Automatic Harmonic Analysis of Melodies

Finlay McAfee

MInf Project (Part 2) Report

Master of Informatics
School of Informatics
University of Edinburgh

2017

Abstract

Acknowledgements

I would like to thank Mark Steedman and Andrew McLeod for their advice and guidance on this project.

Declaration

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

(*Finlay McAfee*)

Table of Contents

Chapter 1

Introduction

This is a report to detail the work carried out in the second year of the MInf Project.

1.1 Summary of contributions

1.2 Previous Year

Chapter 2

Theoretical Background

This chapter goes through the main concepts necessary to understand the design of the systems built in this project. Each section is intended to provide a brief summary of the necessary information on each topic to allow the reader to fully understand the workings of the models in the next chapter, as well as an understanding of the design choices.

2.1 Music Theory

It is safely assumed that the reader is familiar with the concept of music. If not then it is suggested that they study the following albums before returning to read this report: Blood On The Tracks by Bob Dylan [8] and Songs From a Room by Leonard Cohen [5].

Music is fundamentally concerned with two concepts: rhythm and pitch. Rhythm is, at its simplest, a regularly repeating pattern. The periods of repetition that musical rhythm is most concerned with lie around the same frequency as that of the human gait, or heart beat (the reader is left to draw their own conclusions about the significance of this). Pitch is, essentially, also concerned with regularly repeating patterns, only now the domain is hundreds of beats per second. The human ear experiences this as a musical note, rising and falling in pitch as the frequencies increase and decrease.

Music has been studied and practised by many different cultures over the course of human history and there have been many different formalisms for describing it developed. The most common, and that used in this report is *Western Tonal Music* (WTM). Under this paradigm, the continuous range of pitch is divided into repeating *octaves*, where corresponding points on each *octave* are perceived as the same note at different pitches. These ranges are then subdivided into 12 distinct notes. The method used for these divisions is based on ratios between note frequencies, the most commonly used being Equal Temperament [35]. Pieces of music are organised into one of 12 *keys*, each with a *tonic* corresponding to one of the 12 notes. Keys can be thought of as a prior over the distribution of notes in a song, where the tonic note has the highest probability

of being played.

In written music, pitch is associated with vertical direction on a *stave*, shown below.

DIAGRAM.

Rhythm is concerned with the temporal positioning of these notes, both where and when they occur, as well as for how long to play them. In WTM a piece of music has an associated beat or *tempo* that defines the frequency of an underlying, unheard pulse, over which the notes of the piece are laid, typically close to 120 beats per minute. The *time signature* of a piece defines how to treat groups of these beats, with the most common being $\frac{4}{4}$. This corresponds to groupings of four beats (also called *quarter notes* or *crotchets*). These groupings are referred to as *bars* and are used to organise the temporal structure of a melody in a way that is both easy to read and represents the underlying rhythmic properties of the piece. As in western languages, time flows from left to right in written music.

DIAGRAM

The above is a very brief overview of melodic structure in a piece of music. However there is another concept that is fundamental to this problem space, and that is *harmony*. Fundamentally, *harmony* is the relationship between the pitches of two notes, as they are heard together. It is concerned with the *intervals* between the notes and how they are perceived. The most basic interval is the *semitone*, the distance between one note and its neighbour, hence there are 12 semitones in an octave. The harmonic perception of an interval of two or less semitones is dissonant and jarring. The octave itself is an interval, which is perceived as resonant and uncharacterised, another way of saying that two pitches an octave apart are perceived as the same note. The next simplest interval is when two notes are 5 semitones apart, the *perfect fifth*, referred to as a *power chord* in guitar terminology, it adds the most basic harmonic colour. The intervals of 3 and 4 semitones are where harmony takes on an emotional quality, 3 semitones is referred to as a *minor third*, and evokes a feeling of darkness and melancholy, when perceived; whereas 4 semitones, the *major third* evokes the opposite emotion, that of brightness. It is this incredible connection between emotions and intervals that makes musical harmony so fascinating.

INTERVAL DIAGRAM

A *chord* is when more than one note is played at one point in time, typically at least three. The basic three-note chord is called a *triad*, composed of notes that are based on intervals, relative to the note in the first position, the *root*. The other two are the *perfect fifth* and a *third*, either *major* or *minor*. Depending on the *third* that is chosen, the full chord with take on one of these qualities. The following diagram shows the basic C major chord.

DIAGRAM OF C CHORD

A modern musical arrangement, for example a jazz standard, typically consists of both a melody and an associated progression of chords, intended to be played as an accompaniment to the melody. The problem space of this project is concerned with the association between these two, specifically the problem of generating one if the other is

not present. Generating a melody is what is typically considered musical composition, a difficult problem to solve on which much work has been done in recent years CITE ALL THE THINGS. This project concerns itself with the other direction, predicting chord sequences from melodies. This problem is conceptually half sequential classification, half generation, as there is not necessarily a unique chord sequence for every melody, but there is still a ground truth that can, theoretically, be derived from the observed notes. The following piece of music is an example from one of the corpora used for training the models.

DIAGRAM MINUET IN G

DISCUSS WITH WORKED EXAMPLE ON ABOVE

In terms of a traditional classification problem, $f(x) = y$, the x here is a sequence of notes and the y is a sequence of chord labels. Hence we have a sequence to sequence problem.

A small, simplifying assumption that must be made when addressing the problem in this way is for the case of polyphonic music, where more than one note is played at a time in the melody. In this case we can simply treat each combination of notes as a sequence of notes played in quick succession, with order being determined arbitrarily.

2.2 Natural Language Processing

An often-drawn comparison is that of the similarity between music and language. Many postulate that music is itself a form of language [4] (although perhaps it is more enlightening to say that language is a music). This is useful when analysing music as there is a wealth of literature on natural language processing.

The immediate comparison that can be drawn is to Part of Speech (POS) Tagging. In this problem we are trying to assign grammatical tags to a tokenized sequence of words. This one-to-one mapping problem is analogous to chord labelling in the case where each note is labelled with a corresponding chord:

DIAGRAM

Both are examples of a problem where the visible variables that are observed are dependant on the latent variables that are trying to be determined. This problem would be simple if said latent variables were not dependant on each other:

PLATE DIAGRAM

But this is clearly not the case. An adjective is very likely to be followed by a noun, and a G major is very likely to be followed by a C major. In reality the situation looks more like this:

NON-PLATE DIAGRAM - LATENT VARS CONNECTED

The complex relationships between the latent variables is one area where the difficulty arises in these problems. An often used approach to this is to apply the first-order

Markov Assumption. This allows us to assume that each latent variable h_t is only dependant on the previous latent variable h_{t-1} in a temporal ordering $t \in \{1:T\}$. This model is known as a Hidden Markov Model (HMM). POS Tagging is a success story for the HMM [22].

2.3 Hidden Markov Models

The HMM assumes an underlying latent state space $\mathcal{H}$, with transition probabilities between members. These variables are conditionally independent of each other, given the previous h_{t-1}, i.e. the first-order Markov Assumption:

$$P(h_t|h_{t-1}), \quad h_t, h_{t-1} \in \mathcal{H}, \quad t \in \{1:T\} \tag{2.1}$$
$$I(h_t, h_i|h_{t-1}), \quad i \in \{1:T\} \tag{2.2}$$

And a visible variable space $\mathcal{V}$, which are conditionally independent of each other given the corresponding latent variable at time t:

$$P(v_t|h_t), \quad h_t \in \mathcal{H}, \quad v_t \in \mathcal{V}, \quad t \in \{1:T\} \tag{2.3}$$
$$I(v_t, v_i|h_t), \quad \forall i \in \{1:T\} \tag{2.4}$$

Given these assumptions we can write the entire joint probability distribution as:

$$P(\{v_t\}_{t=1}^T, \{h_t\}_{t=1}^T) = P(v_1|h_1)P(h_1)\prod_{t=2}^{T} P(v_t|h_t)P(h_t|h_{t-1}) \tag{2.5}$$

Which simplifies the complexity of the problem enormously.

There are various forms of inference that can be performed on HMMs. In the case of on-line accompaniment prediction, the probability we are interested in would be $P(h_{t+1}|\{v_i\}_{i=1}^t)$. This can be performed by first summing over all possible states at time t, then calculating probabilities of those h_t by propagating recursively back to the start state. This is referred to as the forward algorithm or *filtering* [37]. In terms of message passing in graphical inference, the message is being passed from all observed variables $\{v_i\}_{i=1}^t$, starting at $t = 1$, summing over the latent variables h_t as the message is passed up the chain.

The case looked at for this report is concerned with the off-line variant of this problem, predicting the most probable sequence of latent variables, given the full observed sequence:

$$\arg\max_{\{h\}_{t=1}^T} P(\{h\}_{t=1}^T|\{v\}_{t=1}^T) \tag{2.6}$$

This is calculated by a form of dynamic programming called the Viterbi Algorithm [37], see section 3.1.2.

So this model allows us to generate the most likely sequence of hidden, latent variables given an observed sequence, with the assumptions the each observed variable is only dependant on the others *through* its corresponding latent variable, and that each latent variable is only dependant on its past *through* the previous time step, and is entirely independent of its future. See section 3.1.1 for a discussion on the problems these assumptions raise for this problem space.

2.4 Neural Networks

One immediate concern with the HMM is that it is unable to capture long term dependencies. It is memoryless, or, more accurately, can only remember one thing: the last place it visited. This contradicts our intuitive grammatical understanding of language. A grammar is a tree structured language model, where the following section can be very dependant on something that happened at the start of the sequence. One example of this in music is repeated phrases; you can't repeat something if you have forgotten it.

What we want is a model that can carry information through it then learn when to forget it and when to incorporate it into deciding the output at a given time. A popular model for this type of system is the Recurrent Neural Network (RNN), in particular with a Long Short Term Memory (LSTM) cell.

Work on neural networks architectures has flourished in the past few years, due largely in part to advances in optimised computation on GPUs [29][20], as well as new pre-training methods such as autoencoders [16][45][7]. However the basic concept has not changed very significantly since its their beginnings in Hebbian Learning and the Perceptron in the 1940s and 1950s [15] [36].

2.4.1 Perceptrons and MLPs

A perceptron is a linear classifier based on an abstraction of a neuron. It is, in essence, a weighted sum of inputs with one output.

PERCEPTRON DIAGRAM

The key observation is that these weights can be trained to create a binary classification model, where the two classes are linearly separable. The classic example of a problem this model cannot solve is the XOR logic gate, as in this case a straight line cannot be drawn which separates the two classes [28].

If we take a row of perceptrons with the same inputs but different weights, we now have one layer of a Multi Layer Perceptron (MLP). MLPs are built by taking the outputs of these layers, applying an activation function such as a logistic sigmoid (σ) or a hyperbolic tangent (*tanh*), then treating these values as the inputs to another layer.

MLP DIAGRAM

Another name for these models are Feed-Forward Artificial Neural Networks (ANNs). By continuing like this we can build an arbitrarily large ANN, which can be used to model more and more complex problems. In fact it can be proven that even just a single layer, finite length ANN can be used to approximate any continuous function on compact subsets of $\mathbb{R}^n$ [6][19]. This astounding property, known as the Universal Approximation Theorem, is one of the reasons that neural networks are outperforming many traditional mathematical models currently, in a variety of fields from computer vision [20] to protein modelling [44].

However this theorem gives no algorithm for building these complex models. The difficulty comes in training the weights to correctly model the problem, and in the cases of very deep models this can mean millions of parameters. Even worse, there is no analytic solution in optimising these parameters, they must be trained iteratively. This means a large amount of computation time and resources.

To train an ANN in a supervised manner, an error function is needed between the output of the network and the training labels. To continue the notation introduced in the HMM section we will let the $\{v_i\}_{i=1}^{T}$ be the input to the network, $\boldsymbol{v}$ using vector notation. The output variables are re-named $\{y_i\}_{i=1}^{S}$, or $\boldsymbol{y}$, to avoid confusion with the hidden layers of the network, $\boldsymbol{h}_i$, for the ith layer. Let the training label associated with $\boldsymbol{y}$ be denoted $\boldsymbol{t}$. A superscript in parentheses, $\boldsymbol{y}^{(j)}$, will be used to denote the jth instance of data, from a data sample of size N. Hence we have an error function:

$$\sum_{j=1}^{N} E(\boldsymbol{y}^{(j)}, \boldsymbol{t}^{(j)}) \tag{2.7}$$

The weights are then updated in a way that minimises this error function. To do this it is necessary to take the derivative of the error with respect to each weight. This is simple to do with a single layer but becomes more complex with a deep network. The method for achieving this is called Back Propagation and was created by Paul Werbos in 1974 [47]. Methods for iteratively updating the gradients based on these derivatives are called Gradient Descent and tend to take the following form:

$$w_{ik} := w_{ik} - \eta \frac{\partial E(\boldsymbol{y}, \boldsymbol{t})}{\partial w_{ik}} \tag{2.8}$$

Where w_{ik} is the kth weight of the ith layer. These gradients can either be done per data point j or by summing over batches as in 2.7. By this method the neural network can be slowly pivoted towards on optimal configuration, but there is no guarantee of avoiding local minima, and even if it manages to perfectly fit the training data, it will likely be overfit and not generalise to held out test data. Despite these problems, advances in processing speeds and hardware level algorithms [21] have made training these models possible.

2.4.2 Recurrent Neural Networks and LSTMs

So is it possible to use neural networks to predict a sequence of chords from a sequence of notes? Unfortunately, in the current state described, the ANN takes a vector $\boldsymbol{v}$ of fixed-size T as input, and outputs $\boldsymbol{y}$ of fixed size S. What is needed is a network with a flexible input and output size, a tricky problem under the current architecture as this would been adding and removing weights for every data sample. Recurrent Neural Networks (RNNs) solve exactly this problem.

An RNN is a neural network that scales dynamically to it's input. It is composed of *cells* that share weights and are connected together in a chain. A single RNN cell looks like a standard feed-forward network, but it's input v_t is combined with an output from the previous cell h_{t-1}. Architectures vary but it is common for the output of the cell, y_t, to be the output of a second layer that takes h_t as input, and for the output of this hidden layer, h_t, to be passed to the next cell in the chain. Of course these cells can be further stacked to result in deeper architectures that learn higher-level features of their input.

RNN DIAGRAM

To train these models we can unfold them after receiving all of the input, so that it looks like a complex feed-forward model, that back propagate the error in a similar method to before, now called Back Propagation Through Time (BPTT) [48].

ROLLED OUT RNN BPTT

A problem with this method, referred to as the vanishing gradient problem [17], is where the earlier weights in the network receive a much smaller update than those closer to the final output, as the back propagated gradient at time t has been through $T-t$ applications of the chain rule, each involving multiplication with numbers < 1, resulting in the gradient becoming incredibly small. Hence errors caused by weights near the start of the sequence won't be corrected.

One proposed solution to this is the Long Short Term Memory (LSTM) Cell [18]. This is a more complex cell structure involving 3 *gates* and an explicit cell state, c_t, that serves as the system's memory. Each gate combines the input and the cell state in a way that serves an intuitive purpose.

The first gate is called the *forget gate*. It takes as input a concatenation of the input v_t and the output of the previous hidden layer h_t, performs a weighted sum, and uses a logistic sigmoid to output a value between 0 and 1 for each component of the cell state c_t. Through element-wise multiplication, this is used to decide which elements of the cell state to forget at this time step. This is the equation for the input gate before updating the cell state (note that bias terms are omitted).

$$f_t = \sigma(W_f[h_{t-1} : v_t]) \tag{2.9}$$

The next gate is the *input gate*. This is used to determine what new information to add to the cell state. The components to update are chosen in a similar manner to above,

then the *tanh* function is used to generate new updates from the input.

$$i_t = \sigma(W_i[h_{t-1} : v_t]) \tag{2.10}$$
$$c_t^* = \tanh(W_c[h_{t-1} : v_t]) \tag{2.11}$$

These updates are then applied together to the cell state.

$$c_t = f_t * c_{t-1} + i_t * c^* t \tag{2.12}$$

Where here $*$ signifies element-wise multiplication.

Lastly the output gate decides what to output as h_t for this time step. It chooses components by the same method as the previous gates, then generates the output from the cell state.

$$o_t = \sigma(W_o[h_{t-1} : v_t]) \tag{2.13}$$
$$h_t = o_t * \tanh(c_t) \tag{2.14}$$

The full cell takes this shape:

LSTM Image

This cell represents a single layer LSTM model. This could be used as the full model, in which case the h_t would be the outputs y_t of the system. Alternatively these LSTM cells could be stacked, creating a deep learning model where the h_{it} of each layer i would be the input to the layer $i+1$, where h_{1t} are the original inputs v_t.

All this serves to create a model where information can be remembered and forgotten dynamically based on where we are in the input sequence. This is very useful for the purposes of this project as repeated musical phrases and structures could be implicitly captured by the model, as shown to be possible by [10].

LSTMs are currently the cutting edge for many different areas of natural language processing, including language modelling [43] [34], speech recognition [14] and language generation in spoken dialogue systems [46].

For an in-depth discussion on LSTMs, [30] or [13] is recommended.

2.5 Review of Previous Work on Automatic Harmonic Analysis

A brief review of previous work on chord progression estimation and related works will be presented in this section. Note that this is largely unchanged from the report presented last year.

The field of automatic harmonic analysis of music has seen a wide range of research in the past few decades. Much of this stems from the interpretation of music as a form of language, or at least recognising the similarities between music and natural language and exploiting them to achieve certain tasks. The works of [50] and [12] were some of the first applications of computational linguistic theory to the field of music, inspired by the pioneering work of Noam Chomsky on the formal definitions of languages and syntax [26]. In [50] a generative grammar for the parsing of a harmonic phrase was proposed, using an adaptation of the formalism of tonal harmony proposed by [11]. This mechanism was capable of automatically parsing chorales.

An alternative method of harmonic analysis, introduced by [39], was used in another computationally assisted, automatic parsing of music in [42]. Similar to the above, Smolier applied existing NLP parsing techniques to a grammatical formalism of tonal music.

Logic based approaches have also been attempted. In [9] an expert system for the harmonic analysis of chorales is proposed, based on first order predicate logic. This was expanded on by [27], where a full approach applicable to all tonal music was presented. A similar hierarchical logical representation for the computational analysis of music was proposed by [41].

The development of probabilistic machine learning algorithms allowed for a more statistical approach to harmonic analysis to be developed. The work of [23] applies neural networks to chord classification, focusing on the problem of how to best represent pitch. A more qualitative perception model for musical learning is presented in [49].

A popular and successful method of probabilistic harmonic analysis is the HMM based approach. The harmonic analysis of chorales proposed in [1] uses HMM probabilistic inference. In [25] an HMM based method for chord generation from a hummed input is presented. A simple frequency count based HMM is used in the chordal analysis of the MySong application by Microsoft [40,], a system that automatically generates an accompaniment for a real-time vocal input. A more complex HMM based system is utilised by [38] in the automatic transcription of melody and chords from the first eight Beatles albums.

The HMM based approach shall form the basis of the baseline model for this project, where in the following year the markov assumption shall be replaced with a more complex grammar based language model.

In more recent years, a template based approach to chordal analysis has been proposed, notably in the work of [33], [32] and [31]. This method moves away from more statistical techniques and instead assigns a rule-based scoring to a frame, based on whether the observed notes are present in the stored chord model template.

More complex hybrid approaches have also been attempted. A combination of neo-Riemannian transformations, Markov chains and Support Vector Machines are used in [3] for the generation of style-specific accompaniment. This method uses machine learning techniques to decide how probable it is that an observed note is a 'chord tone' for the current chord and is the inspiration for one of the proposed emission models in this report.

It should be noted that the majority of these approaches assume pre-segmentation of the input into distinct chord segments for classification, as does the work presented in this report. A complete model of chordal analysis must take this aspect into account as well. There are many well researched methods for accomplishing this, a good review of a few of these is provided in [33].

Chapter 3

Design and Implementation

The aim of this chapter is to take the reader through the design of the models, by way of example on a piece from the KP Corpus, and to explain the details of the implementation. The models based on HMMs differ in structure from the LSTM models so they is a section for each of these.

3.1 Hidden Markov Model Based System

3.1.1 Design

The first year of this project was focussed on building a HMM based approach to the problem of chord labelling, with a view to use it as comparison to a model that was able to capture a higher level of structure this year. This section will briefly discuss the design of the HMM system, with a focus on a new design that was implemented this year, see [**?**] for an in depth description of the previous year's implementations.

MINUET IN G

Above is the example piece from the previous chapter, Minuet in G (often attributed to Bach). To reiterate, the task is to take as input the sequence of notes in both staves and to predict the sequence of chord labels. The first thing to notice is that there is not a one-to-one mapping between notes and chord label, there are many more of the former. If we let the chord labels be the latent variables $\{h_t\}_{t=1}T$ of our HMM and the notes be $\{v_t\}_{t=1}T$, then v_t represents a group of notes that is generated by one chord. This concurs with our intuition of the problem, where in essence we are using the HMM to model the 'generation' of notes by a sequence of chords. We can think of this as a song generating machine that can be in a number of states, each state representing a chord. This starts in some state and outputs of group of notes, then transitions to another state and outputs a new sequence of notes. In our example the first state would be the chord of G, and the first group would be the first two bars of notes, where the second state is the chord C and the second group is just the third bar of notes.

The models of the first year of this project used exactly this assumption and addressed the problem of going from a sequence of note groups to a sequence of chords. The inherent difficulties in this are that, for each state in the HMM, a smaller model must be constructed to capture the generation of this note group. Three approaches were used, one being a smaller HMM for each step, one being a 'bag of notes' model, analogous to a 'bag of words' model in natural language processing [**?**], and another more complex decision tree model. However these models do not capture how to split the observed notes into groups to start off with, so some pre-processing was assumed and necessary in the experiments. This is unsatisfactory as it does not provide a full model of the problem. Another, more fundamental problem is that the above assumptions discard a crucial indicator of the chord transitions. That is, the transitions between the notes at the end of one group and the beginning of the other. In the example, on the transition from the second chord *C* to the third chord, another *G*, there is a transition from an *F#* note to a *G* note one semitone above it. This is a key indicator of a chord transition as it the melody is stepping up to the root note of the chord from the note below. This is something that would be beneficial to capture in the model.

Hence a new design of the HMM system is proposed and implemented this year, before the step to LSTMs. Consider a modification to the above machine. This machine outputs a sequence of notes one at a time, starting in some state. The time steps now correspond to each note, not to each chord, hence the machine is essentially self-transitioning for each note in the chord group. Then at some point the machine will transition to a new state and start generating more notes, but from this new state. The task is now to take the full sequence of notes and to try to determine to most likely sequence of states this machine was in when it generated them. Note that we now have a one-to-one mapping between the sequence of notes and the sequence of chords, see the re-labelling of the example below:

RELABEL MINUET IN G

This corresponds to the following HMM:

HMM Graph

Now the v_t each correspond to a single note, and the probabilistic model of $P(v_t|h_t)$ is a simple multinomial distribution that can be observed from the training data, instead of a complex mini-sequence generation task.

DISCUSS RESULTS OF RUNNING HMM ON MINUET IN G

3.1.2 Implementation

To implement the Hidden Markov Model used in the previous year's work, it was necessary to be modular in approach, with a re-usable high level structure of state transitions (the Transition Model) and a pluggable variety of chord state to note group models (the Emission Model). This allowed for the implementation and testing of the three main emission models described above. However, the majority of HMM

libraries that exist in Python, the chosen language for this project, are restricted to basic Gaussian and Multinomial emission models [24] [2].

3.2 Recurrent Neural Network Based System

3.2.1 Design

3.2.2 Implementation

Chapter 4

Experimental Results

4.1 Datasets

4.1.1 Weimar Jazz Database

4.1.2 KP Corpus

4.2 Metrics

4.3 Baselines

4.4 HMM Results

4.5 LSTM Results

4.6 Other Results

Chapter 5

Conclusion

5.1 Discussion

5.2 Future Work

Bibliography

[1] Moray Allan and Christopher KI Williams. Harmonising chorales by probabilistic inference. *Advances in neural information processing systems*, 17:25–32, 2005.

[2] L Buitinck. seqlearn. `http://larsmans.github.io/seqlearn/`, 2013.

[3] Ching-Hua Chuan and Elaine Chew. A hybrid system for automatic generation of style-specific accompaniment. In *Proceedings of the 4th International Joint Workshop on Computational Creativity*, pages 57–64. Citeseer, 2007.

[4] Cynthia Cohen. Music: A universal language. *Music and conflict transformation. Harmonies and dissonances in geopolitics*, pages 26–39, 2008.

[5] Leonard Cohen. *Songs from a Room*. Song BMG Music Entertainment, 1969.

[6] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.

[7] Li Deng, Michael L Seltzer, Dong Yu, Alex Acero, Abdel-rahman Mohamed, and Geoffrey E Hinton. Binary coding of speech spectrograms using a deep auto-encoder. In *Interspeech*, pages 1692–1695. Citeseer, 2010.

[8] Bob Dylan. *Blood on the Tracks*. Ram's Horn music, 1975.

[9] Kemal Ebcioğlu. An expert system for harmonizing chorales in the style of js bach. *The Journal of Logic Programming*, 8(1-2):145–185, 1990.

[10] Douglas Eck and Jasmin Lapalme. Learning musical structure directly from sequences of music. *University of Montreal, Department of Computer Science, CP*, 6128, 2008.

[11] Allen Forte. *Tonal harmony in concept and practice*. Holt, Rinehart and Winston, 1962.

[12] Allen Forte. Syntax-based analytic reading of musical scores. 1967.

[13] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 2016.

[14] Song Han, Junlong Kang, Huizi Mao, Yiming Hu, Xin Li, Yubin Li, Dongliang Xie, Hong Luo, Song Yao, Yu Wang, et al. Ese: Efficient speech recognition

engine with sparse lstm on fpga. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 75–84. ACM, 2017.

[15] Donald O Hebb. The first stage of perception: Growth of the assembly. *The Organization of Behavior*, pages 60–78, 1949.

[16] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[17] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[19] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[21] Jens Krüger and Rüdiger Westermann. Linear algebra operators for gpu implementation of numerical algorithms. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 908–916. ACM, 2003.

[22] Julian Kupiec. Robust part-of-speech tagging using a hidden markov model. *Computer Speech & Language*, 6(3):225–242, 1992.

[23] Bernice Laden and Douglas H Keefe. The representation of pitch in a neural net model of chord classification. *Computer Music Journal*, 13(4):12–26, 1989.

[24] S Lebedev. hmmlearn. `https://github.com/hmmlearn/hmmlearn`, Feb 2015.

[25] Hong-Ru Lee and JS Roger Jang. i-ring: A system for humming transcription and chord generation. In *Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on*, volume 2, pages 1031–1034. IEEE, 2004.

[26] Robert B Lees and N Chomsky. Syntactic structures. *Language*, 33(3 Part 1):375–408, 1957.

[27] H John Maxwell. An expert system for harmonizing analysis of tonal music, understanding music with ai: perspectives on music cognition, 1992.

[28] Marvin Minsky and Seymour Papert. Perceptrons. 1969.

[29] Kyoung-Su Oh and Keechul Jung. Gpu implementation of neural networks. *Pattern Recognition*, 37(6):1311–1314, 2004.

[30] C. Olah. Understanding lstm networks. `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`, Aug 2015.

[31] Laurent Oudre, Cédric Févotte, and Yves Grenier. Probabilistic template-based chord recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(8):2249–2259, 2011.

[32] Laurent Oudre, Yves Grenier, and Cédric Févotte. Template-based chord recognition: Influence of the chord types. In *ISMIR*, pages 153–158, 2009.

[33] Bryan Pardo and William P Birmingham. Algorithms for chordal analysis. *Computer Music Journal*, 26(2):27–49, 2002.

[34] Karl Pichotta and Raymond J Mooney. Using sentence-level lstm language models for script inference. *arXiv preprint arXiv:1604.02993*, 2016.

[35] Eric Regener. *Pitch notation and equal temperament: A formal study*, volume 6. University of California Press, 1973.

[36] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[37] Stuart J Russell and Peter Norvig. Artificial intelligence: a modern approach (international edition). 2002.

[38] Matti P Ryynänen and Anssi P Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *Computer Music Journal*, 32(3):72–86, 2008.

[39] Heinrich Schenker. *Harmony*, volume 1. University of Chicago Press, 1979.

[40] Ian Simon, Dan Morris, and Sumit Basu. Mysong: automatic accompaniment generation for vocal melodies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 725–734. ACM, 2008.

[41] Alan Smaill, Geraint Wiggins, and Mitch Harris. Hierarchical music representation for composition and analysis. *Computers and the Humanities*, 27(1):7–17, 1993.

[42] Stephen W Smoliar. A computer aid for schenkerian analysis. In *Proceedings of the 1979 annual conference*, pages 110–115. ACM, 1979.

[43] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Interspeech*, pages 194–197, 2012.

[44] Karolis Uziela, David Menéndez Hurtado, Nanjiang Shu, Björn Wallner, and Arne Elofsson. Proq3d: Improved model quality assessments using deep learning. *Bioinformatics*, page btw819, 2017.

[45] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[46] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. Multi-domain neural network lan-

guage generation for spoken dialogue systems. *arXiv preprint arXiv:1603.01232*, 2016.

[47] P Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, 1974.

[48] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[49] Gerhard Widmer. Qualitative perception modeling and intelligent musical learning. *Computer Music Journal*, 16(2):51–68, 1992.

[50] Terry Winograd. Linguistics and the computer analysis of tonal harmony. *journal of Music Theory*, 12(1):2–49, 1968.