logarithms of numbers is equivalent to taking the product of the numbers). Hinton advocates this approach instead of using weighted arithmetic means because these products of experts are much more efficient in high-dimensional spaces.

One application of probabilistic finite state grammars to harmonisation is described by Ponsford et al. (1999). The data set used here is a selection of 84

saraband dances, by 15 different seventeen-century French composers. An automatically annotated corpus is used to train Markov models using contexts of different lengths, and the weighted sum of the probabilities assigned by these models used to predict harmonic movement. The longest contexts considered contain four symbols: longer contexts not only exponentially increase the size of the model, but offer progressively less benefit. The predictions made by a Markov model effectively suggest what fragment of the training data is to be repeated; in musical grammars long fragments are unlikely to recur, and we generally want to avoid having obvious quotations from the training data in our generated output. Ponsford et al. create new pieces first by random generation from their models, and secondly by selecting randomly-generated pieces which match a prepared template. Using templates gives better results, but the great majority of randomly-generated pieces will not match the template and so will have to be discarded, and another attempt made. Without templates the most likely piece is simply two identical major chords. Ponsford et al. note that even with the longer context of four symbols, the cadences are poor: genuine pieces tend to use formulaic sequences of notes in closing, which their models fail to produce.

2.2.4 Neural networks

Hild et al. (1992) use neural networks to harmonise chorales. The system described here divides harmonisation into three subtasks: first a 'harmonic skeleton' is built, then a 'chord skeleton' is instantiated from this, then 'ornamentation' is added to the notes of the chords. The harmonic skeleton predictor is a neural network, trained for each beat on a context of the previous three harmonies, the previous, current, and next melody notes, the position in the bar, and a stress marker. Final ornamentation is predicted by another neural network, also trained for each beat in the examples on an appropriate representation of context. The step between these, however, where chords are chosen to instantiate more general harmonies, includes constraint satisfaction. Parallel fifths, for example, are penalised by this Knowledge Engineering aspect of the system, so that they will

be filtered out when the best chord is chosen from all those compatible with the pre-decided harmony.

2.3 Goals for a harmonisation system

If we take the system developed by Hild et al. as an example, we can ask in what directions it would be interesting to develop it. We already know that Knowledge Engineering approaches can work quite well, but can a less supervised system succeed at chorale harmonisation? Hild et al.'s system is promising in that, like Ponsford et al.'s system, it breaks harmonisation down into simpler subtasks, but here it is unclear how much the Knowledge Engineering aspect is relied upon – can we adequately represent the complexities of chorale harmonisation by iterating subtasks, or is it really the constraint system that is doing the work?

Of course, no system we produce will truly be 'unsupervised', since such a system would need to be capable of considering all possible models, of all kinds, for its training data and choosing the very best. However, in the following chapters we will keep the aim of trying to construct a simple, relatively unsupervised system. To achieve this, decisions made by the model will be based as much as possible on statistics learnt from the training data rather than programmed-in biases. At the same time, so that we can plausibly obtain good results from a relatively small amount of training data, we will retain some amount of supervision by imposing an unchangeable division into subtasks. In the limited domain of chorale harmonisation this kind of supervision will prevent invalid chorales, but still allow the system to learn from a composer's individual style, something which is prevented when harmonic rules are provided in advance.

Each decision in Hild's system only looks for a local maximum: later iterations cannot go back and challenge decisions already made. The system is pursuing a 'greedy search', with no backtracking. Ideally we would like to be able to find the harmonisation which has the maximum probability over the whole chorale, rather than the local maximum for a particular attribute. Ponsford et al.

noted the problems their system had in predicting cadences, and although they blame this on the length of the context used by their system, if we look for the global maximum then we should be able to predict cadences even with a short context. By looking for the global maximum our system will end up planning ahead, choosing the note now that will put it in the best position for making later choices.

The rest of this dissertation will investigate creating a chorale harmonisation system, following these aims: it should be 'less supervised', avoiding pre-written rules about harmony; it should be simple, making decisions based on short-term models which look at the local context; it should be capable of being used to find globally probable harmonisations, rather than 'greedily' choosing the local maximum at each decision.

2.4 Predictability and cross-entropy

Before we investigate different systems it is worth asking how we can compare them, and what measure we can use to back up our feeling that a system is performing well or badly. Hild et al. state of their system that, 'An audience of music professionals judged the performance [...] to be on the level of an improvising organist.' While this evaluation is easy to understand at an immediate level, it is hard to know what it actually means. To start with, what did these musicians think was the general standard of improvising organists – are they giving the system a great commendation or is this a tactful way of saying that it makes mistakes? It would be preferable if we could conduct a more precise evaluation than this.

One way of evaluating a predictive system is to feed in an unseen harmonisation by Bach and ask how likely the system thinks it is. In general, a better predictor will allocate the unseen harmonisation a higher probability; averaging over a test data set should smooth out the noise and make this more reliable. We can compare the relative performance of systems by comparing 'perplexity': given the appropriate contexts, which system allocates higher probabilities to the events

which actually occur? The precise measure we will use, taken from information theory, is 'cross-entropy', or the negative log likelihood per symbol (Shannon, 1948). To estimate the cross-entropy, we iterate over some test data, summing the negative base two logarithm of the probability of each symbol given our model; finally, we divide by the number of symbols traversed. A better model of some data will assign a greater probability to the events which actually occur in the data, and will therefore have a lower cross-entropy.

Chapter 3

Data

This chapter provides a description of the chorale harmonisation data with which we will be working, and of the specific machine-readable edition which we will use. An example of a chorale harmonisation in the textual format provided by this edition is given, and the annotations included in the files are explained. Reasons are given for the division of the data into separate sets of training and test data.

3.1 Bach's chorales

The data set we will be using is the surviving chorale harmonisations of Johann Sebastian Bach. This has long been a standard data set among musicians, and more recently has been used in music-based machine learning work. Some of the chorales were published during Bach's lifetime, but most were edited after his death by his son Carl Philipp Emmanuel. They are available in several printed editions – for example, Riemenschneider (1941) – and also in various freely-available electronic editions. The edition used here (Bach, 1998) includes the chorales not only in a MIDI format suitable for playback using a sequencer, but a textual file format, annotated with bar numbers, phrase markings, and harmonic symbols.

The chorale data set is relatively large, containing 384 chorale harmonisations. Since all the harmonisations are by the same composer we can reasonably assume that a single model might be able to explain them all. Bach's chorales are rela-

tively well-understood, and although, for example, some chorale harmonisations composed for special occasions are especially complex, they do seem to share a coherent musical style with the simpler harmonisations.

In some cases Bach wrote more than one harmonisation of the same chorale melody. It could be argued that we should remove these as duplicates, but this sort of interference with the data does not seem justified. At the least, a chorale melody which appears several times is almost certainly more typical, so even if we end up effectively counting it twice we are only giving it a justifiable weighting.

These different harmonisations of the same melodies should also remind us that we should not attempt or expect to generate harmonisations precisely the same as Bach's. Even a perfect model would need external data to achieve this: for example, for what occasion was the chorale harmonisation originally composed?

Some important data that we will not take into account is the words of the chorales. Each chorale melody remains associated with the words of the hymn to which it originally belonged, and research has shown that at times Bach illustrates the meaning of the words in his harmonisation of the relevant fragment of music. It would be an interesting enhancement of a system like this to learn the emotional content of words in the chorale texts and to make this feature available to a generative model, but the system described here makes no attempt to do this.

3.2 Working with the chorales

The textual edition of the chorales was chosen for use for several reasons. Whereas MIDI is a binary data format, and cannot easily be read directly by humans, the textual format is easy to read. Computer packages for music notation often make some attempt to read in MIDI files and to present them in a score notation, but such transcription is hard to perform well, since the MIDI format was not designed to provide the necessary information. Moreover, the textual format is probably easier for non-musicians to read than a score format. The textual format is obviously easy to extend with additional data, and the files used here do in fact include data

that could not elegantly be represented in a MIDI file. A plain-text file format makes it easy to develop systems to parse and process the chorale data, without having to rely on a particular set of routines to read data in and write it out. It means that we can, for example, use standard tools to search for all occurrences of an event, a function that a music notation package might not support at all.

3.3 Data format

Figure 3.1 shows one of the chorale harmonisations from our data set, in score notation. Figure 3.2 shows the beginning of the same chorale harmonisation as it is actually stored, in the textual file format. The file begins with headers which include fields giving the chorale's number ('Choralname'), key ('Tonart'), and time signature ('Takt'). When the harmonisation itself is notated below the headers, each line represents the same passage of time. Four columns show what is happening in each of the four musical voices. If a new note starts during the time period covered by a line, its pitch appears there, in the appropriate column. Two further columns are used to number the beginnings of phrases ('Phrase') and bars ('Takt'), while the final column ('Harmonik') provides an annotation of the behaviour of the harmonisation, by giving a harmonic symbol to describe each beat. The harmonic symbols here are 't', representing a tonic minor triad, and 'D3', representing a type of dominant chord.

The textual format does have the disadvantage that it cannot be played back as music by standard tools. However, using a standard MIDI interfacing module (Burke, 2000) I wrote a short program in Perl (`chorale2midi.pl`), which reads in a chorale in plain text and feeds out a MIDI file. As a proof-of-concept I also wrote an equivalent program to reverse the process (`midi2chorale.pl`), producing a text file from the MIDI file. While this is not in itself very useful, since we already had the textual version, and the conversion process removes any annotations from the files, it shows that it would be possible to automatically convert MIDI files into the plain-text format if it was required in future. Given a set of data

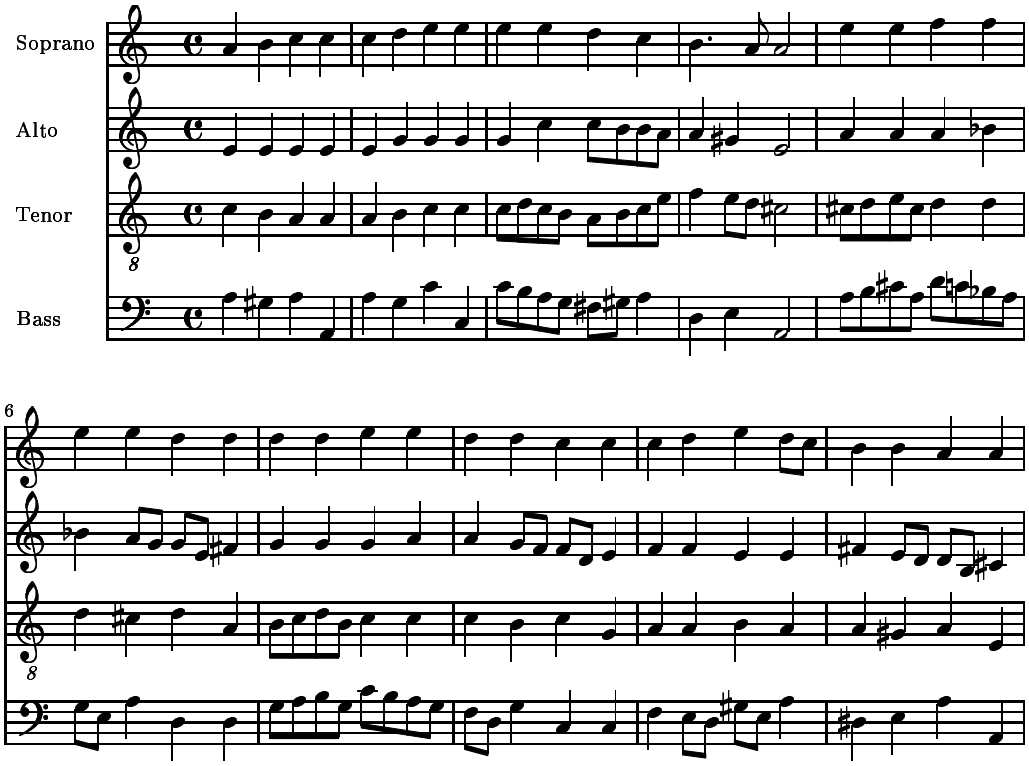

Figure 3.1: Chorale K11, BWV 26.6, ‘Ach wie nichtig, ach wie flüchtig’

```
Choralname = bch011
Anzahl Stimmen = 4
Tonart = A-moll
Takt = 4/4
Tempo = 100
Notentextausgabe in 16tel-Schritten:

PHRASE  TAKT     SOPRAN  ALT   TENOR  BASS     HARMONIK

     1     1      A 1    E 1    C 1    A 0     t


                  H 1    E 1    H 0    G#0     D3


                  C 2    E 1    A 0    A 0     t


                  C 2    E 1    A 0   A -1     t


     2     2      C 2    E 1    A 0    A 0     t
```

Figure 3.2: Beginning of data file for chorale K11, BWV 26.6, ‘Ach wie nichtig, ach wie flüchtig’

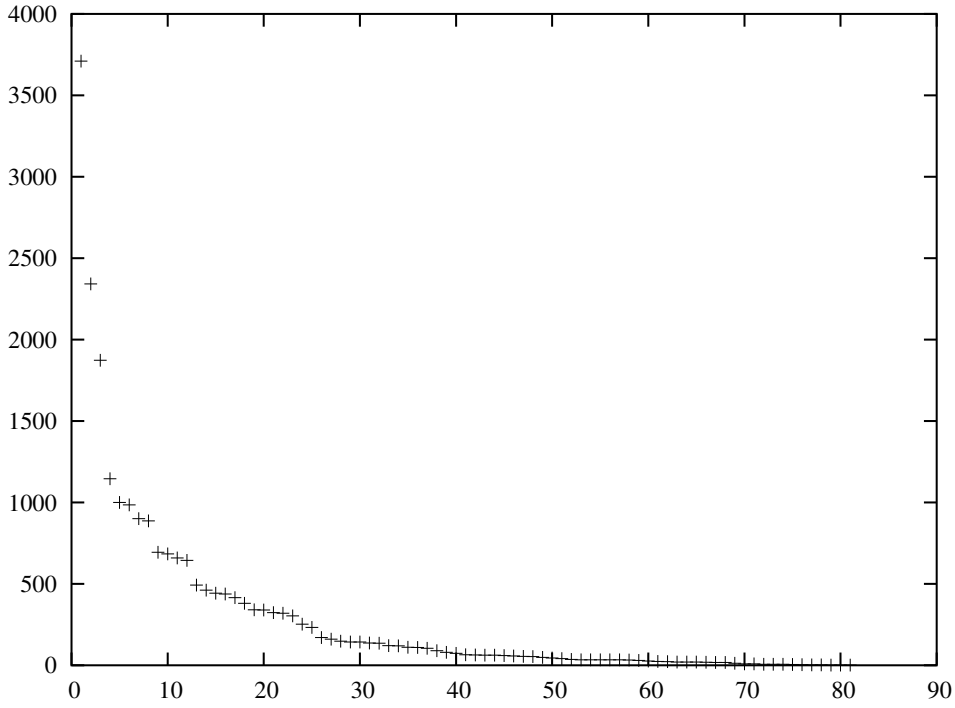

Figure 3.3: Frequency of the Nth most common harmonic symbol

in MIDI format, it would be possible to tune the conversion program, configuring for example the number of beats per minute that we wish to use, and to produce versions in the textual format without having to re-enter all the data manually. A further program (`chorale2lilypond.pl`) was written to produce output in GNU Lilypond format for typesetting in score notation.

In total 81 different harmonic symbols appear in the annotations of all the chorale harmonisations. The most common harmonic symbols are 'T', a tonic chord, and 'D', a dominant chord. The least common harmonic symbols are 'VTp5', 'Vd5' and 'SS5', which each occur only once in the entire set of harmonisations. As figure 3.3 shows, the harmonic symbol frequencies follow a Zipf-like curve, often found in statistical natural language. The data is sparse, and we cannot have any confidence that we will produce a good model of the least frequent occurrences.

To ensure that the programs created all used the same interpretation of the data

files, I wrote the necessary parsing routines and included them in a reusable Perl module (`Chorale.pm`).

3.4 Training and test data

Since chorales in major and minor keys are known to exhibit different harmonic behaviour, two separate groups of training and test data were needed. Once the chorales had been categorised by their major or minor key, four data sets were used for each category: a training set containing 40% of the available chorales, and three test sets each containing 20% of the available chorales.

Using several test data sets in this way allowed methods to be fairly tested on fresh test data after they had been optimised on another set of validation data. The third test data set was not used until the final system had been prepared, so that even accidental optimisation would not be carried out.

To remove any bias from the original ordering of the data, chorales were randomly allocated to these sets. The allocations made are listed in the appendix.

Chapter 4

Sequence prediction

This chapter shows how we can create Markov models of properties of chorale harmonisations. A description of the mathematical basis for various kinds of Markov model is followed by a comparison of the predictive power of different models of our data set.

4.1 Theory

Sequence prediction methods allow us to model, for example, states which vary over time. We can pick already-observed events which we think will affect what will happen next, and build a model to predict what will happen next from these observed variables.

Given a record of the weather over a period, we might hope to find patterns relating the weather on a day to the weather on the previous day, or on the previous few days. If we built a model of these patterns, we could then use it to predict the next day's weather, given our knowledge about the weather on the previous days. A very simple model might use only two states, 'sunny' and 'rainy'. Perhaps our model would say that if it is sunny today, there is a 70% probability it will also be sunny tomorrow, and that if it is rainy today there is only a 10% chance it will be sunny tomorrow. Since there are only two states in our model, and since the probabilities of all possible states must sum to 1, there must equivalently be a

30% chance that a sunny day will be followed by a rainy one, and a 90% chance that a rainy day will be followed by another rainy one. We can represent these probabilities as a matrix:

$$\begin{pmatrix} P(y_t = \text{sunny}) \\ P(y_t = \text{rainy}) \end{pmatrix} = \begin{pmatrix} 0.7 & 0.3 \\ 0.1 & 0.9 \end{pmatrix} \begin{pmatrix} P(y_{t-1} = \text{sunny}) \\ P(y_{t-1} = \text{rainy}) \end{pmatrix}.$$

We can use this equation iteratively to work forwards from a day which we know was sunny or rainy, and work out the probability of each future day being in either state. The model which this equation represents is a first-order Markov chain, since it only uses the state at one preceding time step to make its predictions.

In the case of the weather we would not expect the model's results to be reliable many days ahead, and in fact for most sequences of events such a simple model will be less and less useful the further ahead we try to look. However, for our purposes here we need only be concerned with short-term prediction, and with how much a sequence generated by our model looks like a genuine sequence. We can use the short-term predictions to calculate the likelihood of sequences from our data set according to our model, and when we generate new sequences we do not need to predict Bach's actual harmonisation, but something that is like a Bach harmonisation.

In general, for mutually-exclusive events S_0 to S_M, a first-order Markov model will take this form:

$$\begin{pmatrix} P(y_t = S_0) \\ P(y_t = S_1) \\ P(y_t = S_2) \\ \cdots \\ P(y_t = S_M) \end{pmatrix} = \mathbf{A} \begin{pmatrix} P(y_{t-1} = S_0) \\ P(y_{t-1} = S_1) \\ P(y_{t-1} = S_2) \\ \cdots \\ P(y_{t-1} = S_M) \end{pmatrix},$$

$$\mathbf{A} = \begin{pmatrix} P(y_t = S_0 | y_{t-1} = S_0) & P(y_t = S_0 | y_{t-1} = S_1) & \cdots & P(y_t = S_0 | y_{t-1} = S_M) \\ P(y_t = S_1 | y_{t-1} = S_0) & P(y_t = S_1 | y_{t-1} = S_1) & \cdots & P(y_t = S_1 | y_{t-1} = S_M) \\ \cdots & \cdots & \cdots & \cdots \\ P(y_t = S_M | y_{t-1} = S_0) & P(y_t = S_M | y_{t-1} = S_1) & \cdots & P(y_t = S_M | y_{t-1} = S_M) \end{pmatrix}.$$

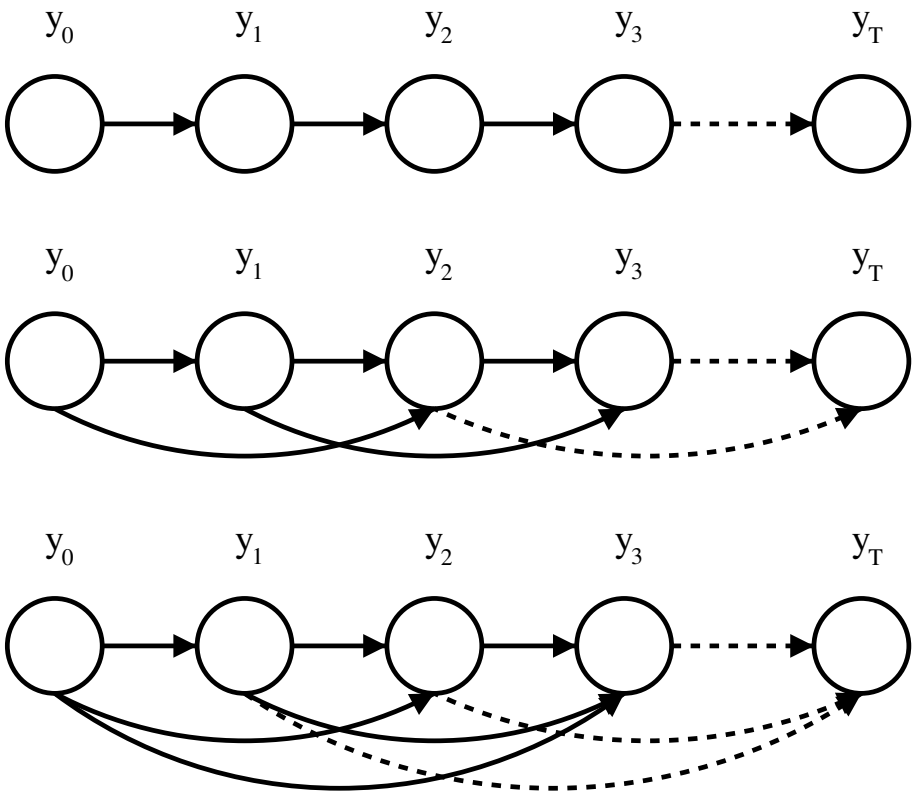

Figure 4.1: Markov chains of orders 1, 2, and 3

This model makes the assumption that y_t is conditionally independent of $y_{t-2}, y_{t-3} \dots y_0$, given y_{t-1}:

$$P(y_t = S_{i_t} | y_{t-1} = S_{i_{t-1}}, y_{t-2} = S_{i_{t-2}} \dots y_0 = S_{i_0}) = P(y_t = S_{i_t} | y_{t-1} = S_{i_{t-1}}).$$

If we want to create a model for some training data, we can use maximum likelihood estimates of these probabilities:

$$P(y_t = S_j | y_{t-1} = S_i) = \frac{P(y_t = S_j, y_{t-1} = S_i)}{P(y_{t-1} = S_i)} \approx \frac{freq(y_t = S_j, y_{t-1} = S_i)}{freq(y_{t-1} = S_i)}.$$

If we instead claim that we also need to take into account y_{t-2}, we get a second-order Markov chain; if we claim that we need to take into account y_{t-1}, y_{t-2} and y_{t-3}, we get a third-order Markov chain. These three models are represented graphically in figure 4.1. Markov chains are very well-understood models, and

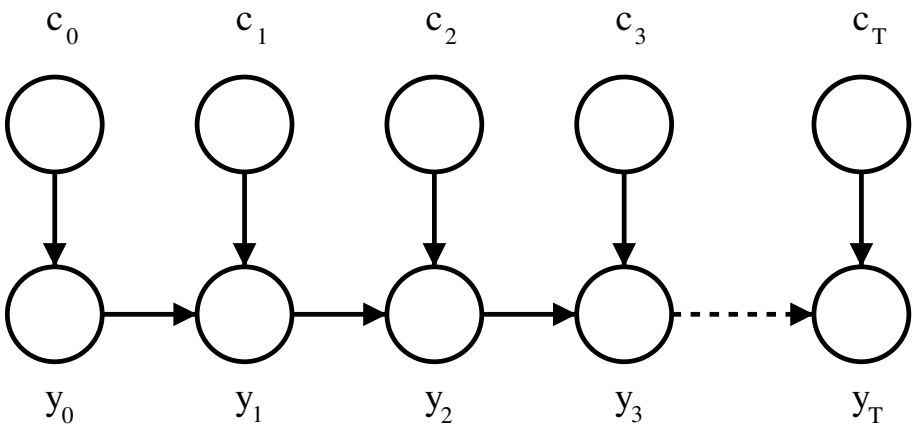

Figure 4.2: Markov model with additional context

have a long history of use in modelling aspects of language. Markov himself used them to model sequences of letters found in a text (Markov, 1913).

We do not need to only take into account the previous values of the variable which we are trying to predict. If we can observe other relevant variables, then we can include these in our model. Figure 4.2 shows a first-order Markov model where the next state of a variable is also dependent on some external context. We can easily include this context in our maximum likelihood framework:

$$P(y_t = S_k | y_{t-1} = S_i, c_t = V_j) \approx \frac{freq(y_t = S_k, y_{t-1} = S_i, c_t = V_j)}{freq(y_{t-1} = S_i, c_t = V_j)}.$$

While these estimates will maximise the likelihood of the training data with respect to our model, they may prove problematic when we try to apply them to new data. When the frequencies of items in a data set are distributed according to a Zipf-like curve it is likely that we will come across items in unseen data which were not present in our training data. Indeed, the longer the sequences we consider, or the more additional context we use, the more likely it becomes that we will come across something which had a zero frequency in our training data.

We can mitigate this problem by, for example, constructing several Markov models of different orders, and using them together. We can 'smooth' the probabilities we assign by using a weighted average of the individual model probabilities:

$$\begin{aligned} P^*(y_t = S_k | y_{t-1} = S_j, y_{t-2} = S_i) &= \lambda_0 P(y_t = S_k) \\ &\quad + \lambda_1 P(y_t = S_k | y_{t-1} = S_j) \\ &\quad + \lambda_2 P(y_t = S_k | y_{t-1} = s_j, y_{t-2} = S_i). \end{aligned}$$

Appropriate values for the weights λ_k can be calculated using a validation set of held-out data. A simpler method which can also work well is to 'back off' to a model which assumes fewer dependencies when we find an item with a zero frequency (Katz, 1987; Chen and Goodman, 1998). For example:

$$P^*(y_t = S_k) = \begin{cases} P(y_t = S_k | y_{t-1} = S_j, y_{t-2} = S_i), & \text{if } A; \\ P(y_t = S_k | y_{t-1} = S_j), & \text{if } \neg A, B; \\ P(y_t = S_k), & \text{if } \neg A, \neg B. \end{cases}$$

$$\begin{aligned} A &: freq(y_t = S_k, y_{t-1} = S_j, y_{t-2} = S_i) > 0. \\ B &: freq(y_t = S_k, y_{t-1} = S_j) > 0. \end{aligned}$$

Backing off relies on the assumption that the models which assume more dependencies are better models of the underlying processes, so that a model which assumes more dependencies should be used in preference to a model which assumes fewer dependencies.

Smoothing across models like this still does not help us with individual events in unseen data which were not present in our training set. To deal with this, we need to smooth the conditional probabilities $P(y_t = S_i | \mathbf{C})$ over all possible i for each context $\mathbf{C}$. One simple possibility is to use 'additive smoothing', adding some value δ to all the observed counts and renormalising (Lidstone, 1920; Nivre, 2000). This modifies our probability estimate: with N the number of observed states S_j,

$$P(y_t = S_j | y_{t-1} = S_i) \approx \frac{freq(y_t = S_j, y_{t-1} = S_i) + \delta}{freq(y_{t-1} = S_i) + \delta(N+1)}.$$

To calculate the cross-entropy (negative average log-likelihood per symbol) for a Markov model, we can simply iterate over the test data, summing $\log P(y_t|\mathbf{C})$ for whatever context $\mathbf{C}$ we are using, and once we come to the end divide by the number of symbols we have traversed.

4.2 Application

The simplest sequence prediction model we can apply to the harmonisation problem is a first-order Markov chain using the harmonic symbols with which our data set is annotated. We can also produce higher order Markov chain models of the harmonic sequence, and use these models in combination. These models will not be useful generative models, since they take no account of the melody line in making their choices – they will produce the same sequences whatever the melody. However, they do allow us to look at the intrinsic predictability of the harmonic symbols.

Another very simple, but more powerful, model, uses the melody notes to predict the harmonic symbols. Again, Markov models of different orders can be used in combination. Whereas we can obviously use only the preceding harmonic symbols, we can use the current melody note, so these models have additional information about the harmonic behaviour at the current step, compared to models which only use the harmonic symbols themselves.

A third possibility is to use both notes from the melody and the preceding harmonic symbols.

The chorale data files are already organised as sequential data. For these models we only need to look at two columns of the data (‘Sopran’ and ‘Harmonik’), and we are only interested in every fourth line in these columns: these lines represent the beat of the music, and are the only lines on which harmonic annotations are given. When a file is read in, special symbols are added to the beginnings and

ends of the sequences. This means that we do not, for example, need to maintain a separate matrix of initial symbol probabilities, since the usual training method will produce appropriate probabilities for the transitions from the beginning-of-sequence marker. Chorales are transposed into C major or C minor as they are read in, so that we do not have to create a separate model for each key but can use all our data to create a overall models of the major and minor key harmonisations. Events which continue over multiple time steps are represented by their plain symbols when they begin, and then at subsequent time steps by modified forms of their symbols which have a prefix added to show continuation.

I wrote a library of routines to train and use Markov models, and used this to create these models from the chorale data training set. The results are described below. We will use backing off to smooth across models of different orders, and additive smoothing to smooth frequencies to account for unseen data.

4.3 Results and discussion

The various models described above allow us to examine the relationship between the harmonic symbols with which the data is annotated and the melody notes in the soprano line. By comparing the predictive power of these models we can discover how useful different pieces of contextual information are.

To investigate how useful contexts of different sizes would be, I trained Markov models on the major and minor training data using contexts of up to eight symbols in length, then iterated over test data using each model in turn, noting how many sequences were encountered which had not been seen in the training data. The results for models predicting the next harmonic symbol from the preceding harmonic symbols are shown in table 4.1, the results for models predicting the next harmonic symbol from the melodic context in table 4.2, and the results for models using both harmonic and melodic context in table 4.3. In each case there are two sets of results, since the chorales in major and minor keys are treated separately.

Context	Number	Proportion
0	0	0%
1	214	8.10%
2	760	28.8%
3	1456	55.2%
4	1950	73.9%
5	2216	84.0%
6	2349	89.0%
7	2397	90.8%
8	2425	91.9%

Context	Number	Proportion
0	0	0%
1	147	6.76%
2	612	28.2%
3	1304	60.0%
4	1702	78.3%
5	1888	86.9%
6	1984	91.3%
7	2018	92.9%
8	2031	93.5%

Table 4.1: Number of unseen sequences for various lengths of harmonic context, and proportion of all sequences unseen: major (left) and minor (right)

Table 4.1 shows the number of such unseen sequences for models predicting the next harmonic symbol from the preceding harmonic symbols, and the proportion of all sequences encountered which were unseen, for contexts from zero to eight symbols long. When a context of zero length is used, we only ask if individual symbols have been seen in the training set; as we increase the context length and look at longer sequences, we become more and more likely to find unseen sequences which were not present in the training data. For these models the number of symbols in the context is the same as the number of quarter-note beats we are taking into account in making our predictions, so the maximum contexts considered here of length eight represent, for example, two bars of music in common time. We can see that the proportion of sequences encountered in the test data which our models have seen in the training data quickly drops off as the context length increases.

Table 4.2 shows the equivalent results for models predicting the next harmonic symbol from the melodic context. The models using the melodic context have a consistently lower proportion of unseen symbols than the models which use the preceding harmonic symbols. This is understandable, because while there are

Context	Number	Proportion
0	0	0%
1	184	6.96%
2	452	17.1%
3	940	35.6%
4	1444	54.7%
5	1777	67.3%
6	1932	73.2%
7	2040	77.3%
8	2096	79.4%

Context	Number	Proportion
0	0	0%
1	127	5.84%
2	260	12.0%
3	673	31.0%
4	1072	49.3%
5	1327	61.1%
6	1486	68.4%
7	1579	72.7%
8	1618	74.5%

Table 4.2: Number of unseen sequences for various lengths of melodic context, and proportion of all sequences unseen: major (left) and minor (right)

Context	Number	Proportion
0	0	0%
1	603	22.8%
2	1436	54.4%
3	2022	76.6%
4	2283	86.5%

Context	Number	Proportion
0	0	0%
1	411	19.0%
2	1225	56.4%
3	1688	77.7%
4	1902	87.5%

Table 4.3: Number of unseen sequences for various lengths of harmonic and melodic context, and proportion of all sequences unseen: major (left) and minor (right)

more than eighty different harmonic symbols in the data set, we do not expect the melody notes to vary by much more than an octave. Taking this into consideration, the proportion of unseen harmonic contexts actually rises fairly slowly with the context length, showing that even for the longer lengths of context some sequences are repeated across different chorales.

Table 4.3 shows the number of unseens sequences for models using both harmonic and melodic context. These results look at how many unseen combinations of harmonic symbols and melody notes we find, so it is not surprising that the proportion of unseen sequences rises more quickly than when we take the harmonic symbols or melody notes by themselves. Since models use compound symbols in their contexts, with each contextual unit encoding a combination of a harmonic symbol and a melody note, a context four symbols long for these models is comparable to a context eight symbols long for the previous models.

Using the same set of models, we can see how this falling off in sequence coverage affects the quality of our predictions using the different contexts. Table 4.4 shows how the harmonic symbol based models perform, table 4.5 shows the performance of the models which base their predictions on melody notes, and 4.6 shows the performance of the models which use both melody notes and harmonic symbols.

In table 4.4 we can see how using different numbers of the previous harmonic symbols affects the quality of our predictions for the next harmonic symbol. As in all these tables, a model of order zero uses a zero length context – that is, it makes predictions based on the overall probabilities of different symbols without taking into account the context at all. If we use individual models, then as we increase the length of the context the models very quickly perform worse than the model of order zero. This is shown in the table by a higher cross-entropy value. The models which perform worse than the model of order zero are suffering from 'sparse data': there are too many unseen sequences for them to be able to consistently make useful predictions. Smoothing with models of lower orders overcomes this problem, since the predictions of the lower order models can be used when the higher order models come across an unseen sequence. We can see that smoothing

Model order	Single model	Smoothed models
0	4.31	4.31
1	3.80	3.80
2	4.37	3.48
3	6.78	3.28
4	10.0	3.18
5	12.4	3.15
6	13.7	3.14
7	14.3	3.14
8	14.5	3.14

Model order	Single model	Smoothed models
0	4.64	4.64
1	3.74	3.74
2	4.14	3.35
3	6.70	3.15
4	10.5	3.09
5	13.0	3.07
6	14.0	3.06
7	14.6	3.06
8	14.7	3.06

Table 4.4: Cross-entropies for models predicting harmonic symbols from previous harmonic symbols: major (above) and minor (below)

Model order	Single model	Smoothed models
0	4.31	4.31
1	3.20	3.20
2	3.46	2.95
3	4.39	2.76
4	6.82	2.62
5	9.27	2.56
6	10.6	2.54
7	11.4	2.53
8	11.9	2.53

Model order	Single model	Smoothed models
0	4.64	4.64
1	3.38	3.38
2	3.32	3.05
3	3.88	2.78
4	5.68	2.61
5	7.70	2.52
6	9.23	2.49
7	10.2	2.48
8	10.5	2.47

Table 4.5: Cross-entropies for models predicting harmonic symbols from melody notes: major (above) and minor (below)

Model order	Single model	Smoothed models
0	4.31	4.31
1	3.36	2.60
2	7.34	2.30
3	11.3	2.25
4	13.5	2.24

Model order	Single model	Smoothed models
0	4.64	4.64
1	3.10	2.59
2	7.20	2.33
3	11.4	2.31
4	13.5	2.31

Table 4.6: Cross-entropies for models predicting harmonic symbols from melody notes and previous harmonic symbols: major (above) and minor (below)

with models of lower order greatly improves the predictive power of our Markov models. If we use smoothed models the cross-entropy reduces as we increase the model order, since higher order models are used only where they can make useful predictions. The tables show contexts up to eight symbols long; by that length the higher order models are only being used rarely, and the cross-entropy values only show very small improvements over the models restricted to using shorter contexts.

Table 4.5 shows that the melody notes appear to be more useful than the harmonic symbols as contextual information for predicting the next harmonic symbol, since the models which use them have lower cross-entropies. However, it is important to note that the final melody note in our context is the one being played on the beat whose harmonic symbol we want to predict. A context of harmonic symbols can only run up to the previous beat, since otherwise we would have no prediction left to: the final element of our context would be what we were trying to find. Since there are generally several possible ways of following from the preceding harmonies, the harmonic symbol based models can only guess what direction will be taken, whereas the models which use the melody notes have the possibilities narrowed down to those which are compatible with the final melody note in their context.

Even lower cross-entropies in table 4.6 show that the best model of the data is given by using both the melody notes and the preceding harmonic symbols. Even taking into account that, for example, the first order model here combines two symbols in its context, this model consistently performs best.

Chapter 5

Hidden Markov Models

This chapter describes Hidden Markov Models, and explains how we can apply them to chorale harmonisation. We will see that even a simple Hidden Markov Model benefits from its ability to 'plan': we can make predictions which take into account the probability of the entire sequence of which they are a part, and can find the globally most probable sequence.

5.1 Theory

This section introduces Hidden Markov Models, and relates some useful results. A useful longer treatment of Hidden Markov Models is provided by Rabiner (1989).

Instead of working directly from observed states, Hidden Markov Models assume that observed events occur because of underlying hidden states. The graphical form of a Hidden Markov Model is shown in figure 5.1.

An ordinary Markov assumption is made concerning the transition probabilities between the hidden states; here we will use a first-order model, such that we assume:

$$P(s_t = S_{q_t} | s_{t-1} = S_{q_{t-1}}, s_{t-2} = S_{q_{t-2}} \ldots s_0 = S_{q_0}) = P(s_t = S_{q_t} | s_{t-1} = S_{q_{t-1}}).$$

However, we now also need transition probabilities to model how the observed

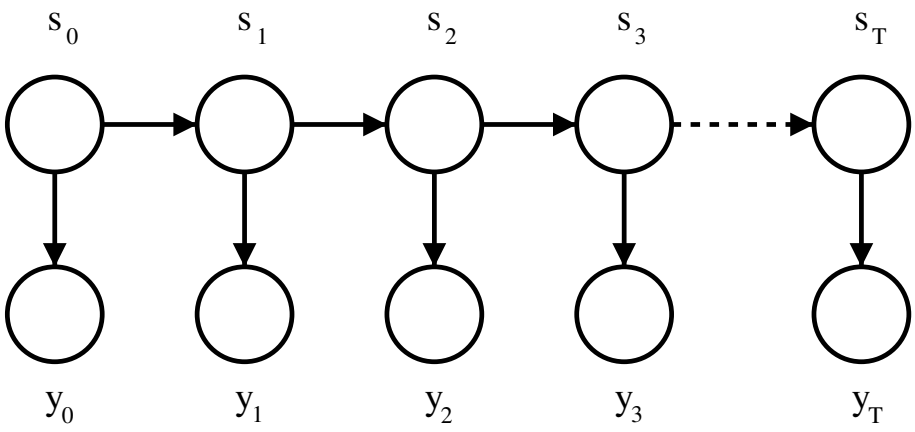

Figure 5.1: Hidden Markov Model

event results from the hidden state. We make a similar assumption that:

$$P(y_t = Y_{i_t} | s_t = S_{q_t}, \ldots s_0 = S_{q_0}, y_t = Y_{i_t}, \ldots y_0 = Y_{i_0}) = P(y_t = Y_{i_t} | s_t = S_{q_t}).$$

Thus the probability of a particular state and observed event given the preceding state is:

$$P(y_t = Y_k, s_t = S_j | s_{t-1} = S_i) = P(y_t = Y_k | s_t = S_j) P(s_t = S_j | s_{t-1} = S_i). \quad (5.1)$$

To find the probability of a particular sequence of observed events $y_0 = Y_{i_0}, y_1 = Y_{i_0}, \ldots y_T = Y_{i_T}$, we can sum over all possible state sequences. We define $\alpha_t(j)$ as the probability of seeing the first t observed events of the sequence and finishing in state j:

$$\alpha_t(i) = P(y_0 = Y_{i_0}, y_1 = Y_{i_1}, \ldots y_t = Y_{i_t}, s_t = S_j).$$

We can then use these variables to find the sequence probability by induction, using these forward probabilities:

$$\alpha_0(j) = P(s_0 = S_j) P(y_0 = Y_{i_0} | s_0 = S_j);$$

$$\alpha_t(j) = \left(\sum_k \alpha_{t-1}(k) P(s_t = S_j | s_{t-1} = S_k) \right) P(y_t = Y_{i_t} | s_t = S_j);$$
$$P(y_0 = Y_{i_0}, y_1 = Y_{i_0}, \ldots y_T = Y_{i_T}) = \sum_j \alpha_T(j). \quad (5.2)$$

5.2 Hidden Markov Models as generative systems

We would like to be able to use Hidden Markov Models to generate new harmonisations, given information about chorale melodies. This generative system can also be described in terms of classification: we wish to label each time step in the melody with an appropriate harmony. We can achieve this by treating the melody notes as observed symbols, emitted by underlying harmonies. Finding an appropriate harmonisation is therefore a question of finding an appropriate state sequence to explain the observed events. Two possible approaches are described below: maximum a posteriori (MAP) estimation using the Viterbi algorithm, and sampling from the conditional probability of the states given the outputs.

5.2.1 Viterbi algorithm

If we have a sequence of events which we are viewing as outputs from a Hidden Markov Model, how can we find the most likely state sequence? We do not just want to find the most likely state at each time step. Choosing the local maximum at an individual time step might bring us into a state where the rest of the sequence had a very low probability. Instead, we want to find the sequence which is globally most probable. Hidden Markov Models have the advantage that we can indeed easily find the globally most probable state sequence, using the Viterbi algorithm (Viterbi, 1967).

Given a particular sequence of observed events $y_0 = Y_{i_0}, y_1 = Y_{i_0}, \ldots y_T = Y_{i_T}$, we can define $\delta_t(j)$, the maximum probability of any individual state sequence which produces the first t observed events of the sequence and finishes in state j. We can then use induction to find the maximum probability of any individual state sequence which produces the entire observed sequence. So that we know which

state sequence it is which has this maximal probability, we instantiate variables $\psi_t(j)$ to record the states used along each partial maximum-probability path.

We find the maximum probability of any individual state sequence producing the observed events:

$$\begin{aligned}
&\delta_0(j) = P(s_0 = S_j)P(y_0 = Y_{i_0}|s_0 = S_j);\\
&\psi_0(j) = 0;\\
&\delta_t(j) = max_k \left(\delta_{t-1}(k)P(s_t = S_j|s_{t-1} = S_k)\right) P(y_t = Y_{i_t}|s_t = S_j);\\
&\psi_t(j) = argmax_k \left(\delta_{t-1}(k)P(s_t = S_j|s_{t-1} = S_k)\right);\\
&P^* = max_j\, \delta_T(j).
\end{aligned}$$

Then we work backwards to extract the state sequence in question, $s_0 = S_{q_0}, s_1 = S_{q_1}, \ldots s_T = S_{q_T}$.

$$\begin{aligned}
q_T^* &= argmax_j\, \delta_T(j);\\
q_t^* &= \psi_{t+1}(q_{t+1}^*).
\end{aligned}$$

5.2.2 Sampling

We can also generate random state sequences according to the probability distribution of our model. Using $\alpha_{t-1}(j)$, the probability of seeing the first $t-1$ observed events of a sequence and finishing in state j, we can calculate the probability of seeing the first $t-1$ events, finishing in any state, and then transitioning to state k at the next step:

$$\begin{aligned}
&P(y_0 = Y_{i_0}, y_1 = Y_{i_1}, \ldots y_{t-1} = Y_{i_{t-1}}, s_{t-1} = S_j, s_t = S_k)\\
&\qquad = \alpha_{t-1}(j)P(s_t = S_k|s_{t-1} = S_j).
\end{aligned}$$

We can use this to calculate $\rho_t(j|k)$, the probability that we are in state S_j at time $t-1$ given the observed event sequence $Y_{i_0}, Y_{i_1}, \ldots Y_{i_{t-1}}$, and given that we will be in state S_k at time t:

$$\begin{aligned}\rho_t(j|k) = P(s_{t-1} = S_j | y_0 = Y_{i_0}, y_1 = Y_{i_1}, \ldots y_{t-1} = Y_{i_{t-1}}, s_t = S_k) \\ = \frac{\alpha_{t-1}(j) P(s_t = S_k | s_{t-1} = S_j)}{\sum_l \alpha_{t-1}(l) P(s_t = S_k | s_{t-1} = S_l)}.\end{aligned}$$

To instantiate a state sequence $s_0 = S_{v_0}, s_1 = S_{v_1}, \ldots s_T = S_{v_T}$, we first choose the final state by sampling from its probability distribution according to our model:

$$P(s_T = S_j | y_0 = Y_{i_0}, y_1 = Y_{i_1}, \ldots y_T = Y_{i_T}) = \frac{\alpha_T(j)}{\sum_l \alpha_T(l)}.$$

Once we have chosen v_T such that the final state $s_T = S_{v_T}$, we can use the variables $\rho_t(j|k)$ to move back through the sequence:

$$P(s_t = S_j | y_0 = Y_{i_0}, y_1 = Y_{i_1}, \ldots y_T = Y_{i_T}, s_{t+1} = S_{v_{t+1}}) = \rho_{t+1}(j|v_{t+1}).$$

5.3 Application

In principle we would like to use the hidden states, transition probabilities and emission probabilities which give the best model of the data. Here we will train a Hidden Markov Model directly by taking data annotations as the hidden states and using maximum likelihood estimation to calculate the conditional probabilities we need.

For example, if we treat the harmonic symbols with which our data set is annotated as hidden states, we can find the best sequence of harmonic symbols according to our model by finding the Viterbi path. As well as being useful mathematically, it makes sense musically to think of harmonies as underlying a piece, and to think of the melody as being emitted at each time step from the underlying harmonic state.

To compare a Hidden Markov Model of the relationship between the melody and harmonic symbols with our earlier models, we want to calculate the cross entropy (negative average log likelihood per symbol). We need to calculate the probability of a test harmonisation $s_0 = S_{v_0}, s_1 = S_{v_1}, \ldots s_T = S_{v_T}$ according to our model, given a test melody $y_0 = Y_{i_0}, y_1 = Y_{i_1}, \ldots y_T = Y_{i_T}$.

$$P(s_0 = S_{v_0}, s_1 = S_{v_1}, \ldots s_T = S_{v_T} | y_0 = Y_{i_0}, y_1 = Y_{i_1}, \ldots y_T = Y_{i_T})$$
$$= \frac{P(s_0 = S_{v_0}, s_1 = S_{v_1}, \ldots s_T = S_{v_T}, y_0 = Y_{i_0}, y_1 = Y_{i_1}, \ldots y_T = Y_{i_T})}{P(y_0 = Y_{i_0}, y_1 = Y_{i_1}, \ldots y_T = Y_{i_T})}.$$

We can use equation 5.2 to find the probability of the event sequence according to our model, and we can use equation 5.1 iteratively to find the joint probability of a state sequence and event sequence. Since $\log \frac{a}{b} = \log a - \log b$, we can find the log conditional probability of the harmonisation given the melody by subtracting the log probability of the melody from the log joint probability. The cross-entropy is the total summed conditional probability divided by the number of observed symbols we have traversed.

I wrote a library of routines to train and use Hidden Markov Models, and used this to create these models from the chorale data training set. Functions from the ordinary Markov model library are used where possible. For example, the same function can be used for maximum likelihood estimation of transition probabilities with either model type.

5.4 Results and discussion

Using Hidden Markov Models with melody notes as the observed events and the harmonic symbol annotations from our data set as the hidden states, trained by maximum likelihood estimation from our training data, we obtain the cross-entropy values shown in table 5.1.

These cross-entropy figures are higher than our best results for the ordinary Markov models. This suggests that the smoothed high order Markov models provided a better model of the data. However, the real benefit of Hidden Markov

Cross-entropy (major)	Cross-entropy (minor)
2.76	2.58

Table 5.1: Cross-entropies for Hidden Markov Models treating the melody notes as observed events and the harmonic symbols as hidden states

Method	Cross-entropy (major)	Cross-entropy (minor)
Local maxima	1.20	1.41
Global maximum	0.84	0.87

Table 5.2: Cross-entropies for sequences of harmonic symbols generated by taking the locally most likely outcome at each step (local maxima), and by finding the globally most likely sequence using the Viterbi algorithm (global maximum)

Models comes because the form of the model allows us to take the whole sequence into account when making predictions. To demonstrate this we will generate sequences of harmonic symbols using two methods. The 'local maxima' method takes the locally most likely outcome at each step, which is equivalent to generation using a simple Markov model, and the 'global maximum' method finds the globally most likely sequence, using the Viterbi algorithm. We can compare cross-entropy values for the sequences generated by the two approaches to show the improvement in overall sequence probability which we can gain by using a Hidden Markov Model and the Viterbi algorithm as described above.

Table 5.2 shows the values we obtain for these two methods of sequence generation. It is clear that the Viterbi algorithm allows us to find significantly more probable sequences than we obtain by taking the local probability maximum for each decision. It is quite normal for the probabilities of our generated sequences to be greater than the probabilities of Bach's harmonisations under our model, as they are here. In fact, this is always the case for the Viterbi path: it is by definition the most probable sequence, so it must have a probability greater than or equal to the probability of the 'true' sequence from the data set.

It would be possible for it to turn out that our globally most likely sequences were boring, sticking too much to the most likely states where the sequences in the data set stray further afield. However, in almost all cases sequences generated by taking the local maximum at each step will be more guilty of this, since looking for the global maximum enables us to 'plan', and take a more unusual and therefore lower probability path now if it will raise the overall probability. Even if our global maximum sequences are too probable, it is still better to use a Hidden Markov Model than an ordinary Markov model, since we can use the sampling method described in section 5.2.2 to generate less likely sequences in an informed manner, still taking into account an entire state sequence rather than looking at single decisions independently.

Chapter 6

Final harmonisation model

This chapter describes a harmonisation model which takes into account the results of the previous chapters. We divide the task of harmonisation into three subtasks. Each subtask is discussed, and examples of chorale harmonisations generated by the overall model are given.

6.1 Building a harmonisation model

Previous harmonisation systems discussed in Chapter 2 suggest that we will be more successful if we divide the harmonisation task into multiple subtasks. Ideally we would conduct these subtasks in parallel, forming a 'product of experts' (Hinton, 1999) which efficiently categorises the space of possible chorale harmonisations. However, under certain conditions we can justify conducting the subtasks one after the other, feeding the results of one subtask into the next.

This kind of serialisation of subtasks will only succeed if no subtask before the final one precludes any valid harmonisations, and if no subtask produces results that will finally generate an invalid harmonisation. We need to be able to reach all valid harmonisations in the state space, and we need to make sure we do not produce results at one stage which are useless later. For example, if our first harmonisation subtask never generates certain harmonies, then these will never appear in the final harmonisations. Similarly, we need to ensure that our first

subtask generates output from which a valid harmonisation can later be created; not all sequences of harmonies will have allowable instantiations as notes. Of course, since we are working with probabilities rather than rules, the system will only judge different harmonisations as more or less probable rather than valid or invalid, but this does not affect the underlying issue.

6.2 The model

Following Hild et al. (1992), we will divide the task of harmonisation into three subtasks. First we will build a 'harmonic skeleton', labelling each beat with a harmonic symbol. Secondly we will build a 'chord skeleton' by filling in notes, aiming for them to fit with these harmonic symbols and form coherent lines of music in themselves. Third and finally comes 'ornamentation', as we fill in notes off the beat to improve each of the three additional lines of music we have added to harmonise the original melody.

6.2.1 Harmonic skeleton

We have already discussed possible ways of finding an optimal sequence of harmonic states for a given melody in Chapters 4 and 5. We will use the Hidden Markov Model created in Chapter 5 to solve the first subtask in our harmonisation system.

By using this model we treat the notes of the melody as an observation sequence 'emitted' by the hidden harmonic states. This makes sense in musical as well as mathematical terms: a harmonic symbol represents a set of possible chords, and we can view the melody note as one of many possible instantiations of the underlying harmonic state of the piece at that instant.

While some sequences of harmonic symbols would have no valid instantiation as chords, any prohibitions that we might list are short term, affecting only adjacent harmonies. Therefore we can justify making our decision on a sequence of harmonic symbols a separated subtask, since our model ought to take into account

this kind of prohibition in weighing up the probabilities of transitions between harmonic states.

6.2.2 Chord skeleton

This subtask requires a note to be decided at each beat of the melody for each of the three voices added in our harmonisations. We will use another Hidden Markov Model here. The harmonic symbols decided by the previous subtask will now be treated as an observation sequence, and we will generate chords as a sequence of hidden states. This model aims to 'recover' the fully filled-out chords for which the harmonic symbols are a shorthand.

To encode chords into hidden states for a Hidden Markov Model, we need to produce single symbols which represent the relationships between the four musical lines at a particular time step. We can do this by using an augmented harmonic symbol, which not only shows the harmonic category of a chord, but represents the chord as a set of intervals from a bass note. Figure 6.1 shows an example of this encoding. Given a sequence of these chord symbols, and a chorale melody line, we can unambiguously construct a sequence of full chords.

We can separate out producing a sequence of chords as a subtask since from any valid sequence of harmonies we will be able to build a sequence of chords that will be valid input for the ornamentation stage. In fact, because we are using a Hidden Markov Model, this subtask might be able to work around problems in its input: the model has the capability to ignore input for which it can find no suitable chords, if the overall probability of the state sequence given the input sequence is increased by doing this.

6.2.3 Ornamentation

This subtask allows additional notes to be added into the three musical lines we have created alongside the melody. For example, we may want to add in faster notes to lessen any large jumps in pitch. This 'ornamentation' was created separately for each line of music.

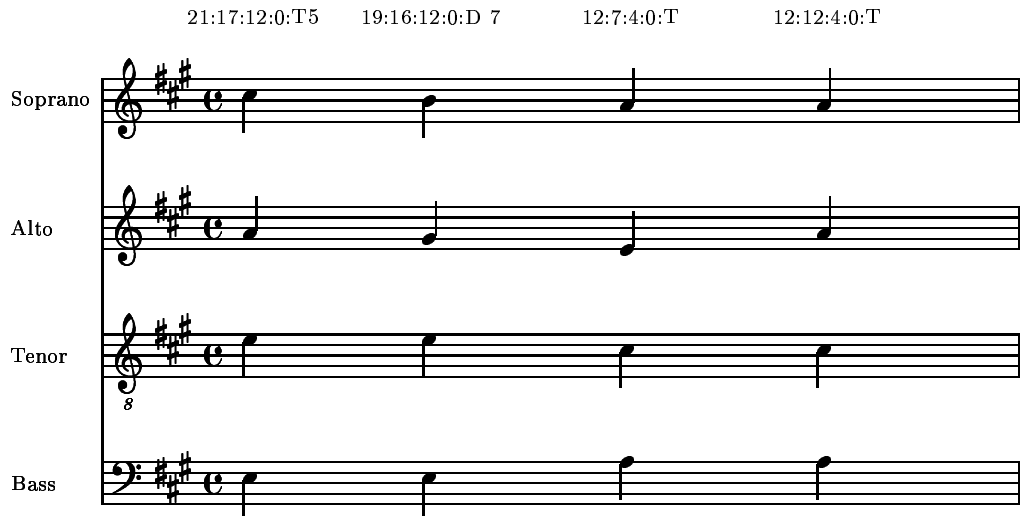

Figure 6.1: Example of chord encoding

The data format, as described in Chapter 3, resolves all rhythms onto sixteenth-note steps, with four of these steps making up each beat. 'Ornamentation' was encoded by listing the interval from the first of these notes to the note played on each of the four steps. This representation means that any transposition of a musical phrase is encoded in the same way. A compound symbol was used as the observed state, made up of the notes on the current and next beat, and the current harmonic symbol. The next note was included since the shorter notes that this subtask is intended to add fill out the movement between the two notes that frame them.

While it could be argued that the faster notes which we add in ornamentation should be created along with the main notes in a line of music, treating all ornamentations of a chord as different would greatly increase the size of the model required. Not only would this slow down the calculation of the most likely state sequence exponentially, but it would make our data more sparse in proportion. Similarly it would be preferable for our model to take into account the interactions between ornamentation in different voices, but the Hidden Markov Model we are using does not cope well with the sparse probability distribution we would

Subtask	Cross-entropy (major)	Cross-entropy (minor)
Harmonic skeleton	2.80	2.79
Chord skeleton	13.3	12.9
Ornamentation	16.0	16.9
alto	5.57	5.66
tenor	6.20	6.16
bass	4.24	5.07
Total (all subtasks)	32.1	32.6

Table 6.1: Cross-entropies on test set 3, a held-out set, for each subtask

then have to work with. The problems that might be caused by treating each voice independently are lessened because we include the current harmonic symbol in our model.

6.3 Results and discussion

Table 6.1 shows the cross-entropy for each subtask in the final model, measured using a held-out set of data. Since the 'ornamentation' subtask used a separate model for each line of music, the separate cross-entropy values are shown as well as the overall value. The total cross-entropy, with all three subtasks working together, is also shown. Comparing the cross-entropy values, we can see that the ornamentation is least predictable, then the chord skeleton, then the harmonic skeleton. There are many possible solutions for ornamentation, so we would not expect to be able to predict the actual ornamentation with great accuracy. Our model has only taken into account a few of the features which lead to the selection of a particular chord or ornamentation. We can see that by separating out the generation of a harmonic skeleton we have given ourselves a relatively predictable way of creating the framework for our harmonisations.

Figures 6.2 and 6.3 show typical output of the first subtask, which constructs a 'harmonic skeleton'. We can see, for example, that the final notes of both melodies

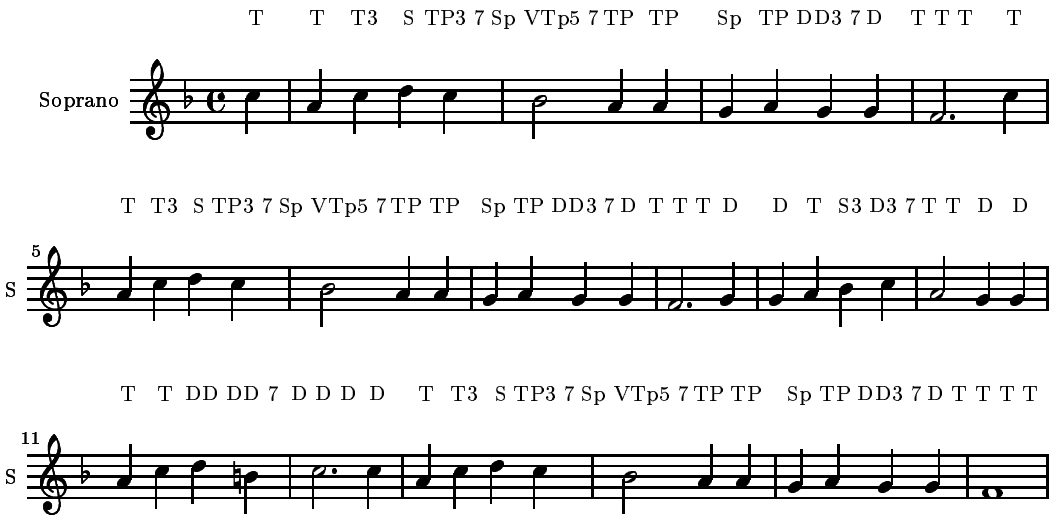

Figure 6.2: Example harmonic skeleton: 'Dank sei Gott in der Höhe', melody of chorale K54, BWV 287

are all assigned the harmonic symbol 'T', showing that these have been labelled as tonic chords, which will make the ends of the harmonisations sound complete. 6.4 shows a less successful harmonic skeleton: here the harmonies appear to be stuck in some unusual states. If several harmonic symbols are equally likely given the melody note, the Hidden Markov Model will pick the state sequence which contains the highest probability transitions. This behaviour usually works well, but here high-probability transitions between unusual states have led to a long sequence of such states being produced. To reduce this problem we could make our decisions conditional on additional features. For example, if we used a higher order Hidden Markov Model then the longer context would allow the model to learn if rare harmonic symbols come as the beginning of a sequence leading back to more frequent harmonies. The first-order model we are using here only learns the immediate consequences of events.

Figure 6.5 shows example output from the second subtask, which constructs a 'chord skeleton', for the same melody whose harmonic skeleton was shown in

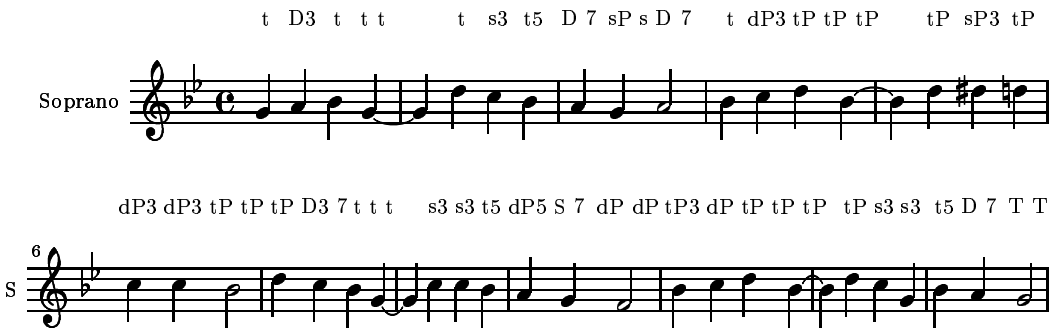

Figure 6.3: Example harmonic skeleton: ‘Schaut, ihr Sünder! Ihr macht mir große Pein’, melody of K303, BWV 408

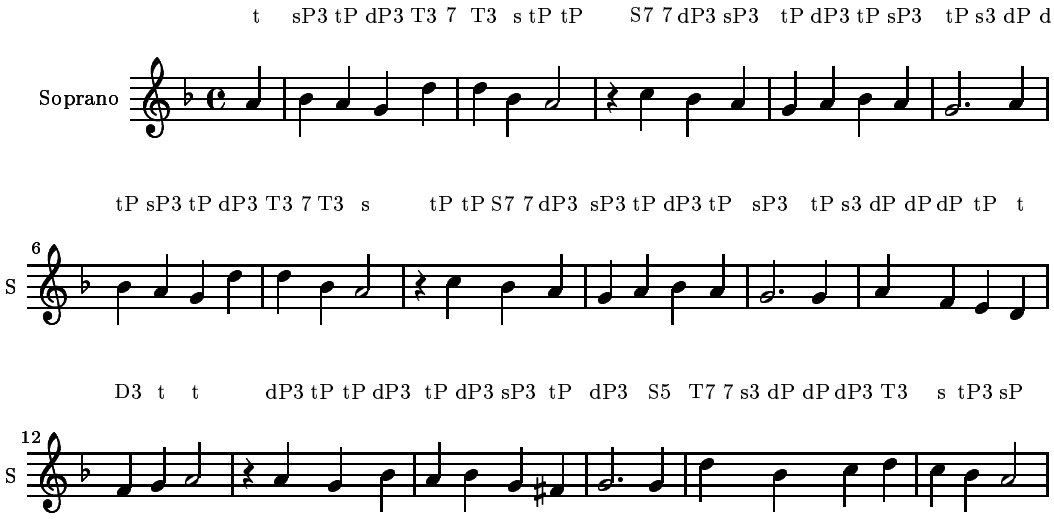

Figure 6.4: Example harmonic skeleton showing problems: ‘Ach Gott, vom Himmel sieh’ darein’, melody of K6, BWV 77.6

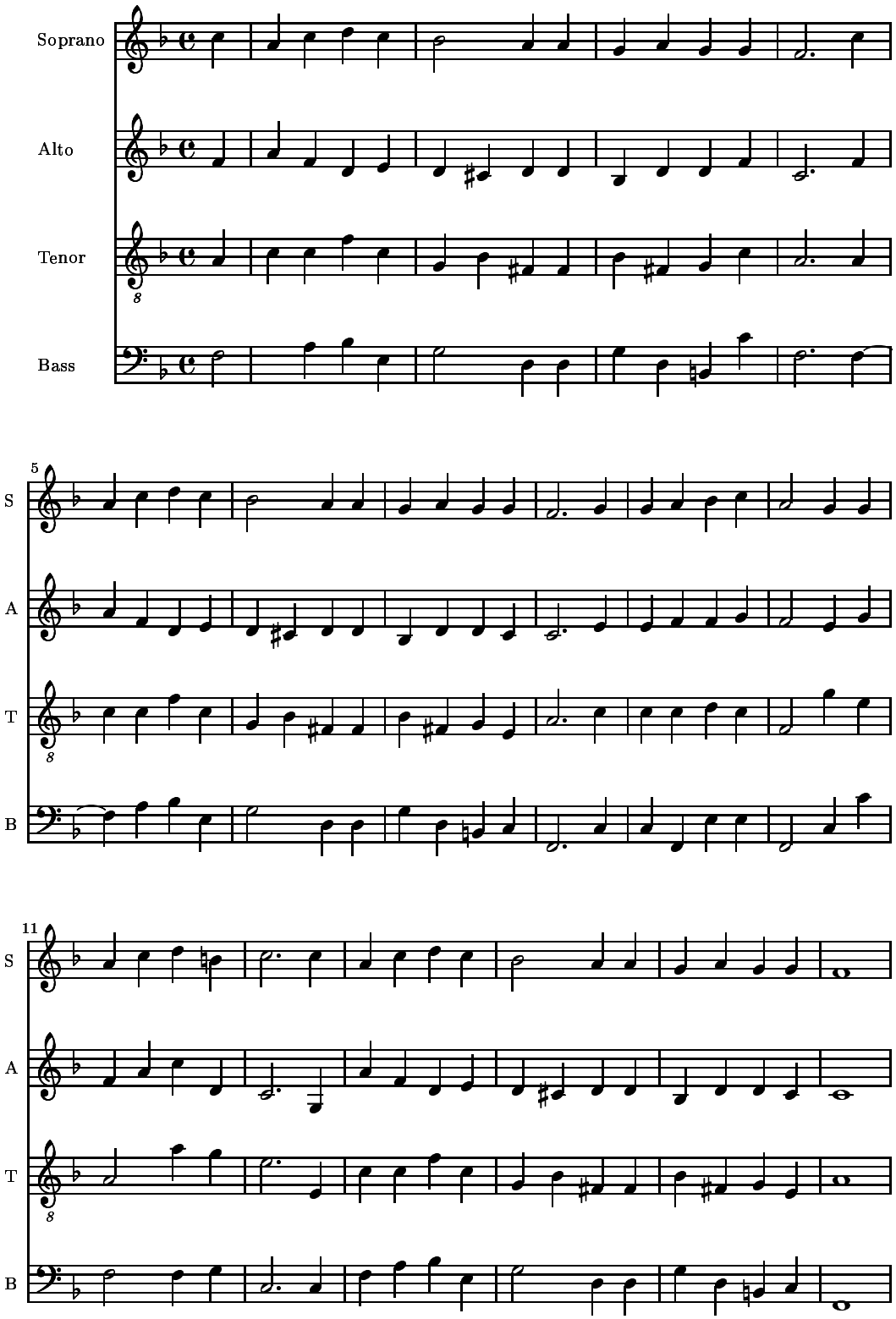

Figure 6.5: Example chord skeleton: harmonisation of ‘Dank sei Gott in der Höhe’, melody of chorale K54, BWV 287

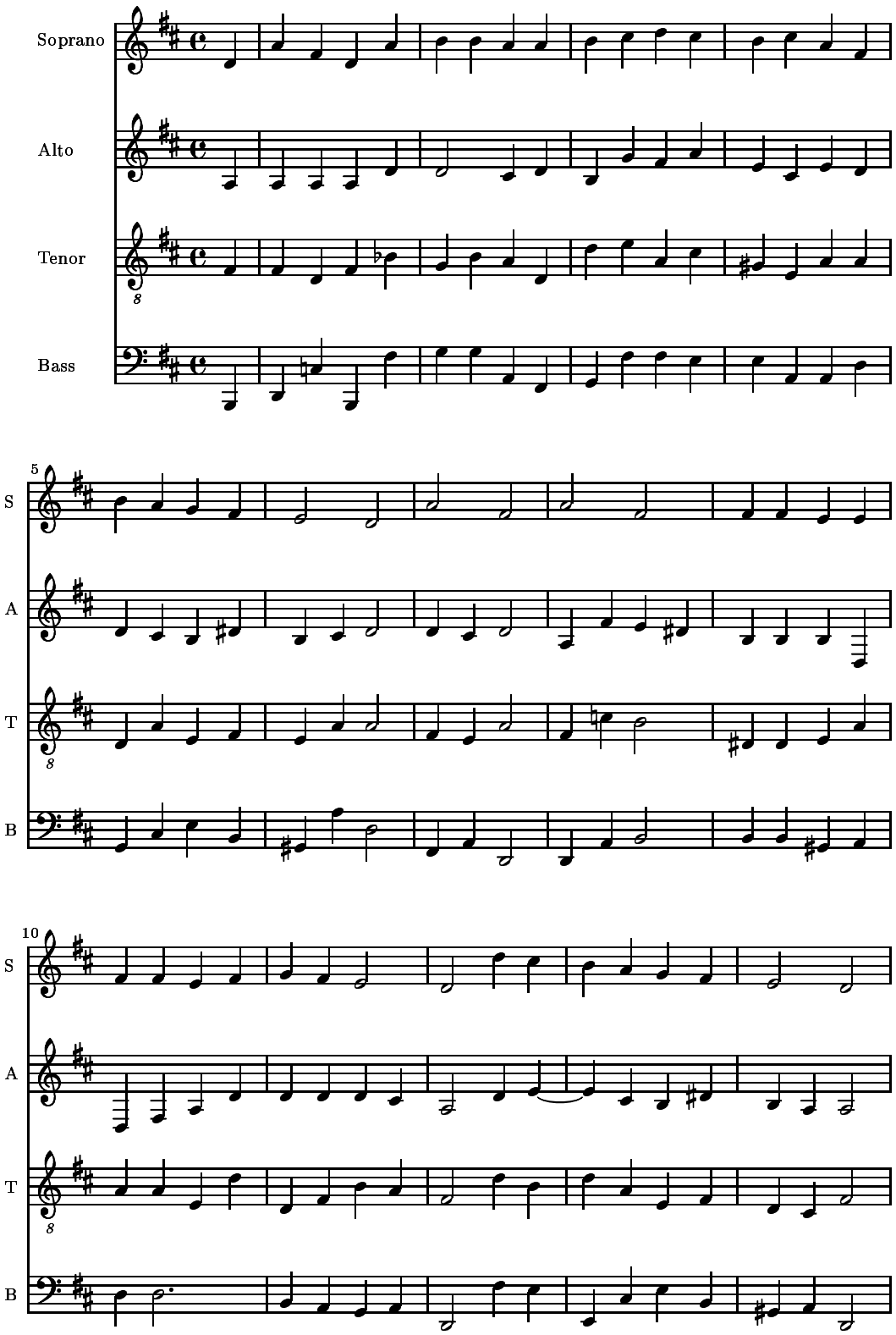

Figure 6.6: Example chord skeleton: harmonisation of ‘Wie schön leuchtet der Morgenstern’, melody of chorale K377, BWV 36.(2).4

figure 6.2. Although the model selects whole chords from its training data to instantiate the harmonic symbols decided by the first subtask, we can see that reasonable lines of music have been generated: the notes tend not to jump by excessive intervals, but to move in acceptable patterns. Since the model works with whole chords, the lines of music do not cross over or move too far apart. Figure 6.6 gives another example, in which we can see problems with the lines of music straying outside their expected ranges. In the first bar both the alto and bass lines are unexpectedly low. This can happen because the encoding used in our model gives no opportunity for voice ranges to be learnt, since chords are considered only as transposable sets of intervals. To address this problem we would need to introduce some way for the probability of a chord to be judged according to the absolute pitches of its notes, not only the transposed intervals.

Figure 6.7 shows example output from the final 'ornamentation' subtask, for the same melody whose harmonic skeleton was shown in figure 6.5 and whose chord skeleton was shown in figure 6.2. We can see that the combination of the three subtasks has resulted in a reasonable harmonisation. The added ornamentation makes the lines of music flow better, and makes the passage between different harmonies more graceful. Figure 6.8 shows an example where we can see an unfortunate jarring combination of ornamentation, in bar 8 (the third bar on the second line). Our model makes no attempt to prevent unpleasant combinations of ornamentation being placed vertically together, on the same beat, hoping that bad vertical combinations will be heard as allowable passing dissonances. However, considering the simple way we model ornamentation, the results seem surprisingly good.

Figures 6.9, 6.10, and 6.11 show some example harmonisations generated by the system for melodies from the set of held-out data. Possible enhancements to the system, and other suggestions for future work, are discussed in Chapter 7.

6.4 Example audio files

Example audio files, as well as the source code of the programs used, can be downloaded from http://www.srcf.ucam.org/~mma29/2002/harmony/.

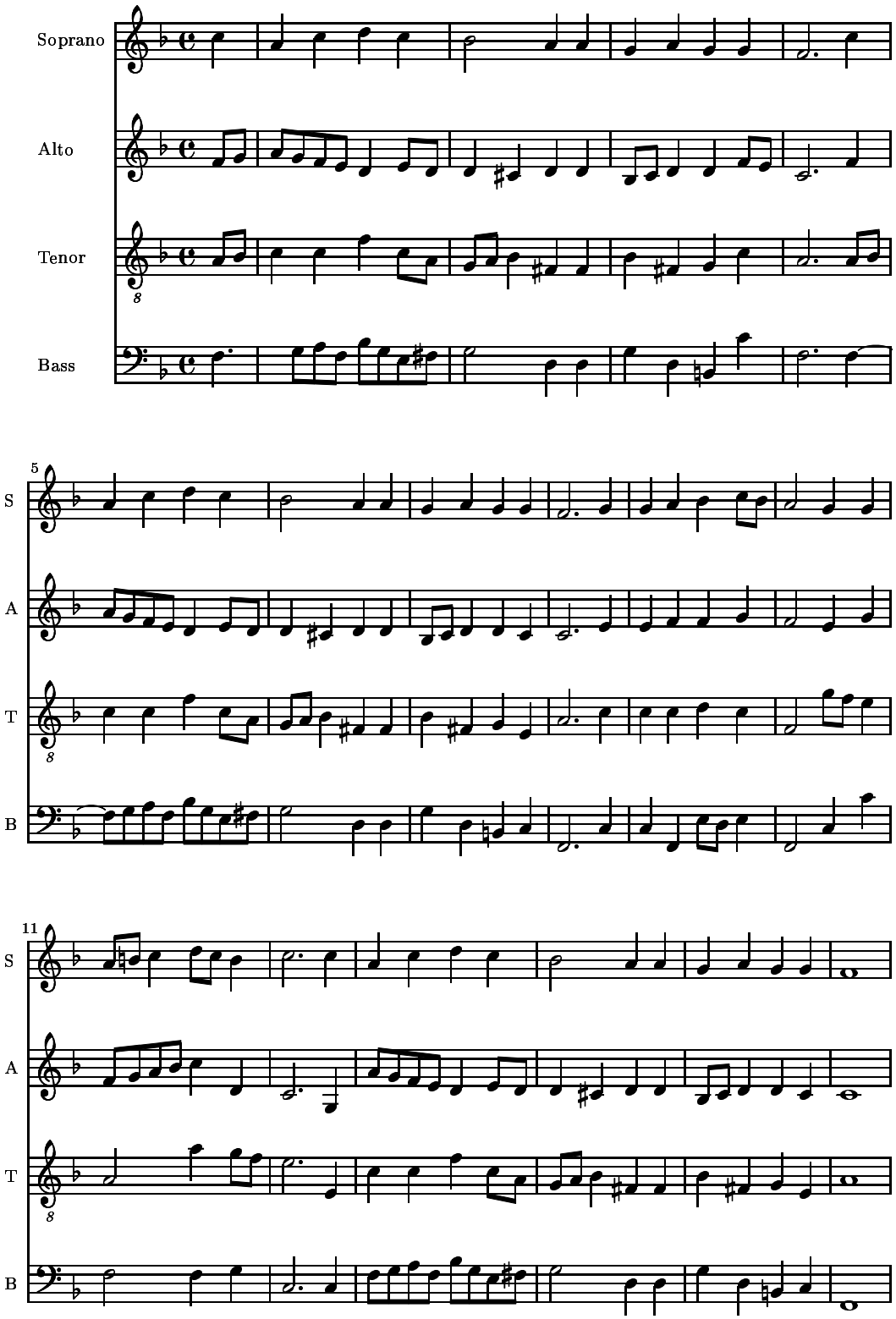

Figure 6.7: Example illustrating ornamentation: harmonisation of ‘Dank sei Gott in der Höhe’, melody of chorale K54, BWV 287

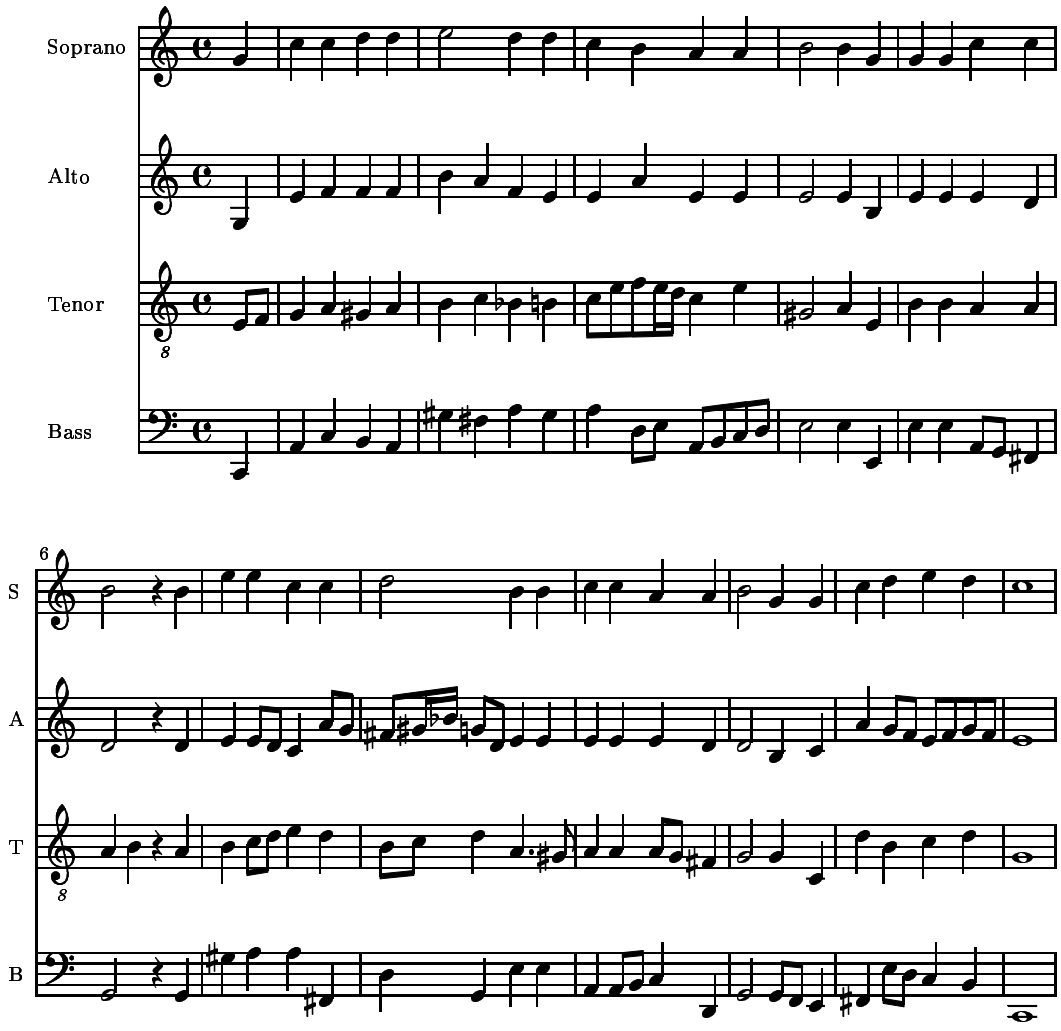

Figure 6.8: Example illustrating ornamentation: harmonisation of ‘In allen meinen Taten’, melody of chorale K211, BWV 367

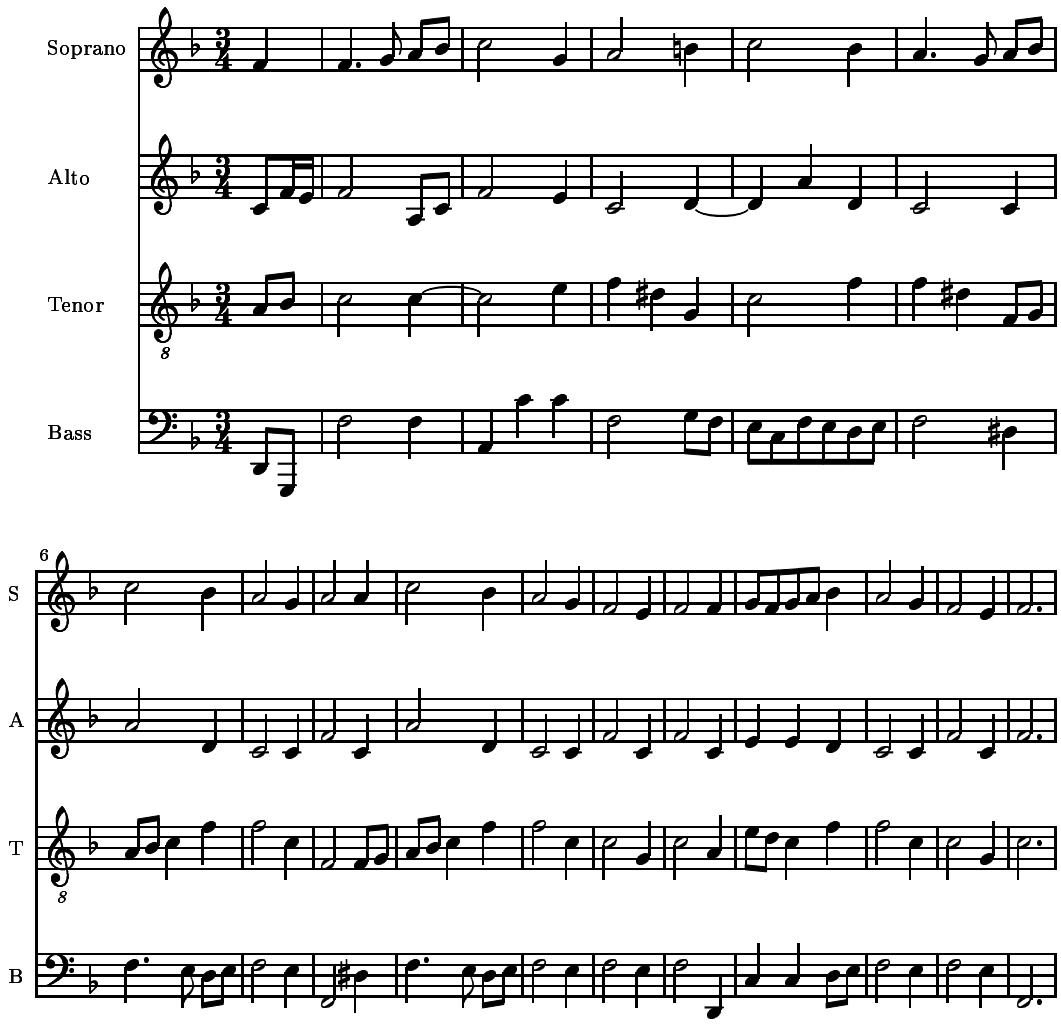

Figure 6.9: Example harmonisation: ‘Erstanden ist der heilige Christ’, melody of chorale K85, BWV 306

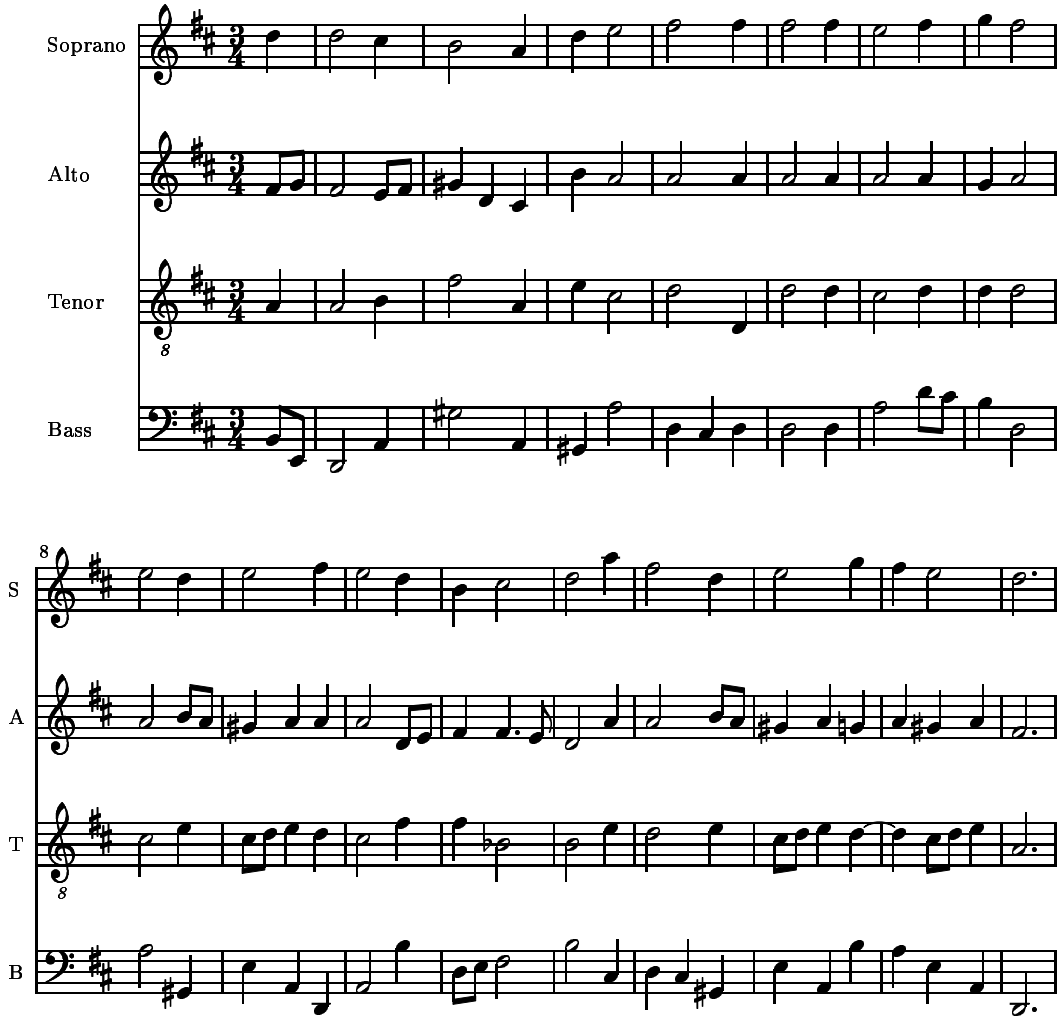

Figure 6.10: Example harmonisation: ‘Für deinen Thron tret’ ich hiermit’, melody of chorale K132, BWV 327

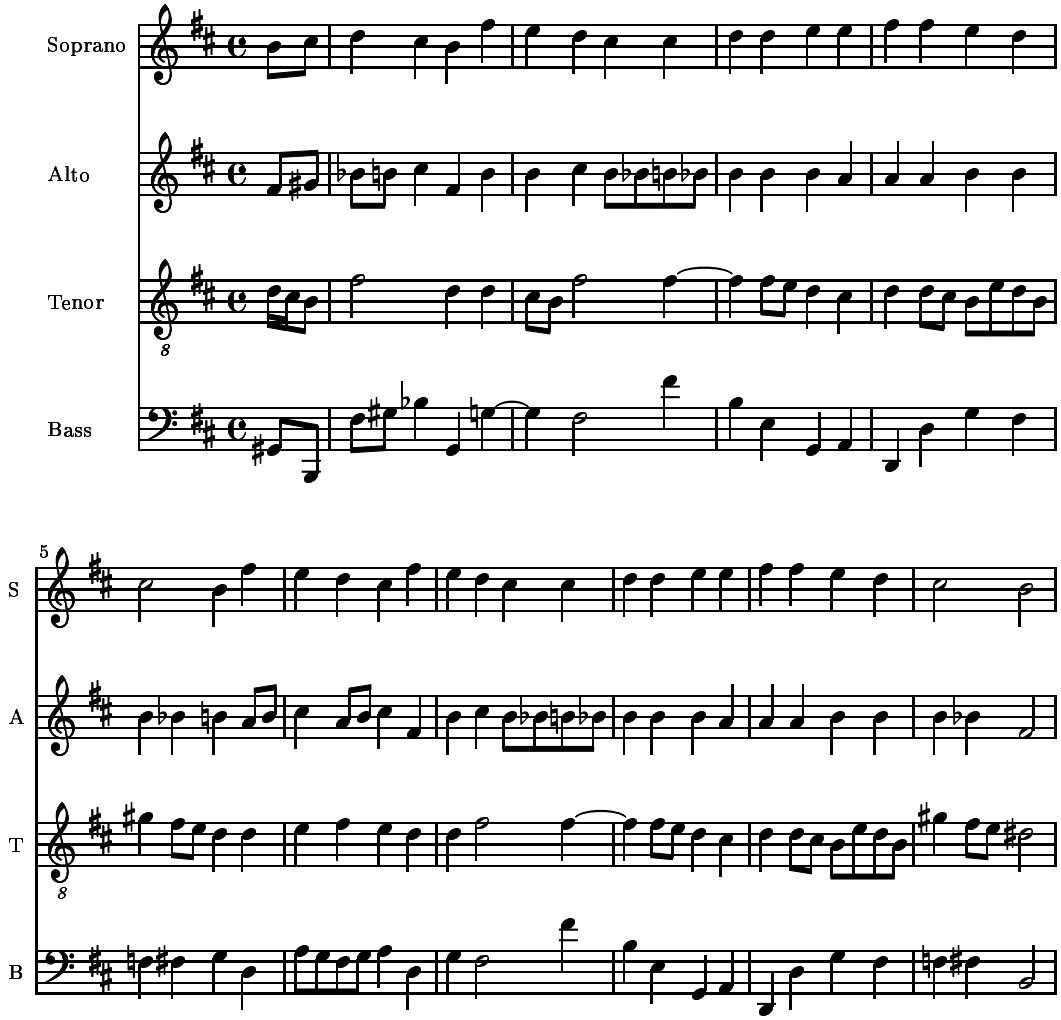

Figure 6.11: Example harmonisation: ‘Wir Christenleut’, melody of chorale K380, BWV 110.7

Chapter 7

Conclusions and future work

This chapter asks what conclusions we can make from the preceding material, and gives suggestions for future work, including possible enhancements to the model described in Chapter 6.

7.1 Conclusions

In this dissertation we aimed to create a harmonisation system which learns its harmonic rules by example, rather than using a manually-programmed knowledge base; which is simple, making the assumption that we only need to model short-term dependencies in the local context; and which can find globally probable harmonisations, rather than 'greedily' choosing the local maximum at each decision.

The results in Chapter 6 show that these aims were coherent and successful. While there are many ways in which the system described there could be improved, it demonstrates that a simple Hidden Markov Model system is capable of learning how to harmonise chorale melodies. The Hidden Markov Model allows us to find the global probability maximum, and we can also sample from the probability distribution at each step while taking into account the whole sequence of transitions and emissions leading up to that point. This offers a significant improvement over simple Markov models, which only use the local maximum

according to the local context, and allows us to keep our model relatively small, since the short-term dependencies work together to provide a long-term probability distribution.

Perhaps most significantly, the system described in Chapter 6 shows that a pre-programmed knowledge base is not necessary for chorale harmonisation. The system was given some supervision, since the problem was divided into three subtasks, but the models were not given any harmonic knowledge beyond that contained in their training data.

Our system is able to 'plan' like the constraint-based systems described in Chapter 2, but here knowledge acquisition has been integrated into the overall task. The system learns its 'rules' by statistical observation of example harmonisations, like previous sequence prediction systems, but the Hidden Markov Model uses the probabilities it learns in a more structured fashion, taking into account whole sequences rather than making independent decisions on individual symbols.

7.2 Future work

Two simple enhancements to the system described in Chapter 6 would almost certainly increase its performance without too great a computational cost. First, we could make our decisions conditional on the position in the bar of music. Music theory recognises a hierarchy of stressed beats within the bar, and harmonic movement should correlate with these stresses. Secondly, we could take into account the probability of a note given a model of the distribution of pitches for each voice. Since our system only considers the intervals within chords, it does not take into account the natural range of each voice, and in this way we could introduce the concept of voice range as a statistical distribution rather than enforcing it by adding pre-decided 'knowledge' to the system. This enhancement would also lead to a more general improvement in the harmonisations produced, since by preventing the bass note from descending too far it would keep the musical lines closer

together, which is itself more pleasing, and leads to more interesting harmonies as the chords seek solutions to the greater constraints on their movements.

The simple sequence prediction models described in Chapter 4 could be enhanced in various ways. More complex methods of smoothing different orders of model would be likely to improve their performance. 'Probabilistic Suffix Automata' might offer an alternative solution (Ron et al., 1994). Probabilistic Suffix Automata have a variable memory length dependent on the context: a tree of conditioning events is grown during training. In music they could allow a model, for example, to learn to use a longer context to deal with cadences.

We saw that using harmonic symbols and melody notes together gave the best model. Using a model which was adapted to deal with several sources of information might increase performance further. We might for example use a multiple viewpoint system like Conklin and Witten (1995). Alternatively, a logarithmic opinion pool (Hinton, 1999) would allow us to efficiently model a high-dimensional space: with such a 'product of experts' we might, for example, factorise the probability of a chord given the previous chord as the product of separately modelled probabilities of the individual notes following the notes in the previous chord, and of the intervals in the chord following the intervals in the previous chord, or other such relationships. Ghahramani and Jordan (1996) and Saul and Jordan (1998) show how we can use mixture distributions to create factorial Hidden Markov Models which provide an efficient representation of large state spaces. However, such a model would nevertheless have a much higher time complexity than the models used in this dissertation. If we were ready to accept a larger model, we could directly extend our system to a higher order, using a longer context.

Another approach that would allow us to make use of additional features for harmonisation would be to use a log-linear model. McCallum et al. (2000) proposed Maximum Entropy Markov Models, similar to Hidden Markov Models but using log-linear probability distributions. Lafferty et al. (2001) note that Maximum Entropy Markov Models suffer from 'label bias', preferring hidden states which can only be followed by a few other states; they describe Conditional Ran-

dom Fields, which do not suffer from this problem. Their results show that Maximum Entropy Markov Models can perform worse than standard Hidden Markov Models, while on the same data the Conditional Random Field performs a little better. Conditional Random Fields could allow us to create a better model of chorale harmonisation, taking account of more dependencies in our data, but it is not yet clear how they can be efficiently trained. Even when they used their optimal Maximum Entropy Markov Model parameters as an initialisation for a Conditional Random Field model, the remaining Conditional Random Field training took ten times more iterations than the Maximum Entropy Markov Model training had taken in total.

All the models we have considered aim to generate reasonable harmonisations of the chorale melodies, without any further constraints of form. Bach's own harmonisations, however, reflect more specific intentions: for example, some are intentionally complex, and others intentionally simple. If we could quantify this kind of intent, then we could train aspects of our harmonisation system on more specific groups of examples. As long as we ensured that enough data was available to train each aspect of the system, the end results should be improved by using more specific and therefore more homogeneous training data in this way. As an alternative to labelling the chorales according to types, it would be possible to use a text categorisation approach (Yang and Liu, 1999) and automatically label the chorales according to their similarities and differences. A still more sophisticated harmonisation system would take into account the words of the chorales as it decided upon their harmonic movements, since Bach's own harmonisations are affected by the properties of the texts as well as of the melodies.

Appendix

Allocation of chorales to data sets

Training set (major):

205 177 257 281 117 266 375 258 351 237 301 362 292 361 313 19 159 69 110 245 218 75 234 285 78 130 64 304 219 128 356 119 104 158 360 89 86 275 364 239 97 338 279 294 286 363 3 357 162 335 293 270 314 242 276 236 290 282 269 67 88 109 76 378 306 359 23 118 353 339 268 238 352 247 241 77 92 212 246 233

Test set 1 (major):

4 98 291 13 150 254 134 179 103 365 334 220 215 366 146 123 261 296 80 272 99 51 277 12 298 61 90 139 175 154 47 193 135 8 127 315 17 100 228 163 108

Test set 2 (major):

309 155 54 211 30 24 243 46 101 74 235 107 131 189 136 147 20 82 312 273 181 227 208 389 271 115 240 221 244 377 259 176 203 1 267 263 256 284 262 329

Test set 3 (major):

91 354 87 230 209 153 376 341 114 151 58 85 178 278 152 62 68 14 355 132 340 255 66 323 60 113 358 192 231 280 214 289 204 9 283 129 202 295 194 102 81

Training set (minor):

106 18 141 160 217 206 53 311 388 37 195 39 125 171 333 156 33 373 232 10 96 5 31 63 65 223 342 308 48 249 45 253 229 83 22 387 372 200 174 274 251 93 196 32 382 332 168 310 34 172 343 287 164 49 248 328 148 379 70 2 145 226 305 161 28 327 142 57 381 41 222 190

Test set 1 (minor):

173 224 26 140 260 316 16 126 300 250 144 124 73 137 71 79 347 186 344 21 198 349 11 384 207 197 40 52 84 318 302 185 337 322 38 165

Test set 2 (minor):

303 187 6 15 182 350 325 169 252 143 42 43 265 348 371 25 367 170 330 183 345 55 370 324 35 383 331 116 7 374 368 167 122 94 111 120

Test set 3 (minor):

29 138 50 386 380 288 210 27 36 317 225 216 166 385 320 264 188 112 201 319 326 59 56 95 346 72 149 105 307 157 321 369 184 180 299 44

Bibliography

Bach, J. S. (1998). Chorale harmonisations, in a computer-readable edition. ftp://i11ftp.ira.uka.de/pub/neuro/dominik/midifiles/bach.zip.

Burke, S. M. (2000). MIDI::Simple. Available from the Comprehensive Perl Archive Network, http://www.perl.com/CPAN/.

Chen, S. F. and Goodman, J. (1998). An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University.

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

Fux, J. J. (1725). *Gradus ad Parnassum*. Vienna.

Ghahramani, Z. and Jordan, M. I. (1996). Factorial Hidden Markov Models. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems*, volume 8, pages 472–478. The MIT Press.

Hild, H., Feulner, J., and Menzel, W. (1992). HARMONET: A neural net for harmonizing chorales in the style of J.S. Bach. In Lippman, R., Moody, J., and Touretzky, D., editors, *Advances in Neural Information Processing 4 (NIPS 4)*, pages 267–274. Morgan Kaufmann.

Hinton, G. E. (1999). Products of Experts. In *Proceedings of the Ninth International Conference on Artificial Neural Networks (Proc ICANN 99)*, volume 1, pages 1–6.

Holland, J. H. (1975). *Adaption in Natural and Artificial Systems*. University of Michigan Press.

Katz, S. M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recogniser. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-35:400–401.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional Random Fields: probabilistic models for segmenting and labeling sequence data. In *Machine Learning: Proceedings of the Eighteenth International Conference on Machine Learning (ICML-2001)*, pages 282–289.

Lidstone, G. J. (1920). Note on the general case of the Bayes-Laplace formula for inductive or *a priori* probabilities. *Transactions of the Faculty of Actuaries*, 8:182–192.

Löthe, M. (2000). Knowledge-based composition of classical minuets by a computer. In *Proceedings of the AISB Symposium on Artificial Intelligence and Cultural Creativity, Birmingham*.

Markov, A. A. (1913). An example of statistical investigation in the text of 'Eugene Onyegin' illustrating coupling of 'tests' in chains. *Proceedings of the Academy of Sciences, St. Petersburg*, 7(VI):153–162.

McCallum, A., Freitag, D., and Pereira, F. (2000). Maximum Entropy Markov Models for information extraction and segmentation. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML-2000)*, pages 591–598, Stanford, California.

McIntyre, R. A. (1994). Bach in a box: The evolution of four-part baroque harmony using the genetic algorithm. In *Proceedings of the IEEE Conference on Evolutionary Computation*.

Nivre, J. (2000). Sparse data and smoothing in statistical part-of-speech tagging. *Journal of Quantitative Linguistics*, 7(1):1–17.

Pachet, F. and Roy, P. (1995). Mixing constraints and objects: a case study in automatic harmonization. In *TOOLS Europe '95*, pages 119–126. Prentice-Hall.

Pachet, F. and Roy, P. (2001). Musical harmonization with constraints: A survey. *Constraints*, 6(1):7–19.

Phon-Amnuaisuk, S. and Wiggins, G. A. (1999). The four-part harmonisation problem: a comparison between genetic algorithms and a rule-based system. In *Proceedings of the AISB'99 Symposium on Musical Creativity*.

Ponsford, D., Wiggins, G., and Mellish, C. (1999). Statistical learning of harmonic movement. *Journal of New Music Research*.

Rabiner, L. R. (1989). A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285.

Rameau, J.-P. (1722). *Traité de l'Harmonie reduite à ses principes naturels*. Paris.

Riemenschneider, A. (1941). *371 Harmonized Chorales and 69 Chorale Melodies with Figured Bass*. G. Schirmer, Inc.

Ron, D., Singer, Y., and Tishby, N. (1994). The power of amnesia. In *Advances in Neural Information Processing Systems*, volume 6, pages 176–183. Morgan Kaufmann.

Saul, L. K. and Jordan, M. I. (1998). Mixed memory Markov models: decomposing complex stochastic processes as mixtures of simpler ones. *Machine Learning*, 37:75–87.

Schottstaedt, B. (1989). Automatic species counterpoint. Technical report, Stanford University CCRMA.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656.

Towsey, M., Brown, A., Wright, S., and Diederich, J. (2001). Towards melodic extension using genetic algorithms. *Educational Technology and Society*, 4(2).

Tsang, C. P. and Aitken, M. (1991). Harmonizing music as a discipline of constraint logic programming. In *Proc. ICMC 1991*, pages 61–64, Montreal.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–267.

Yang, Y. and Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '99)*, pages 42–49.