UNIVERSITÉ LIBRE DE BRUXELLES

FACULTY OF SCIENCES

DEPARTMENT OF COMPUTER SCIENCES

MELODY GENERATION THROUGH TEMPERATURE-SCALING WITH IDYOM: A PYTHON IMPLEMENTATION

TANGUY LISSENKO

SUPERVISORS:

PROF. HUGUES BERSINI

LLUC BONO ROSSELLÓ

MASTER THESIS IN COMPUTER SCIENCE - ACADEMIC YEAR 2023/2024

Abstract

This master thesis presents the development of a Python-based reimplementation of the Information Dynamics of Music (IDyOM) model, the state-of-the-art model for modeling cognitive processes related to melodic perception and expectancy. The original implementation of IDyOM in Common Lisp presents accessibility challenges for researchers in music cognition, as Lisp is not widely used in this field. Therefore, the primary objective of this work is to create a more accessible and extensible Python version of IDyOM that maintains its predictive capabilities and allows for further model extensions. Our results indicate that the reimplemented model exhibits superior performance compared to IDyOMpy, a competing Python program, in predicting human patterns of expectation. However, while the reimplemented model yields valuable results, its performance is lower than that of the original IDyOM model when trained under identical conditions. As a specific extension of the model, we developed a random-walk sequence generation process that can serve as a co-creativity tool for composers. We also assessed the effectiveness of temperature scaling in generating melodies belonging to the style of the training set. As anticipated, experiments utilizing random-walk with temperature scaling revealed a strong negative correlation between temperature and melody ratings. This indicates that lower temperatures produce melodies more consistent with the style of the training set.

Contents

List of Figures

List of Tables

Introduction

Statement of the Problem

This thesis presents a modeling approach for the understanding of the cognitive processes involved in melodic perception and composition. The primary computational challenge addressed is the problem of sequence prediction: given an ordered sequence of discrete events (the context), the model must determine the probability of any subsequent event, thereby predicting the identity of the next event. Regarding melodic perception, each event in a sequence of melodic events is associated with a probability. A lower likelihood indicates a higher degree of surprisal for the listener, as the event was less expected within the given context. For melodic composition, predicting the most likely subsequent events based on the initial context enables the creation of a stylistically appropriate melodic sequence.

An additional challenge in sequence prediction arises from the nature of musical data. In language modeling, events are unidimensional, corresponding to words. However, in music modeling, events correspond to notes in a score and are multidimensional. Notes comprise a finite set of descriptive variables or features, such as pitch, duration, velocity, and onset time. Therefore, one of the primary challenges in modeling music cognition and composition is identifying the appropriate set of features that yield the best predictive results.

Current situation

Early attempts to model melodic perception were rule-based models. These initial models were grounded in universal principles, often drawing analogies to visual perception. Researchers believed that certain fundamental rules could be applied universally to understand how melodies are perceived and processed by the human brain. More recent approaches have turned to statistical learning, moving away from innate and universal rules. These models aim to learn the probabilistic grammar of a musical style by analyzing the regularities and patterns present in a set of melodies. By capturing these patterns, one can make probabilistic predictions about the likelihood of subsequent events in a melody.

Currently, one of the most advanced computational models for auditory melodic perception is IDyOM [Pea05; Pea18] (Information Dynamics of Music). This model simulates a listener who, through statistical implicit learning, progressively acquires an internal representation of musical regularities over both short and long time scales. IDyOM uses probabilistic predictions to model the listener's experience of surprisal/unexpectedness for each note in a melody. The surprisal is a measure of how unexpected a particular event is within a given context.

Advantages and Limitations of the Current Situation

IDyOM is the state-of-the-art model for melodic perception, as it best accounts for patterns of expectations derived from behavioral experiments. Furthermore, the model allows for studying of different musical styles, since the knowledge lies in datasets composed of melodies belonging to that style. Data formats include **kern and MIDI. This capability enables researchers to efficiently study melodic perception by simulating a listener in various contexts. For instance, the model can simulate listeners with high or low musical exposure, those exposed to different musical styles, and those utilizing knowledge accumulated over their lifetime.

However, the current situation also presents several disadvantages. Firstly, IDyOM is implemented in Common Lisp, which is not a popular choice for research, especially within music psychology and cognition. Additionally, the installation process is complex for users unfamiliar with this ecosystem of tools, involving multiple steps and requirements, including the installation of SBCL, Emacs, Quicklisp, and Sqlite. In fact, the user has to run the program within Emacs, as it uses the SLIME mode, which is an Emacs mode for developing Common Lisp applications. These factors make IDyOM a model that is not easily accessible for researchers and difficult to modify, as learning a set of technologies that are not widely used in the field is necessary.

To overcome these problems, attempts have been made to make the model more accessible or to re-implement it in another language. However, each attempt has failed to meet at least one of the following criteria: a model that can account for patterns of expectations from behavioral data and is also easy to use and modify for research.

Regarding models of melodic compositions, despite the amount of research on the subject, attempts to sample stylistically successful melodies from statistical models have often been unsatisfactory [DE10; Pin56; Pea05; CW03b]. This motivates a sampling technique using a random-walk process with temperature scaling to produce more stylistically robust melodies.

Research Objectives

First Objective

The primary objective is to implement a model of melodic perception as a reimplementation of IDyOM [Pea05]. This implementation is done in Python, a popular programming language for research and machine learning. By reimplementing the model in Python, we aim to make it easily accessible and extensible for the research community. Additionally, the reimplementation should match the performance of IDyOM in accounting for the observed patterns of expectations exhibited by human listeners.

Second Objective

Since the processes and representations involved in melodic perception are hypothesized to be equivalent to those in melodic composition, past research has utilized IDyOM for melody composition by sampling notes from the distribution returned by the model [Pea05]. With the implementation of IDyOM in Python, it becomes feasible to extend the model for various tasks, including melodic composition.

Consequently, the second objective is to adapt existing techniques for sampling melodies from statistical models of musical structures, specifically to IDyOM. We will employ temperature scaling, which modifies the distribution from which events are sampled to either produce melodies that are more stylistically coherent or melodies that favor exploration. While this technique does not propose a cognitive process for composition and thus resides outside the cognitive domain, it is a valuable tool for musical inspiration. We hypothesize that this approach will empower the model to generate melodies that are more stylistically successful.

Scope of the Research

Both the statistical model developed in this work and its extension for music generation are limited to the study and generation of melodies (see Chapter 1). Consequently, this research does not address musical content with vertical structure, where different parts contribute to global harmony.

Furthermore, the musical data used in this research consists of melodies in a symbolic representation, represented as a sequence of high-level symbols. We do not focus on audio or signal-based representations of music, and therefore, variations in timbre that could influence a listener's expectations are not considered.

Dissertation Outline

Chapter 1 provides a brief introduction to the fundamentals of music theory. The aim is not to be exhaustive but to assist readers who are familiar with computer science but are unfamiliar with the formal system of music.

Chapter 2 is a review of different models of melodic perception. We present IDyOM, as well as other implementations that were intended to make IDyOM it more suitable for research. We also review different models of melodic perception, such as traditional rule-based models. Then we review the literature on melodic composition using statistical models of musical structures.

Chapter 3 presents the various components of the reimplementation of IDyOM developed in this research. We will focus on the theory behind the implementation so that it can be reproduced, not on how the Python code works since it will be subject to changes in the future.

Chapter 4 presents the details of the process of sequence generation using the developed Python IDyOM implementation. We introduce the random sampling technique, a common method in sequence generation. As a modification to the traditional random sampling process, we incorporate temperature scaling [Guo+17] to reduce uncertainty or enhance confidence in the generation process.

Chapter 5 presents the methodologies, experiments, and results from our IDyOM implementation: a description of the dataset used in the experiments, an overview of the experiments conducted to evaluate melodic expectancy and finally we assess the effectiveness of temperature scaling combined with random walk in producing stylistically successful melodies.

Chapter 6 is the conclusion, where we discuss our findings and reflect on the limitations of our research.

1

Fundamentals of Music Theory

This chapter provides a brief introduction to the fundamentals of music theory. Given that this research utilizes datasets composed of melodies, we will review key concepts such as pitch, intervals, rhythm, scales, time signatures, and key signatures, as well as the definition of a melody. The aim is not to be exhaustive but to assist readers who are familiar with computer science but unfamiliar with the formal system of music.

1.1 Pitch and Intervals

1.1.1 Pitch

Pitch refers to how high or low a sound is. It is determined by the frequency of the sound wave. Higher frequencies produce higher pitches, and lower frequencies produce lower pitches. We can denote a note by its pitch using the term *tone*.

1.1.2 Pitch Classes

In the Western musical notation system, there are twelve distinct pitch classes, each represented by a letter and, if necessary, an accidental. The complete set of pitch classes includes:

$$\mathrm{A}, \mathrm{A}\sharp/\mathrm{B}\flat, \mathrm{B}, \mathrm{C}, \mathrm{C}\sharp/\mathrm{D}\flat, \mathrm{D}, \mathrm{D}\sharp/\mathrm{E}\flat, \mathrm{E}, \mathrm{F}, \mathrm{F}\sharp/\mathrm{G}\flat, \mathrm{G}, \mathrm{G}\sharp/\mathrm{A}\flat$$

Using accidentals, such as sharps (#) and flats (♭), allows for the modification of these pitch classes, raising or lowering the pitch by a semitone, respectively. For instance, C# is one semitone higher than C, while B♭ is one semitone lower than B.

1.1.3 Intervals

An interval is the distance between two pitches. We measure this distance in *semitones* and *whole tones*. A *semitone* is the smallest musical interval commonly used in Western music. It is the distance between two adjacent keys on a piano keyboard. For example the intervals from C to C♯ or E to F are semitones. A *whole tone*, consists of two semitones. For example, the interval from C to D spans two semitones.

An *octave* is the interval between one musical pitch and another having half or twice its frequency. In simpler terms, when a note is played, its octave can be found by going up or down the scale by 12 semitones. For example, we denote the note C in the fourth octave as C_4 and we denote the note C in the fifth octave as C_5, which is 12 semitones higher than C_4.

There are names for every other interval, however it is out of the scope of this simple explanation. Furthermore one can always refer to an interval by the number of whole tones and semitones it spans.

1.2 Scales and Key Signatures

A *scale* is a sequence of pitches in ascending or descending order. There are many different scales that are identified by their patterns of whole and half steps.

The most common scale in Western music is the *major scale*, which follows a specific pattern of whole and half steps. For example, the C major scale consists of the pitches:

C, D, E, F, G, A, B, C

The pattern of whole (W) and half (H) steps/tones for the major scale is:

$$W - W - H - W - W - W - H$$

From a scale, we can derive a *key signature*, which indicates the specific pitches used within a given scale. The key signature is represented by a series of sharps ($\sharp$) or flats ($\flat$) placed at the beginning of a musical staff. For instance, the key signature of C major has no sharps or flats, while the key signature for G major includes one sharp (F$\sharp$). Furthermore, by knowing the key signature and the scale it represents, we can identify which notes are considered *in scale* and which are *out of scale*.

1.3 Rhythm and Time Signature

Rhythm refers to the timing of sounds in music. It is described by the duration of notes and rests. Common note durations in Western music include:

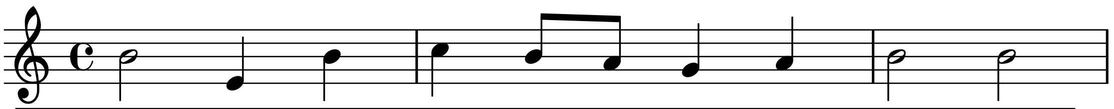
- *Whole note* (W): Lasts for four beats.
- *Half note* (H = $\frac{1}{2}$W): Lasts for two beats.
- *Quarter note* (Q = $\frac{1}{4}$W): Lasts for one beat.
- *Eighth note* (E = $\frac{1}{8}$W): Lasts for half a beat.
- *Sixteenth note* (S = $\frac{1}{16}$W): Lasts for a quarter of a beat.

A *time signature* specifies how many notes of a particular duration are contained in each measure. Common time signatures are 4/4 and 3/4. For example, 4/4 means four beats are in each measure, and a quarter note receives one beat.

1.4 Melodies and Attributes

A *melody* is a linear succession of notes, combining pitch and rhythm to create a musical line. At any given moment, only one note is present, which characterizes it as a monophonic composition. A monophonic composition means that a single voice performs the melody, and a single voice will necessarily sing a melody since it is not possible for that voice to sing two notes simultaneously.

An example of a melody is shown in Figure 1.1. This melody, is conveniently represented in sheet music format. Sheet music is a system of musical notation that displays attributes such as pitch and note durations, as well as global features like key signature and time signature. It is an ideal format for sharing and reproducing compositions. In the table below the sheet music in Figure 1.1, various attributes of each note in the melody are listed. The onset times are measured using a quarter note as the smallest unit of time. Intervals are represented by the number of semitones they span. The time signature is 4/4, and the key signature is in C major; however, it is important to note that this brief excerpt may not accurately represent the tonality of the entire piece.



	$\mathbf{e}_1$	$\mathbf{e}_2$	$\mathbf{e}_3$	$\mathbf{e}_4$	$\mathbf{e}_5$	$\mathbf{e}_6$	$\mathbf{e}_7$	$\mathbf{e}_8$	$\mathbf{e}_9$	$\mathbf{e}_{10}$
Pitch	B_4	E_4	B_4	C_5	B_4	A_4	G_4	A_4	B_4	B_4
Duration	H	Q	Q	Q	E	E	Q	Q	Q	H
Onset	0	2	3	4	5	5.5	6	7	8	10
Interval	$\perp$	-7	7	1	-1	-2	-2	2	2	0

Figure 1.1: The first three bars of Chorale 10 from the dataset used in this research, and the associated values for different musical attributes.

We have seen that musical content contains a variety of attributes, each contributing to the character of the melody. Therefore when developing our model for melodic predictions, it is important to consider these different attributes, as some may hold more significance than others. We will have to select the best subset of attributes to enhance the predictive accuracy of the model. While the attributes presented here represent only a fraction of what can be analyzed, many additional attributes can be derived from these basic attributes.

2

State of the Art

The perception of a melody belonging to a specific musical style depends on our knowledge. A musical style is defined by constraints that are specific to a culture. A musical style is characterized by a collection of pieces with similar patterns and constraints in their compositional choices. According to the composer and author Leonard Meyer, these constraints can be viewed as probabilistic grammars that define the syntax of a musical style [Mey57]. Humans acquire these grammars as cognitive components that enable communication and understanding between composers, performers, and listeners. The obtention and modification of this grammar affect our perceptive abilities.

In this chapter, we review the literature on the modeling of melodic perception and composition. Modeling melodic perception requires the acquisition of this probabilistic grammar and a mechanism to make predictions based on the grammar. We begin with an explanation of the Implication-realization theory [Nar90], a rule-based model at the origin of the research on melodic perception. Then we briefly go over the statistical learning hypothesis (SLH) and the probabilistic prediction hypothesis (PPH) which are important to understand the research explained here, especially where we can make the bridge between a computational problem and the field of cognitive science. Next, we present melodic perception models based on statistical learning theory, those that learn exclusively from data. We focus on IDyOM and its various implementations and attempts to make it more accessible for research. Finally, we present several studies that have used statistical models with different sampling techniques to generate melodies [Con03; CW03b; Pea05; DE10].

2.1 The Implication-Realization Theory

2.1.1 The Theory

One theory for the acquisition of such grammars is the implication-realization theory given by Eugene Narmour [Nar90]. Namour suggests that humans have two cognitive systems for perceiving music: the bottom-up system and the top-down system. The former is universal and innate, while the latter is learned through exposure to new musical content. These components use a set of symbolic rules to assess what a listener expects to hear. The perception of music depends on whether or not these expectations are met. In other words, the IR theory is a quantifiable model of human expectations. The IR theory was formulated by Krumhansl as a quantitative model consisting of a small set of symbolic rules [Kru95]. Since the bottom-up system is innate and universal, it is possible to attempt to quantify the principles of the system exhaustively. In contrast, the top-down system is unique to each individual. The bottom-up system generates expectations based on the Gestalt-like principles of proximity, similarity, and good continuation [Nar90]. The expectations generated depend on the intervals and closures associated with them.

The principles of the bottom-up system model the expectations of listeners subject to musical content. The expectations are generated by the realization or denial of the hypotheses a listener makes about the future state of a piece. These principles are based on the Gestalt principles, which constitute a theory of perception based on pattern recognition [Wag+12]. In the IR theory, musical intervals are associated with a degree of closure. If the degree of closure is strong, it means the end of the current musical structure and we say that the interval is *realized*. If the degree of closure is weak, it is an *implicative* interval since it generates expectations for the next interval. Here are the different principles of the bottom-up system:

- **Registral Direction**: This principle suggests that when a listener hears a melodic interval within a small range, it is expected that the melody continues in the same registral direction. In contrast, large intervals imply a change in registral direction. The registral direction refers to the upward or downward movement of pitch in the sequence.

- **Intervallic Difference**: Small intervals imply an interval similar in size while large interval implies a smaller interval.
- **Registral Return**: If a realized interval changes registral direction with respect to the implicative interval, then this implies a return to the pitch region of the implicative interval's first tone.
- **Proximity**: Describes a general implication for small intervals (5 semitones or less). The implication varies according to the absolute size of the interval.
- **Closure**: Intervals between two consecutive notes have a degree of closure. Closure is determined by two conditions: a change in the registral direction, and a change to a smaller interval.

The first three principles take boolean values, while the last two have a larger value space.

2.1.2 Testing the Theory

The formalization of the IR theory was tested against patterns of expectations given by participants in empirical studies. Regarding the continuation of a two-tone context, the 5 principles of the IR theory accounted for 64% of the variance in the mean continuation ratings. Regarding the continuation of a longer context, the 5 principles of the IR theory accounted for 65% in mean continuation rating [Sch96]. Regarding patterns of expectations for each note in a melody [MWJ92], the IR theory accounted for 13% of the variance in the data [PW06].

The results for the continuation of a two-tone context and a longer context favor the IR theory. However, it has been shown that certain principles of the theory explain the variance in the ratings in a redundant way [Sch96]. Therefore Schellenberg simplified the original model, keeping only two modified principles of the original theory. This new model performed better than the original version. In contrast to the positive results of the IR theory when it comes to the continuation of a context, the IR theory is unable to account for complex patterns of expectation in a melody.

2.2 Statistical Learning of Melodic Expectancy

As mentioned in the previous section, the IR theory is inefficient for accounting for the variance of melody expectation pattern data. This led to the development of models based on another theory, statistical implicit learning, presented here. This section is necessary to understand the computational models based on this statistical learning theory. The models of expectancy based on the statistical learning theory are in principle able to account for behavioral data better than the quantitative formulation of IR theory [Pea05]. It also requires fewer assumptions about the cognitive mechanisms underlying music perception. This theory is based on two hypotheses: The statistical learning hypothesis (SLH) and the probabilistic prediction hypothesis (PPH).

2.2.1 The Statistical Learning Hypothesis

The statistical learning hypothesis states that enculturation is an implicit statistical learning process. A listener exposed to musical content will progressively acquire internal models of the regularities in music over short and long time scales [Pea18]. Stating this hypothesis, Pearce questions the influence of a bottom-up system and a top-down system on melodic perception, and argues that an innate and general-purpose learning mechanism is capable of learning representations and behaviors that are approximated by the principles of the IR theory. It can encompass the universal representations and patterns of the bottom-up system while making no assumptions about the underlying cognitive mechanisms, and can be applied to any domain.

The mechanism for the acquisition of such an internal model of expectation is called statistical learning/implicit learning. Implicit learning is a process during which a subject, exposed to a stimulus belonging to a complex domain governed by rules, acquires knowledge about the domain without intending to and without being aware of it [Reb67]. It is an unsupervised, implicit mechanism by which the brain encodes the probability of distribution of sequential information such as music or language. Experimental studies designed to test the existence of a statistical learning process in a subject are carried out in this way [Saf+99]: a subject is exposed to sequences generated by a finite-state grammar (Markov model). At the end of exposure to the sequences, one tests the ability of the subjects to have used a learning process and whether they have unconsciously learned knowledge/structures. In other words, the experiment is conclusive if it has acquired an internal model of the finite-state grammar from which the sequences were generated.

A hypothesis for the representation of learned knowledge in the brain is the fragments or chunking hypothesis [PP90]. This hypothesis posits that subjects accumulate fragments or chunks of knowledge, such as bigrams, trigrams, and so on, which are weighted by their frequency of appearance in the training set. When subjects are presented with a new string (input), they decompose this string into n-grams and classify it as grammatically correct if it can be decomposed into the sub-n-grams learned during exposure. The chunking hypothesis forms the basis of many computational models of melodic expectancy, such as IDyOM [Pea05; CW03b; Gui18], as it aligns closely with the architecture of a Markov model.

2.2.2 The Probabilistic Prediction Hypothesis

The probabilistic prediction hypothesis (PPH) [Pea05; Pea18] hypothesis assumes that once an internal model of expectation has been obtained through statistical learning, it can then be used to generate probabilistic predictions that organize and process the musical representations encountered. A probabilistic prediction is the process by which the brain estimates an event that is most likely to occur. These probabilistic predictions play an important role in many of our cognitive processes. In fact, evidence for probabilistic predictions has been found in many human cognitive processes such as emotional experience [Ege+13], recognition memory [AAP18], and phrase boundary detection [PW06], among others.

2.3 Statistical Modelling of Melodic Expectancy

Having presented the statistical learning theory, we now review computational models that simulate this statistical learning mechanism [CW03b; Pea05; GRP22]. These models utilize the internal expectation models learned through statistical learning to make probabilistic predictions. Some models extend these probabilistic predictions into generative systems, effectively becoming models of composition.

We focus on models capable of learning melodic structures, though it is also feasible to learn harmonic, timbral, and rhythmic structures through statistical learning [RR12]. Additionally, our review is limited to models employing the chunking hypothesis [PP90] as their knowledge representation theory. These models are commonly referred to as Markov models, n-gram models, or context models.

2.3.1 Context Models and Multiple Viewpoint Systems

This first model of melodic perception by Conklin and Witten [CW03b; WC90] is not primarily a study of music cognition but rather a research focused on the computational and information theory aspects of music. Despite this, it introduces several original and novel concepts that form the foundation of the IDyOM model. These include the use of context models, the prediction by partial matching (PPM) inference method, the multiple viewpoint system, and the combination of predictions from a short-term model (STM) and a long-term model (LTM). Since these concepts form the foundation of every model discussed here, as well as the basis of IDyOM, we will review them in the following sections.

2.3.1.1 Context Model

We describe here a context model, which is a subclass of the probabilistic finite-state, Markov class of grammars. The model must be able to make a prediction $P(e|c)$ for an event e given a context c. The context is a fixed-sized discrete sequence of events that are before the event e. From this prediction $P(e|c)$, one can derive the unexpectedness/surprise of the event given the context with the information content (IC) such that

$$h(e_n \mid e_1^{n-1}) = -log_2 P(e_n \mid e_1^{n-1}) \tag{2.1}$$

The uncertainty of the model in its prediction regarding a specific context is defined as the entropy - the average information content - of the predictive context [Sha51].

$$H(e_n \mid e_1^{n-1}) = -\sum_{e \in \xi} P(e_n \mid e_1^{n-1}) h(e_n \mid e_1^{n-1}) \tag{2.2}$$

An appropriate model architecture for this task is the context or Markov/n-gram model. It is a machine-learning procedure that learns to predict events based on sequences of events belonging to a language. The learning and processing of the knowledge is done from a database of sequences over an event space ξ. This process can be compared to lexical acquisition by statistical learning in infants [SAN96]. The learning phase consists of determining all the sequences in the data set and assigning them a frequency of occurrence. An order n must be determined, which will be the maximum size of the sequence. Once the learning phase is done, an inference method is needed for predictions - to compute the probabilities of an event in a context.

2.3.1.2 Prediction by Partial Matching

From this database of sequences, an inference method is needed to compute the prediction $p(e|c)$. A straightforward method is the maximum likelihood estimation using the longest matching context in the database.

However, to compute the entropy, one assumes that the system is can predict any sequence, even the least likely. This is necessary because a sequence never observed in the database would be predicted with a probability of 0. However the logarithm of zero is not defined. Furthermore, a model that never observed a sequence during its training phase must still be able to predict these sequences, as they may belong to reality. Therefore we want a model that is non-exclusive since every sequence can be predicted with non-zero probability [CW03b] although the sequence might be highly improbable. Therefore we need another inference method than the simple maximum likelihood estimation.

A solution for this requirement is the partial matching algorithm (PPM) [CW03a] where predictions are computed by a weighted linear combination of lower-order predictions.

2.3.1.3 Multiple Viewpoint System

Music is multidimensional, involving various elements beyond mere pitch. A musical note is characterized by elements such as pitch, duration, and timing, including onset time and rhythmic placement. For instance, intervals, or the pitch differences between consecutive notes are important in defining melodic contours. Additionally, the position of notes within a measure impacts the listener's experience. Given this multidimensional nature, the prediction method discussed earlier can be enhanced by making predictions for different *views* on the musical surface, as defined by Conklin's multiple viewpoint system. The integration of various musical features enables more accurate predictions by considering the role of different musical features on our perception.

A multiple viewpoint system [WC90] is a system for making statistical learning predictions using a representation of independent views of the musical surface, each view modeling a specific musical phenomenon. While language can be modeled using only the words of the language, music is more complex because it is multidimensional and these events have an internal structure composed of multiple attributes. Each event on the musical surface represents a note as in a score. An event comprises a finite set of descriptive variables - basic attributes - which take their value from a finite alphabet.

By having different viewpoints that model different types of events on the musical surface, it is also necessary to combine the predictions of these models. For example if one viewpoint models the pitch of notes and another viewpoint models the duration of notes, one must combine the prediction of the two models in a final prediction. The techniques used for combination are varied such as simple arithmetic/geometric mean or alternatives weighted using information theory metrics [WC90; CW03b; Pea05].

2.3.1.4 Short and Long-Term Models

In data compression, it is common to use the predictions of a single model built incrementally as the code is constructed and input data is encountered. However, a method to have better probabilistic predictions is to combine the predictions of two models into a single final prediction. Indeed, when trying to compress text with PPM, model performance is significantly better if we pre-train the model on related and unrelated text [TC98]. This approach is also used in statistical language modeling to improve the performance of n-gram models. We have a model that has been trained on a corpus of text and another *cache* model that is built dynamically on a recent portion of text [KD90].

For musical predictions, a similar method was introduced by Conklin [WC90]. It combines the predictions of two different models: a long-term model (LTM) and a short-term model (STM). Data is added to the LTM as training progresses, while the STM is reinitialized each time a test set composition is predicted. The motivation for the STM is to take advantage of the musical structures of the test composition that might belong only to the test composition. We can draw a parallel with the reality of a listener: the LTM would be the internal probabilistic prediction model optimized as the listener is exposed to new musical content throughout their life. The STM would be a similar internal model, but created from the test composition being processed. As with the different viewpoint models, the predictions of the STM and the LTM must also be combined.

2.3.2 IDyOM

The information dynamics of music (IDyOM) model is the result of research by Marcus Pearce [Pea05; Pea18]. It is a statistical model of melodic expectancy based on research by Conklin [WC90; CW03b]. The model takes up the architectures proposed by Conklin, such as the use of a context model, a short-term and a long-term model, a multiple viewpoint system, and the principle of combining the predictions of the different models into a final prediction. It uses a variant of the PPM algorithm which is PPM*. The idea behind PPM* is not to impose a bound on the order of the sequences in the database; thus we record sequences of any length in the database. We say that the modelling technique is a variable-order technique since for each prediction, the PPM* algorithm determines the appropriate length of context to ensure optimal predictions.

Pearce's research relies on information and compression theory and links the research of Conklin and Witten [CW03b] to the field of cognitive science and musical perception. Indeed, his research is based on the two hypotheses of statistical learning, (SLH) and the probabilistic prediction hypothesis (PPH). Pearce then validated these hypotheses by comparing the model with behavioral data from various studies on melodic perception and expectancy [CL95; Sch96; MWJ92]. This new setting of the same theory has led to several uses of IDyOM in cognitive science to emphasize the importance of probabilistic predictions for music perception [Omi+13; Pea+10; Di +20]. We will review the various methods by which Pearce and others have validated the SLH and PPH hypotheses using IDyOM.

2.3.2.1 Musicological Analysis

An interesting approach to testing the probabilistic prediction hypothesis is to compare the output of the model to the analysis of repertoire pieces by a musicologist. If the comparisons are similar, then IDyOM successfully simulates human cognitive abilities. This approach simply supports the hypothesis, not proves its existence, as it is difficult to generalize to a large number of pieces. However, it does have the advantage of being very explicit in the justification of predictions, given the musicologist's explanation.

Pearce did this experiment [Pea18], he compared an output of IDyOM with Meyer's analysis [Mey73] of the 3 excerpts from Schubert's Octet for a String and Winds. Since Meyer's analysis focuses solely on pitch analysis, IDyOM has been trained to predict pitch only. In every case, IDyOM reflected the analysis of Meyer with the probabilities and IC associated with each note [Pea18].

2.3.2.2 Behavioral Expectedness

To validate the probabilistic prediction hypothesis, the patterns of expectation derived from the model must be compared with patterns of expectations observed in experiments with human listeners. The statistical model must account for the patterns of expectations at least as well as the quantitative formulation of the IR principles. Also since the statistical model training is unsupervised and learns only through exposure to music, this would demonstrate that there is no need for innate and universal bottom-up rules to account for the observed patterns of expectations. And therefore it would demonstrate that musical expectancy can be accounted using statistical induction only [PW06]. Pearce conducted 3 experiments [PW06] to compare the expectation patterns of IDyOM with patterns observed in humans in three different studies [CL95; Sch96; MWJ92]. The contexts in these studies are increasingly complex due to their increasing length.

For each experiment, they compared the statistical model (IDyOM) with the two-factor model of Schellenberg (plus a tonality predictor) [Sch97] which is a simplification of the IR theory without loss of predictive power. They compared the models with respect to scope, selectivity, and simplicity. These criterion are common when comparing psychological models [Cut+92; Sch+03]. The model's scope is its ability to generalize across contexts (e.g. Stimuli and experimental methods). If the scope is too broad, the model could explain random data which is not the purpose of the model. The simplicity is the extent to which IDyOM subsumes the function of the two-factor (IR) model in accounting for behavioral data in an experiment. If IDyOM makes better predictions than the two-factor model and does not subsume certain functions of the model, then we know that the two-factor model is less simpler than IDyOM. Selectivity is the ability of the model to explain data it should explain better than random data. In other words, the model must be selective in its explanatory capacity. For each experiment, the results were in favor of IDyOM. Indeed IDyOM outperformed the IR theory, especially when predicting full patterns of listener's expectations over melodies.

2.3.2.3 A Model for Enculturation

To validate the Statistical Learning Hypothesis (SLH), Pearce tested whether IDyOM, a statistical model for probabilistic predictions, can simulate the effects of enculturation through statistical learning [Pea18]. If IDyOM replicates these effects, it suggests that implicit statistical learning drives enculturation. Two models were created: one for Western listeners, trained on Western folk songs, and another for Chinese listeners, trained on Chinese folk songs. These models made predictions within and between cultures, averaging the Information Content (IC) for each piece to determine its unpredictability for each model. The long-term model (LTM) was trained on the culture's corpus, while the short-term model (STM) was trained on the specific piece's corpus.

IDyOM correctly classifies compositions belonging to a culture with a very low error rate, supporting the SLH. This suggests that listeners become enculturated through implicit statistical learning over time, and that IDyOM is a valid model for studying enculturation.

2.4 Implementations

In this section, we review the different implementations of the IDyOM model. For each model, we will also explain why it does not fully meet one or both of the following conditions: (1) being a model of melodic perception that accurately accounts for patterns of expectations derived from behavioral data, and (2) being easy to use and extend for research purposes. This discussion will further underscore the need for a Python implementation.

2.4.1 IDyOM

The original IDyOM implementation is thoroughly detailed in Chapter 3, which covers all the theoretical aspects of the model's reimplementation.

While the original implementation delivers optimal results as the state-of-the-art model for capturing patterns of expectation [Pea05], its use of Common Lisp makes it less accessible and extensible for research purposes.

2.4.2 py2lispIDyOM

py2lispIDyOM [GRP22] is a Python package that is a wrapper around the original IDyOM program. The purpose of this package is to provide a Python interface for the original IDyOM model and, therefore, make it more accessible to the research community. Indeed the original IDyOM program requires some degree of familiarity with Common Lisp, Emacs, and the SLIME mode, which is an Emacs mode for developing Common Lisp applications. py2lispIDyOM gets around this problem by providing access to a Python interface for configuring and experimenting with IDyOM.

Since py2lispIDyOM uses the original IDyOM implementation, the output of the two programs are equivalent and py2lispIDyOM can therefore account for patterns of expectations given in behavioral experiments. However, we identify two drawbacks to this program. The first drawback is that the user still has to go through the painful process of installing IDyOM, which includes the installation and configuration of SBCL, Emacs, Quicklisp, and Sqlite. The second drawback is that the user cannot modify and extend IDyOM without learning Common Lisp. As a result, py2lispIDyOM is a good alternative for carrying out experiments, but it lacks the functionality to extend the model to further research in this area.

2.4.3 IDyOMpy

IDyOMpy [Gui18] attempts to reimplement IDyOM in Python, which is more oriented towards research than Common Lisp. The aim was to implement a model that would produce results similar to those of IDyOM and that is easy to modify and extend.

However, the model has several problems that make it difficult to work with efficiently. The first issue is that it only implements a small set of viewpoints (pitch, length, interval, and velocity). In comparison, we find 31 different viewpoints in the first IDyOM implementation [Pea05] without including linked viewpoints. The second issue is that despite the lack of viewpoints, the model cannot account for patterns of expectations given in behavioral experiments. It is also very far from the output of IDyOM configured only with the viewpoints of IDyOMpy. We will show the lack of performance of

IDyOMpy in Chapter 5 to motivate the need for a new implementation.

2.4.4 Idyoms

Idyoms [Har24] is a Julia implementation of the original IDyOM model. This implementation extends the capabilities of the original model by extending its use to any domain, not just music, as long as the domain can be represented as a multi-dimensional sequence of symbols. The results returned by this implementation are, according to the author, close to those of IDyOM [Har24].

In this implementation, viewpoints are entirely generic (not specific to music), therefore it is the the user's responsibility to define their atomic viewpoints on their particular data. This means the implementation is more generic but means that the user is responsible for implementing the interface. This feature is great for defining viewpoints belonging to different domains. However, it is not practical in the case of research on melodic perception, as the user has to define each viewpoint and write the code to link a certain musical encoding format (MIDI, **kern) to the encoding used in the program.

2.5 Modelling Melodic Composition

In previous sections, we explored the principles of statistical learning, probabilistic predictions, and the role of computational models of melodic expectancy in understanding musical perception. Building on this foundation, we now shift our focus to melodic composition models, which aim to generate melodies that correspond to the statistical properties of a specific musical style. While we will explore various techniques for sampling melodies independent of the particular objectives of statistical models, our primary focus will be on IDyOM. This emphasis is based on the hypothesis that the processes and representations involved in melodic perception are similar to those found in melodic composition [Pea05].

The goal of a melodic composition model is to create melodies that possess a high likelihood of occurring within a given genre. Drawing from Chomsky's concept of creativity, these models should be able to produce phrases (or melodies) that have not been previously encountered or composed. Our focus here is on statistical models of melodic composition, which are trained on musical corpora to perform their analytical and generative tasks. While rule-based models can achieve considerable stylistic success [Cop89], they are inherently limited by the symbolic rules imposed by their developers. Such constraints restrict the set of possible outputs, preventing these models from demonstrating the creativity that Chomsky describes.

There are many different statistical model architectures: context/n-gram models, complex variable-order n-gram models, deep learning architectures, etc [HCC17] for the computational generation of music. However, their purpose is the same, they estimate a probability distribution over events of the musical surface. In this way, the same statistical model can be analytic or synthetic. In the analytic case, the aim is to classify a melody as belonging to a melody class or not. This classification is made based on the probability distribution, which gives a probability $p_m(s)$ for the melody s, according to the model m. In the synthetic case, the same distribution can be used to sample valid tones and reject those with a low probability. Another method is to navigate the solution space using as a heuristic the probability for the piece returned by the analytical model. This section describes various sampling techniques used in the literature [Con03], together with examples of applications and problems encountered. These sampling methods can be used regardless of the power and complexity of the model.

2.5.1 Random Walk

The random walk/sampling method is probably the most widely used in the literature. At each stage, it consists of sampling a random event from the probability distribution. The termination criterion is reached when the generated sequence reaches a predefined length. However, this method is flawed as it is “greedy” and we cannot be sure that the overall probability of the piece will be high. Indeed, it is possible to reach a local minimum: a state from which there is no transition to a note with a high probability, only low probabilities. An additional problem is the zero frequency problem. It occurs when an item with no continuation is chosen. A proposed solution uses an interpolated smoothing strategy such as the PPM algorithm [CW03b; Pea05].

In the 1950s, random walk with Markov/n-gram models was tested by various research projects to generate simple melodies [Pin56; Bro+57; Hil59]. The Banal Tune-Maker [Pin56] is a Markov chain trained on 39 nursery tunes transposed into the key of C major to allow more data for each transition. The model is of order 1 (1-gram model), therefore a state represents a single note. Another system [Bro+57] from the same period is similar in its simplicity, except that it uses a larger order. The compositions generated by these systems are far from belonging to the tonal language, unlike the melodies in the corpus from which the system was trained. However, the pioneering nature of this composition technique has increased the popularity of some of the pieces generated. One famous example is the Illiac suite, a composition for a string quartet which is very popular since it is the first computer-generated piece and uses a Markov model. Davismoon and Eccles [DE10] highlighted the weaknesses of Markov processes using random walk for melody generation. Although a Markov chain encapsulates useful stylistic information, the melodies generated are unsatisfactory. They identify pitch-related problems and end-point problems.

Conklin and Witten investigated random sampling using a complex statistical model for melody generation. It is a context model that uses a multiple viewpoint system with a short-term and a long-term model. The selected viewpoints were not determined through a formal feature selection process; rather, they were intuitively chosen based on their potential to most reduce the entropy of the language most effectively. The model has been used to produce a single melody only. It is generated by random sampling and an existing melody is given as context to condition the generation of the first event. However, there is no discussion of the quality of the melody generated. Furthermore, no evaluation method is specified. Thus it can be assumed that this melody is the best result that stands out from many less convincing results.

2.5.2 Hidden Markov Models

A hidden Markov model is a statistical model where observed events are generated based on underlying hidden states. Whenever an event is sampled, the model transitions to a new state based on the transition probabilities. As a result, several sequences of hidden states can produce the same sequence of events. The decoding step is the process by which, from an observed sequence of events, the most likely sequence of hidden states is found. This process can be done by Viterbi decoding [Con03], a dynamic programming algorithm. This relationship between hidden states and observed events can be used in music to model the relationships between different events on the musical surface. For example, Farbood and Schoner [FS01] used a counterpoint line as observed events. Another counterpoint line is sampled from the hidden states using the Viterbi algorithm. This line is the one that is the most likely to occur according to the observed events. The disadvantage of this technique is that Viterbi decoding is often not easily applicable to complex powerful models [Con03].

2.5.3 Stochastic Sampling

Stochastic sampling includes methods such as *Gibbs sampling* and *Metropolis sampling*. The algorithm starts from an existing sequence s which has a probability $P_m(s)$. The methods stochastically modify the initial sequence so that the resulting sequence is unbiased with respect to the distribution $P_m()$. These methods are not greedy as the simple random walk is, since we perform multiple substitutions. However, they still can fall in a local minima, where valid substitutions do not improve the overall probability of the sequence.

Pearce investigated the use of metropolis sampling with a complex statistical model (IDyOM) to generate melodies. It is expected that metropolis sampling generates melodies that are more representative of the capacity of the model in comparison with random sampling. The experiment aimed to answer the question of whether the internal finite-state grammar acquired through statistical learning can be used effectively for composition. Based on the ratings of stylistic success given by expert judges, it is concluded that metropolis sampling using a model of melodic expectancy is incapable of satisfying the objective. Indeed, melodies generated by the system are rated lower than those from the dataset.

3

IDyOM Model Reimplementation

This chapter presents the various components of the model of melodic expectancy developed in this research. The model is a reimplementation of the IDyOM model [Pea05] in the Python programming language. We focus here on the theory behind the implementation so that it can be reproduced, not on how the Python code works since it will be subject to changes in the future. One can find the code, including documentation and instructions on how to run the program, in this repository [1]. We explain which mechanisms are needed in the learning phase and the testing phase.

3.1 Learning Phase

The learning phase consists of capturing the regularities present in a corpus of melodies. The melodies in the corpus vary in size. A melody of size n must be representable as a sequence of discrete events (e.g. pitches). We represent the sequence as an n-gram - a tuple of events of size n - such that $e_1^n = (e_1, ..., e_n)$. The regularities of the corpus are stored in the context database.

3.1.1 The Context Database

The context database captures the regularities of the corpus. It stores the frequencies of various n-grams encountered in the melodies. This allows the model to learn patterns

[1]https://github.com/lissenko/PyIDyOM

and predict future events based on past sequences. The context database can be seen as a variable order Markov model , where each order corresponds to the length of the n-gram. This means we capture dependencies between events with context that vary in size. When a sequence is encountered for the first time, it is added to the database with an initial frequency of 1. If the sequence already exists in the database, its frequency count is incremented by 1.

An example of construction of a context database is given in Table 3.1. The database is constructed from a single sequence of event GGDBAGGABA and has a maximum order $n = 3$.

(A) : 3	(AB) : 1	(ABA) : 1
(G) : 4	(AG) : 1	(AGG) : 1
(D) : 1	(GA) : 1	(GAB) : 1
(B) : 2	(GG) : 2	(GAB) : 1
	(GD) : 1	(GGA) : 1
	(DB) : 1	(GDB) : 1
	(BA) : 2	(DBA) : 1
		(BAG) : 1

Table 3.1: Small context database after incorporation of the sequence GGDBAGGABA

3.1.2 Different Views on the Musical Surface

The context database described above is constructed for a single type of discrete event of the musical surface (e.g. pitches). However, in practice, the model stores different contextual databases corresponding to different event types (e.g. a pitch database and a duration database). This takes advantage of the multidimensional nature of a melodic sequence. Music is multidimensional, since it can be broken down into different types of features, and is, therefore, more complex to study than language. Using different contextual databases allows more accurate predictions to be made, taking advantage of the complex nature of music. This will be explained in more detail when defining the multiple viewpoint system in Section 3.2.3.

3.1.3 Data Augmentation

We present a data augmentation technique inspired by the implementation of the IDyOMpy model [Gui18]. The idea is to expand the corpus of melodic sequences by transposing each melody up by 6 semitones and down by 5 semitones, thus covering all 12 tones. This can be formulated as follows: Given a sequence of pitches $\{p_1, p_2, \ldots, p_n\}$, we generate new sequences $\{p_1 + k, p_2 + k, \ldots, p_n + k\}$ for $k \in \{-5, -4, \ldots, 5, 6\}$.

It is hypothesized that by increasing available data, the model will access a greater number of transitions between contexts and events, thereby improving its predictive accuracy. We will test the effectiveness of this augmentation technique in Chapter 5.

One might wonder why we transpose the melodies up and down, instead of transposing every melody in the training and test sets to the same key (e.g., C major or C minor). However, by doing so, we would assume processes of human musical cognition; it is not necessarily the case that we interpret one key as another. Just like in visual perception, where we do not react to green in the same way as we do to red, our response to different musical keys may vary.

3.2 Testing Phase

During the testing phase, the system receives a set of melodies as input. Each melody can be analyzed as a sequence of discrete events e_1^n, where n is the length of the sequence and e_i is the event at the i-th position. For each sequence e_1^n, the model determines the likelihood of the melody $P(e_1^n)$ such that

$$P(e_1^n) = P(e_1) \cdot P(e_2 \mid e_1) \cdot P(e_3 \mid e_1^2) \cdot \cdots \cdot P(e_n \mid e_1^{n-1}), \tag{3.1}$$

where e_1^k denotes the subsequence of events from the first event up to the k-th event.

Therefore, it is required to make predictions for each event e_i given its preceding context e_1^{i-1}. This involves estimating the conditional probability $P(e_i \mid e_1^{i-1})$ for each event in the sequence. The information content of an individual prediction represents the surprise or unexpectedness of the event given its context. The sequence of unexpectedness for each event in a melody models the signal of surprise experienced by a listener as they hear the melody.

The system can make predictions at various levels, then integrated into a single final prediction. We will explain these different levels of predictions using a bottom-up approach, starting with the prediction for an event with a single attribute on the musical surface and concluding with predictions for notes that encompass multiple attributes of the surface.

3.2.1 Predictions

In practice, one does not compute the likelihood of an event given its context starting from the first event of the sequence. Instead, for computational efficiency, it is preferable to fix an order bound n where the context consists of the preceding $n-1$ events. Therefore, a prediction of order n, $P(e_i \mid e_{(i-n)+1}^{i-1})$, estimates the likelihood of the event e_i given the context $e_{(i-n)+1}^{i-1}$. Having a way of making predictions, one can find the distribution of events at position i given the context $e_{(i-n)+1}^{i-1}$ as $\{P(e \mid e_{(i-n)+1}^{i-1})\}_{e \in A}$ with A being the alphabet of every possible discrete event. From a prediction, one can derive the unexpectedness or surprisal of the event using the information content. The information content, measured in bits, indicates the amount of information required to encode the symbol; if it is high, it was not expected in the given context.

$$h(e_i \mid e_{(i-n)+1}^{i-1}) = -\log_2 P(e_i \mid e_{(i-n)+1}^{i-1}) \tag{3.2}$$

In addition to calculating the information content of individual predictions, it is also possible to compute the entropy of the model given the context. The entropy quantifies the average uncertainty or unpredictability of the system's predictions over the set of possible events. For a given context $e_{(i-n)+1}^{i-1}$, the entropy $H(e \mid e_{(i-n)+1}^{i-1})$ is defined as:

$$H(e_i \mid e_{(i-n)+1}^{i-1}) = -\sum_{e_i \in A} P(e_i \mid e_{(i-n)+1}^{i-1}) \log_2 P(e_i \mid e_{(i-n)+1}^{i-1}) \tag{3.3}$$

where A is the alphabet of possible discrete events. This formula calculates the expected value of the information content of all potential events e given the context

$e_{(i-n)+1}^{i-1}$. A higher entropy value indicates greater uncertainty in the predictions, meaning that the system faces a wider range of possible events, while a lower entropy value suggests more predictability.

3.2.2 Prediction Estimation

We will now present methods to estimate predictions $P(e_i \mid e_{(i-n)+1})$. At this level of prediction, predictions are made for univariate events, a single view on the musical surface $\tau \in \xi$. The maximum likelihood method is the naive method and the PPM algorithm tackles the problems of the maximum likelihood method.

3.2.2.1 Maximum Likelihood

The maximum likelihood estimation (MLE) method is used to predict the probability of an event e_i given a context $e_{(i-n)+1}^{i-1}$. The likelihood is computed by considering the frequency of the event e_i occurring after the given context $e_{(i-n)+1}^{i-1}$ and normalizing it by the total number of occurrences of all possible events following the same context. This is formally expressed as:

$$P(e_i \mid e_{(i-n)+1}^{i-1}) = \frac{c(e_i \mid e_{(i-n)+1}^{i-1})}{\sum_{e \in A} c(e \mid e_{(i-n)+1}^{i-1})} \tag{3.4}$$

where $c(e_i \mid e_{(i-n)+1}^{i-1})$ is the count of occurrences of the event e_i given the context $e_{(i-n)+1}^{i-1}$, and the denominator sums the counts of all possible events e in the event space A given the same context.

The choice of the context size/order n is important, as it affects the prediction accuracy. The choice of n depends on the available data and the nature of the event space. Larger n values can capture more detailed contextual information but require more data to provide reliable frequency estimates. Higher-order predictions, made with a large value for the order n, are very specific to the context but might be statistically unreliable since longer contexts appear less frequently and do not have many possible continuations. On the other hand, lower-order predictions, made with a small value for the order n, are more statistically robust since they appear more often with many possible continuations.

An issue with maximum likelihood estimation is the zero-frequency problem. If a context is not observed in the context database or an event has not been observed after a given context, the likelihood of such a sequence will be 0, as $c(e_i \mid e_{(i-n)+1}^{i-1}) = 0$. This zero likelihood is problematic because it results in an infinite surprisal, given that surprisal is inversely proportional to likelihood. Additionally, the information content and entropy formulas require using the logarithm of the probability, which is undefined for a value of 0. Therefore, we need a method that provides a non-zero value for each sequence, ensuring that even highly improbable events have a positive likelihood.

3.2.2.2 PPM Algorithm

The PPM (Prediction by Partial Matching) algorithm [CW03a] addresses the problems of the maximum likelihood method by using a smoothing technique to adjust the maximum likelihood estimates and generate probabilities for unseen sequences. When an unseen sequence is encountered, the algorithm combines distributions estimated by lower orders, which do not have zero probabilities. This can be expressed by the *back-off smoothing* framework as follows:

$$P(e_i \mid e_{(i-n)+1}^{i-1}) = \begin{cases} \alpha(e_i \mid e_{(i-n)+1}^{i-1}), & \text{if } c(e_i \mid e_{(i-n)+1}^{i-1}) > 0 \\ \gamma(e_i \mid e_{(i-n)+1}^{i-1}) \cdot P(e_i \mid e_{(i-n)+2}^{i-1}), & \text{if } c(e_i \mid e_{(i-n)+1}^{i-1}) = 0 \end{cases} \tag{3.5}$$

For a given context $e_{(i-n)+1}^{i-1}$, if the event e_i occurs with a non-zero frequency, then the estimation $\alpha(e_i \mid e_{(i-n)+1}^{i-1})$ is used. Otherwise, one recursively backs off to a scaled version of the estimate $P(e_i \mid e_{(i-n)+2}^{i-1})$, where the scaling factor or escape probability $\gamma(e_i \mid e_{(i-n)+1}^{i-1})$ is chosen so that the distribution sums to unity: $\sum_{e \in A} P(e_i \mid e_{(i-n)+1}^{i-1}) = 1$. Recursion stops when a lower-order estimate is non-zero. If all lower-order estimates are zero, a uniform distribution over the event alphabet is used.

Another technique is the *interpolated smoothing* method, which systematically combines the estimation of scaled distributions of lower orders even if the sequence is encountered in the context database:

$$P(e_i \mid e_{(i-n)+1}^{i-1}) = \alpha(e_i \mid e_{(i-n)+1}^{i-1}) + \gamma(e_i \mid e_{(i-n)+1}^{i-1}) \cdot P(e_i \mid e_{(i-n)+2}^{i-1}) \tag{3.6}$$

This technique has been shown to yield better results than backoff smoothing in language modeling [CG96] and is the technique we use in our model, as it was demonstrated by Pearce to give better results in melodic expectancy modeling [Pea05].

One still needs to determine how are defined the functions $\alpha()$ and $\gamma()$. Here, we define the methods C and AX that are used in our implementation. We use method C with the LTM and method AX with the STM since it was proven to work best in the original IDyOM model [Pea05]. However, there are many other escape strategies [Pea05].

Method C [Mof90]: This method assigns a frequency count of $t(e_{(i-n)+1}^{i-1})$ to events that are novel in the current context. The term $t(e_{(i-n)+1}^{i-1})$ in the denominator ensures that the escape probability increases as more distinct events are observed in the context. It is defined as:

$$\gamma(e_i \mid e_{(i-n)+1}^{i-1}) = \frac{t(e_{(i-n)+1}^{i-1})}{\sum_{e \in A} c(e \mid e_{(i-n)+1}^{i-1}) + t(e_{(i-n)+1}^{i-1})} \tag{3.7}$$

$$\alpha(e_i \mid e_{(i-n)+1}^{i-1}) = \frac{c(e_i \mid e_{(i-n)+1}^{i-1})}{\sum_{e \in A} c(e \mid e_{(i-n)+1}^{i-1}) + t(e_{(i-n)+1}^{i-1})} \tag{3.8}$$

where $t(e_{(i-n)+1}^{i-1})$ denotes the total number of different types of events, members of A, that have occurred with non-zero frequency in the context $e_{(i-n)+1}^{i-1}$.

Method AX [MNW98]: This method is an approximation of method P [WB91], which is based on the assumption that novel events occur according to a Poisson process model. It also addresses the computational issues found in method X, a prior approximation of method P. Method AX is defined as:

$$\gamma(e_i \mid e_{(i-n)+1}^{i-1}) = \frac{t_1(e_{(i-n)+1}^{i-1}) + 1}{\sum_{e \in A} c(e \mid e_{(i-n)+1}^{i-1}) + t_1(e_{(i-n)+1}^{i-1}) + 1} \tag{3.9}$$

$$\alpha(e_i \mid e_{(i-n)+1}^{i-1}) = \frac{c(e_{(i-n)+1}^{i-1})}{\sum_{e \in A} c(e \mid e_{(i-n)+1}^{i-1}) + t_1(e_{(i-n)+1}^{i-1}) + 1} \tag{3.10}$$

where $t_k(e_i^j)$ denotes the total number of different types of event, members of A, that have occurred exactly k times in context e_i^j.

3.2.2.3 PPM* Algorithm

The PPM* algorithm addresses the limitations of fixed-order models by dynamically selecting the most appropriate context length for predictions, thereby avoiding the need to impose a fixed order on the model.

One key aspect of PPM* is its policy for selecting the order of the context: A context e_i^j is considered deterministic if it makes only one prediction, i.e., $t(e_i^j) = 1$. For such contexts, the observation of new events is very infrequent. Consequently, the entropy of the distributions estimated for deterministic contexts tends to be lower than for non-deterministic contexts. Since an event will be observed at least as many times in lower-order distributions, the distribution of the deterministic context with the lowest entropy is considered. The policy uses this observation to select the context length: it selects the smallest deterministic matching context, or if none exists, it uses the longest matching context available.

However, this approach requires recording contexts of all sizes in the context database, which poses an issue regarding space and time complexity. Therefore we limit the size of the sequences to a fixed integer n. It is worth noting that a clever way of representing the data using Suffix-tree representations is used in the original IDyOM model [Pea05]. It allows for the context database to store sequences that are unbounded in size. We do not implement this feature in the model since we found that using an appropriate maximum order, prediction quality is equivalent.

3.2.3 Multiple Viewpoints System

The previous section described methods for making predictions for an event e_i given a context $e_{(i-n)+1}^{i-1}$. We assumed that the events were univariate, representing a single view of the musical surface. However, we can use the richness of the musical language to make predictions for different types of attributes of events, leading to a multivariate event representation. This collection of views on the musical surface is defined by Conklin as a multiple viewpoint system [WC90], and it is described in this section.

3.2.3.1 Multiple Viewpoint Representations of Music

The musical surface of a multiple viewpoint system is a sequence of musical events that correspond to notes of a score. Each note can be decomposed into a set of descriptive variables or basic attributes that draw their values from a finite alphabet. Each attribute describes an abstract property of an event and is associated with a type τ. The type specifies the property of the attribute (for example, a type that describes the pitch or a type that describes the duration). A typed attribute τ is associated with the following elements:

- The syntactic domain of τ, written $[\tau]$, is the set of all possible values for the type τ (the alphabet of the type). For example, for a type being the chromatic pitch of an event, $[\tau]$ is $\{60, \cdots, 72\}$ following the MIDI standard.

- The semantic domain of τ, written $[[\tau]]$ is the set of associated meanings to the syntactic domain $[\tau]$. For a type being the chromatic pitch, $[[\tau]]$ is $\{\mathrm{C4}, \mathrm{C\sharp4}, \ldots, \mathrm{B4}, \mathrm{C5}\}$.

- The semantic interpretation of τ, written $[[\cdot]]_\tau : [\tau] \to [[\tau]]$, returns the meaning for an element of type τ. For example for the pitch 60, $[[60]]_{pitch} = C_4$.

- The viewpoint that models a type τ is written $\Psi_\tau : \xi^* \rightharpoonup [\tau]$. It is a partial function which maps sequences of events to elements of type τ (see Table 3.1), where ξ is the event space:

$$\xi = [\tau_1] \times [\tau_2] \times \cdots \times [\tau_n] \tag{3.11}$$

- The type set domain of τ, written $\langle\tau\rangle \subseteq \{\tau_1, \cdots, \tau_n\}$, are the basic viewpoints used to derive the viewpoint. It specifies which basic types the viewpoint is capable of predicting.

Therefore an event $e \in \xi$ is an instantiation of the basic types $\tau_1, \ldots, \tau_n$ and is an n-tuple in the event space. A multiple viewpoint system is a collection of viewpoints. We define the different viewpoint classes:

- **Basic Viewpoints**: Basic viewpoints model fundamental types. Ψ_τ is a projection function. An example of a basic type represents the duration of an event in terms of MIDI ticks.

- **Derived Viewpoints**: A derived type is a type that is not present in the event space ξ but is derived from one or more basic types. Ψ_τ is a selector function that takes a sequence of events and returns the appropriate attribute value. A viewpoint that models a derived type is a derived viewpoint. The function Ψ_τ is partial and might be undefined ($\perp$). The type set $\langle\tau\rangle$ are all the viewpoints used to derive the viewpoint.

- **Linked Viewpoints**: A linked viewpoint models a product type. A product type allows for the representation of interactions between individual types. A product type $\tau = \tau_1 \bigotimes \cdots \bigotimes \tau_n$ has the following properties:

  $$[\tau] = [\tau_1] \times \cdots \times [\tau_n]$$

  $$[\![\tau]\!] = [\![\tau_1]\!] \text{ and } \cdots \text{ and } [\![\tau_n]\!]$$

  $$\Psi_\tau = \begin{cases} \perp & \text{if } \Psi_{\tau_i}(e_1^j) \text{ is undefined for any } i \in \{1,\ldots,n\} \\ \Psi_{\tau_1}(e_1^j),\ldots,\Psi_{\tau_n}(e_1^j) & \text{otherwise.} \end{cases}$$

  $$\langle\tau\rangle = \bigcup_{k=1}^{n} \langle\tau_k\rangle$$

  A linked viewpoint allows the representation language to express disjunctions of conjunctions of attribute values, instead of simple disjunctions of attribute values.

- **Test Viewpoints**: A test viewpoint models a boolean-valued attribute type. It performs a test to identify specific locations in sequences. For example, a type set to true wherever the event is the first in a bar.

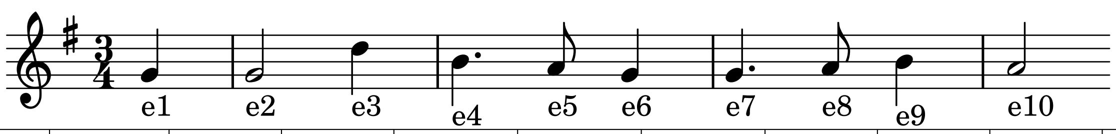
- **Threaded Viewpoints**: Types whose values are only defined at certain points in a piece. This models a base viewpoint at different temporal metrics - where a test viewpoint returns true, or undefined otherwise. The base viewpoint can be basic, derived, or linked. This emphasizes the importance of structures from metric and phrasing groups in music.

3.2.3.2 System Viewpoints

In this research, we utilize types described in the PhD dissertation of Pearce [Pea05]. All types are implemented in Python, except those related to phrase structures, as we employ MIDI encoding instead of **kern encoding. The MIDI encoding does not provide information on phrase structure. Each type is detailed in Table 3.2. In our approach to linked types, we utilize every possible combination of two types. The original IDyOM framework was constrained to specific linked types, which were assumed to yield accurate predictions primarily due to computational limitations. However, given the advancements in computational resources available at the time of this writing, we can now explore all potential combinations of types. Formally, if we define the set of types as $\mathscr{T} = \{\tau_1, \tau_2, \ldots, \tau_n\}$, the linked types can be represented as:

$$\tau_{\text{linked}} = \bigcup_{i \neq j} \left(\tau_i \otimes \tau_j\right)$$

An example of a melody represented as viewpoint sequences is provided in Figure 3.1.



	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
cpitch	67	67	74	71	69	67	67	69	71	69
onset	0	24	72	96	132	144	168	204	216	240
dur	24	48	24	36	12	24	36	12	24	48
cpint	$\perp$	0	7	-3	-2	-2	0	2	2	-2
ioi	$\perp$	24	48	24	36	12	24	36	12	24
contour	$\perp$	0	1	-1	-1	-1	0	1	1	-1
thrbar	$\perp$	0	$\perp$	-3	$\perp$	$\perp$	0	$\perp$	$\perp$	-2
cpint ⊗ dur	($\perp$, 24)	(0, 48)	(7, 24)	(-3, 36)	(-2, 12)	(-2, 24)	(0, 36)	(2, 12)	(2, 24)	(-2, 48)

Figure 3.1: The melody represented as viewpoint sequences

By having different viewpoints that model different types on the musical surface, it is also necessary to combine the predictions of these models. This process is described in the following section.

τ	$[[\cdot]]_\tau$	$[\tau]$	$\langle\tau\rangle$
onset	event onset time	$\mathbb{Z}^*$	{onset}
deltast	rest duration	$\mathbb{Z}^*$	{deltast}
dur	event duration	$\mathbb{Z}^+$	{dur}
barlength	bar length	$\mathbb{Z}^*$	{barlength}
pulses	metric pulses	$\mathbb{Z}^*$	{pulses}
cpitch	chromatic pitch	$\mathbb{Z}$	{cpitch}
keysig	key signature	$\{-7,-6,\dots,6,7\}$	{keysig}
mode	mode	$\{0,9\}$	{mode}
cpitch-class	pitch class	$\{0,\dots,11\}$	{cpitch}
cpint	pitch interval	$\mathbb{Z}$	{cpitch}
cpcint	pitch class interval	$\{0,\dots,11\}$	{cpitch}
contour	pitch contour	$\{-1,0,1\}$	{cpitch}
referent	referent or tonic	$\{0,\dots,11\}$	{keysig}
inscale	(not) in scale	$\{T,F\}$	{cpitch}
cpintfref	cpint from tonic	$\{0,\dots,11\}$	{cpitch}
cpintfip	cpint from first in piece	[cpint]	{cpitch}
cpintfib	cpint from first in bar	[cpint]	{cpitch}
tessitura	deviation from the mean pitch	$\{-1,0,1\}$	{cpitch}
posinbar	event position in bar	$\mathbb{Z}^*$	{onset}
ioi	inter-onset interval	$\mathbb{Z}^+$	{onset}
dur-ratio	duration ratio	$\mathbb{Q}^+$	{dur}
tactus	(not) on tactus pulse	$\{T,F\}$	{onset}
fib	(not) first in bar	$\{T,F\}$	{onset}
thrtactus	cpint at metric pulses	[cpint] $\times\ \mathbb{Z}^+$	{cpitch,onset}
thrbar	cpint at first in bar	[cpint] $\times\ \mathbb{Z}^+$	{cpitch,onset}

Table 3.2: Basic, derived, test, and threaded attribute types used in this search. They are identical to those used in IDyOM, except phrase types.

3.2.4 Combining Distributions

In the previous section, we demonstrated the computation of discrete probability distributions produced by various viewpoint models. To predict multivariate events, it is necessary to combine these distributions. This combination process involves addressing two main challenges: converting the distributions over the basic event space and merging the individual predictions effectively.

3.2.4.1 Conversion over the Basic Event Space

The distribution returned by the viewpoint model m_τ, which models the type τ, is given over $[\tau]$. However, to combine this distribution with those of other viewpoints, it is necessary to convert it over the basic event space ξ. To achieve this, we define the function Ψ'_τ, which is the inverse process of representing a viewpoint sequence. This function maps elements of $[\tau]$ onto elements of the basic type $[\tau_b]$.

$$\Psi'_\tau : \xi^* \times [\tau] \rightarrow 2^{[\tau_b]} \tag{3.12}$$

In practice, this function is implemented by collecting information during the learning phase. When a viewpoint sequence different from a basic viewpoint sequence is encountered during training, we create a set of sequences corresponding to their basic types. With this information, it becomes possible to map an element of $[\tau]$ onto multiple elements of $[\tau_b]$. Since the mapping is many-to-one, the probability mass predicted for an element by the model m_τ will be equally divided among the basic elements it is mapping.

3.2.4.2 Methods to Combine Distributions

In order to use predictions from n viewpoint models $M = \{m1, \ldots, m_n\}$ as a single prediction, we need a method for combining them. Different methods are presented here to compute $P(e)$, the combinations of probabilistic predictions of the M models for event e. The simplest method is to use an arithmetic mean such that:

$$P(e) = \frac{\sum_{m \in M} P_m(e)}{|M|} \tag{3.13}$$

The arithmetic mean does not consider differences in the performance/uncertainty of different models. Therefore Conklin proposes a weighted arithmetic mean [WC90]

$$P(e) = \frac{\sum_{m \in M} w_m P_m(e)}{|M|} \tag{3.14}$$

where the weights are calculated as

$$w_m = H_{\text{relative}}(p_m)^{-b} \tag{3.15}$$

the relative entropy of the model is given by

$$H_{\text{relative}}(P_m) = \begin{cases} H(P_m)/H_{\max}(P_m) & \text{if } H_{\max}(P_m) > 0 \\ 1 & \text{otherwise} \end{cases} \tag{3.16}$$

where the maximum entropy H_{max} for model m is the case where each symbol in the alphabet of event $[m]$ has an equal probability of occurring such that

$$H_{max}(P_m) = H_{max}([m]) = log_2|[m]| \tag{3.17}$$

and $b \in \mathbb{Z}^+$ is an exponential bias towards models with lower relative entropy. When $b = 0$, the arithmetic mean is equivalent to the unweighted arithmetic mean (Equation 3.13). We can understand this weighting by relative entropy by drawing a parallel with the predictive power of the system. From the relative entropy formula, we know that the predictive power of the model is minimized when $H_{\text{relative}} = 1$. Such a model will have random predictions, so we favor models with a relative entropy that tends towards 0, because the predictive power is much higher and the predictions more deterministic.

Pearce introduces the (weighted) geometric mean [Pea05], a novel method to combine the predictions of different statistical models:

$$p(e) = \frac{1}{R}\left(\prod_{m \in M} P_m(e)\right)^{\frac{1}{|M|}} \tag{3.18}$$

and the entropy-based weighted version with weights inspired the weighted arithmetic combination from Conklin (Equation 3.15)

$$p(e) = \frac{1}{R}\left(\prod_{m \in M} P_m(e)^{w_m}\right)^{\frac{1}{\sum m \in M w_m}} \tag{3.19}$$

where R is a normalization factor such that the final distribution over $[m]$ sums to unity. The motivation for a geometric mean instead of an arithmetic mean is that combining distributions through multiplication produces a less uniform distribution [Pea05]. As a result, the calculated distribution has a lower entropy with the geometric mean than the arithmetic mean, and therefore has a higher predictive power. If a model correctly assigns a low probability to an event $e \in [m]$, then the likelihood of that event in the combined distribution will also be low.

In our implementation, we use the weighted arithmetic mean of Equation 3.14. We found during our evaluation of the model (see Chapter 5) that it gives better results than the geometric, or non-weighted alternative.

3.2.5 The Long-and-Short-term Models

In this section, we discuss the integration of a long-term model (LTM) and a short-term model (STM). for predicting musical events. The primary aim of this technique is enhance predictive accuracy by utilizing two models that represent the different cognitive processes of listeners.

The architecture of the system, which displays the combinations of the different viewpoints and of the short term and long term models are summarized in Figure 3.2.

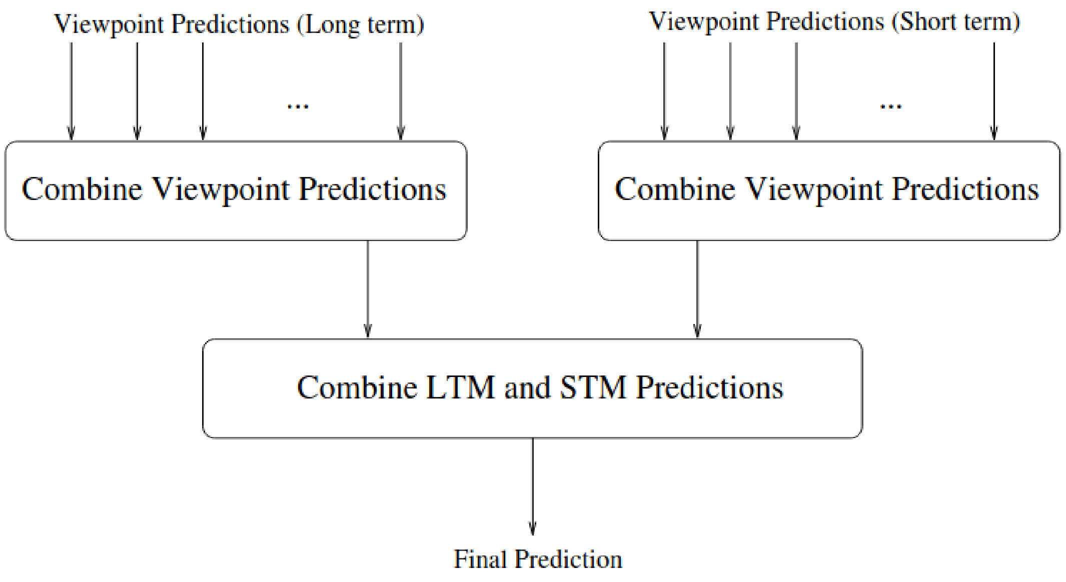

Viewpoint Predictions (Long term)
Viewpoint Predictions (Short term)
...
...
Combine Viewpoint Predictions
Combine Viewpoint Predictions
Combine LTM and STM Predictions
Final Prediction

Figure 3.2: The architecture of a multiple viewpoint system [Pea05]

3.2.5.1 Long-term Model

The long-term model (LTM) is trained on a musical corpus before making predictions. The data learned by the LTM is retained permanently, even after predicting a specific piece. The LTM models the cumulative knowledge that a listener acquires over a lifetime of exposure to a particular musical style.

3.2.5.2 Short-term Model

The short-term model (STM) is trained individually for each member of the test set, and the knowledge that has been learned is discarded before predicting a new member of the test set. It is constructed online for each composition. The STM aims to provide accurate predictions about recently encountered sequences specific to the predicted melody.

3.2.5.3 Combining Long-and-Short-term Models

To obtain a final prediction, the predictions of the LTM and STM must be combined. The techniques used to combine these predictions are the same as those used to combine the distributions of different viewpoints (section 3.2.4). However, it is not necessary to convert the LTM and STM distributions over the event space, as they are already compatible.

4

Temperature-Scaling for Melody Generation

This chapter presents the details of sequence generation using the IDyOM statistical model implemented in the previous chapter. We introduce the random sampling technique technique, a common method in sequence generation. To modify the traditional random sampling process, we incorporate temperature scaling [Guo+17] to reduce uncertainty or enhance confidence in the generation process.

The goal of sequence generation is to sample notes from the probability distribution provided by IDyOM. The resulting sequence of notes forms a melody intended to stylistically match the musical style of the melodies in the dataset that IDyOM was trained on. In this approach, we use a base melody as the foundation for the generation process. This narrows the problem of sampling notes to the problem of sampling pitches. The durations of notes are taken from the base melody. Thus, we sample only the pitches from the model, while the time and key signatures, as well as the durations of notes, are taken from the base melody.

The generated melodies are not intended to be final compositions for direct listening. Instead, they serve as a creative tool for composers, providing system-generated melodies that can inspire and trigger creativity. The temperature parameter allows composers to control the degree of exploration of the system to control the balance between creativity and stylistic consistency.

We will first describe the process of using a random walk for melody generation and then introduce the additional module of temperature scaling, which we insert into the traditional random walk generation process.

4.1 Random Walk for Melody Generation

Consider a random walk process where we sample musical notes from a given probability distribution to generate a melody. Considering the basic type τ that is the chromatic pitch, let $[\tau]$ denote the set of possible pitches to sample.

- **Step 0: Training**: Train the IDyOM model to obtain a probability distribution over the alphabet of pitches $[\tau]$ for each sequence of notes (context).

$$P(e_i | e_{(i-n)+1}^{i-1}) \quad \text{for} \quad e_i \in [\tau]$$

- **Step 1: Initialization**: Define the initial pitch $e_0 \in [\tau]$. It is chosen as the tonic of the key signature of the reference melody.
- **Step 2: Random Walk**: For each time step $t = 1, 2, \ldots, N$:
  1. Sample the next note e_t from the probability distribution P such that $e_t \sim P$.
  2. Append e_t to the melody sequence.

4.2 Temperature Scaling

Temperature scaling is a technique often used in Deep Learning to address the problem of confidence calibration [Guo+17]. Confidence calibration aims to produce probability estimates that accurately reflect the true likelihood of correctness. For example, if a classification model outputs a certain probability of a patient having a disease, this probability might not represent the true likelihood of the patient having the disease. Temperature scaling adjusts the probability vector returned by the model to reflect the true likelihood better.

However, our use of temperature scaling in melody generation has a different purpose. We aim to adjust the distribution over pitches returned by the model to modify its entropy. Increasing the entropy makes the system more uncertain about the note to sample, leading to sequences with more exploration beyond the space of stylistically successful melodies. Conversely, decreasing the entropy makes the system more confident about the note to sample, resulting in greedier choices and a sequence that is more likely to be stylistically successful.

4.2.1 Implementation

Temperature scaling is applied to a probability distribution P over a set of pitches. Given a temperature parameter $T > 0$, the scaled probability P'_i for pitch i is calculated as follows:

$$P'_i = \frac{\exp(\frac{\log(P_i)}{T})}{\sum_j \exp(\frac{\log(P_j)}{T})}$$

When $T = 1$, the distribution remains unchanged. When $T > 1$, the distribution becomes softer (higher entropy), spreading the probabilities more evenly across pitches. When $T < 1$, the distribution becomes sharper (lower entropy), concentrating the probabilities more on the pitches with higher initial probability.

To understand the temperature scaling process more clearly, we break down each component of the equation:

- $\log(P_i)$: The natural logarithm of the original probability of pitch i. Working with logits is necessary because their values are passed to the exponential function, the inverse of the natural logarithm. Without using logits, the original probability distribution would not be correctly recalculated when $T = 1$. Since probabilities are between 0 and 1, the logits are negative.
- $\frac{\log(P_i)}{T}$: The scaled logarithm of the probability. Lower temperatures result in lower scaled logits, concentrating the probability distribution.
- $\exp(\frac{\log(P_i)}{T})$: The exponentiation of the scaled logarithm, converting it back to a probability-like value. As the value passed to the logarithm increases, the corresponding exponential value approaches 0.

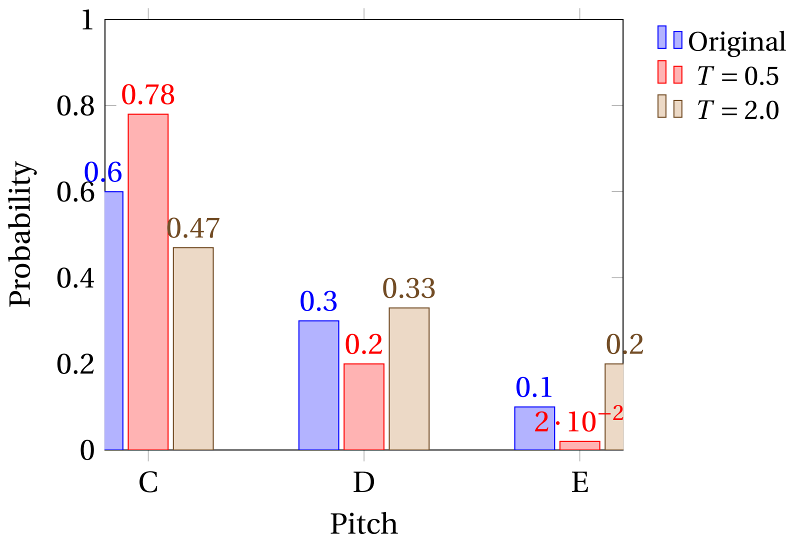
- $\sum_j \exp(\frac{\log(P_j)}{T})$: The normalization term, ensures that the scaled probabilities sum to 1. This is the sum of the exponentiated scaled logarithms for all pitches j.

4.2.2 Example

Consider probability distribution over three pitches: C, D, and E.

$$P = \{\mathrm{C} : 0.6, \mathrm{D} : 0.3, \mathrm{E} : 0.1\}$$

We apply temperature scaling with $T = 0.5$ (decreasing entropy) and $T = 2.0$ (increasing entropy). Figure 4.2.2 illustrates how the probability distribution changes with different values of T.



5

Methodology and Results

This chapter presents the methodologies, experiments, and results obtained from our implementation of IDyOM. The chapter is organized into three sections:

1. The first section describes the dataset used in each experiment.
2. The second section details the experiments conducted to evaluate the IDyOM implementation in terms of melodic expectancy.
3. The third section showcases an extension of our implementation for melody generation, where we assess the effectiveness of temperature scaling combined with random walk in producing stylistically successful melodies.

5.1 The Dataset

The dataset for the following experiments consists of 371 chorales available in the `**kern` format [Hur97] (see `https://kern.humdrum.org/`). These chorales were collected after J.S. Bach's death by his son, C.P.E. Bach, and completed by Kirnberger, a student of J.S. Bach. They were subsequently ordered by Breitkopf & Härtel.

5.1.1 Data Preprocessing

The IDyOM implementation we developed works for melodies (1 voice) and the dataset contains 4-part chorales (4 voices). Therefore, we need to select a single voice from the four voices. We chose the upper voice (the soprano voice) for this purpose, using the `extract` command-line program from the Humdrum Toolkit. Furthermore, since the implementation uses data in MIDI format, it is necessary to convert from `**kern` to MIDI. We used the `hum2mid` command-line program from the Humdrum Toolkit for this conversion. The bash script used for preprocessing is available in Appendix B.

5.2 Modelling Melodic Expectancy

To evaluate the performance of the IDyOM model described in Chapter 3, we will analyze three systems developed using this model:

- **System A** is designed to minimize model uncertainty over the dataset, measured using cross-entropy. By selecting a set of features that minimizes cross-entropy, this system aims to achieve high performance. Cross entropy serves as a measure of the model's average uncertainty when predicting a given sequence of events.
- **System B** is designed to fit the behavioral data on melodic expectations gathered from human subjects [MWJ92].
- **System C** also aims to fit the behavioral data on melodic expectations from human subjects [MWJ92]. However, this system uses the data augmentation technique described in Section 3.1.3.

Subsequently, we will compare the performance of our models with that of the original IDyOM and IDyOMpy models.

5.2.1 Experimental Methodology

The purpose of the following experiments is to evaluate how well the developed systems replicate human melodic expectations. For each system, we used a stepwise forward selection algorithm [AB95] to select an optimal set of viewpoints. Starting with an empty set, the algorithm iteratively considers every feature addition and then deletion, selecting features that provide the greatest improvement in the fitness function. The process concludes when no further addition or deletion improves the model. The viewpoint space (feature space) consists of the viewpoints described in Table 3.2, including linked viewpoints with a maximum link number of 2.

To compare the outputs of our systems to human melodic expectations, we use data from an experimental study conducted by Manzara [MWJ92]. In this experiment, participants estimated the entropy of two musical pieces (Chorale 61 and Chorale 151). Participants were given an initial amount of capital, S_{n-1}. At each position in the melody, they bet a proportion, p, on which of the 20 proposed chromatic notes was most likely to occur. This process continued until they correctly bet on the note from the original melody, and then they moved on to the next note. At each stage, the capital was adjusted by $20pS_{n-1}$, with correct bets increasing the capital more than incorrect bets. To derive an entropy value for a participant at any stage n, the formula $\mathrm{IC} = \log_2 20 - \log_2 S_n$ was used. By comparing these entropy profiles to those computed by our systems for Chorales 61 and 151, we can evaluate how well our models replicate human melodic expectations.

5.2.2 Experiment 1: System A

The results of viewpoint selection for System A are presented in Table 5.1. The table shows the cross-entropy obtained for the subset of viewpoints selected at each stage. Cross-entropies were computed using 10-fold cross-validation over the dataset. Throughout the stages, viewpoints were only added since their deletion did not reduce the cross-entropy. We achieved a cross-entropy of 1.908 bits/symbol, which is lower than the average cross entropy of 2.045 bits/symbol reported by Conklin & Witten [CW03b] and 1.953 bits/symbol reported by Pearce [Pea05]. Since the datasets and models used were similar but not identical, we cannot confidently conclude that our implementation minimizes the cross-entropy of perceived sequences better than other models. However, we hypothesize that our extended set of viewpoints, which comprehends every linked combination, was useful in reducing the cross-entropy.

Regarding the viewpoints selected in the experiment, we observe that all the selected viewpoints, except the first one (`cpitch`, which is a primitive type), were linked viewpoints. Furthermore, three of the selected viewpoints represent conjunctions of pitch and time-related attribute types. This indicates strong correlations between pitch structure and rhythmic structure within the corpus of chorale melodies.

The model with the final subset of selected viewpoints accounted for 34.25% of the

Stage	**Viewpoint Added**	**Viewpoint Dropped**	**H**
1	cpitch	-	2.031
2	cpitch⊗cpintfref	-	1.989
3	cpitch⊗cpintfip	-	1.976
4	cpitch⊗cpintfib	-	1.952
5	cpitch⊗keysig	-	1.950
6	cpitch⊗dur	-	1.917
7	cpitch⊗ioi	-	1.910
8	cpitch⊗dur-ratio	-	1.908

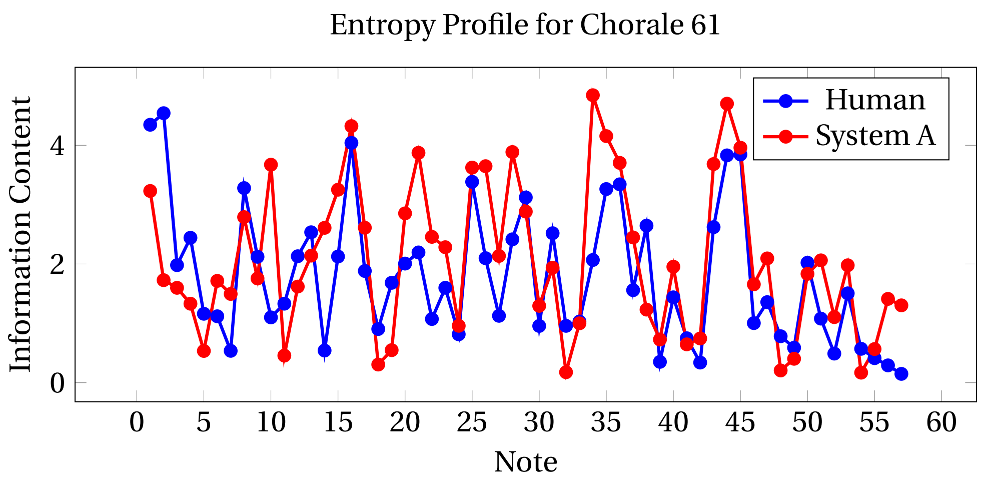
Table 5.1: Results of viewpoint selection for System A: reduce entropy over the Dataset.

variance in the mean uncertainty estimates reported by Manzara. Specifically, the model explained 43.2% of the variance in the mean uncertainty for Chorale 61, [$R^2_{\text{adj}} = 0.432, F(1,55) = 43.58, p < 0.0001$], and 25.3% for Chorale 151, [$R^2_{\text{adj}} = 0.253, F(1,27) = 10.50, p < 0.01$]. Profiles for both System A entropy and human entropy are shown in Figure 5.1 for Chorales 61 and 151.

5.2.3 Experiment 2: System B

The results of the viewpoint selection process for System B are presented in Table 5.2. This table displays the adjusted R-squared values for Chorale 61, Chorale 151, and their mean, obtained for the subset of viewpoints selected at each stage. The adjusted R-squared value, which ranges from 0 to 1, measures how well the variance in human uncertainty is explained by the model. Throughout the stages, viewpoints were only added, as their removal did not increase the adjusted R-squared value.

Regarding the viewpoints selected in the experiment, we observe, similar to System A, that all the selected viewpoints, except for the first one (`cpitch`, a primitive type), were linked viewpoints. Furthermore, three selected viewpoints represent conjunctions of pitch and time-related attribute types. This again indicates strong correlations between pitch structure and rhythmic structure within the corpus of chorale melodies. The model with the final subset of selected viewpoints accounted for 39.1% of the variance in the mean uncertainty estimates reported by Manzara. Specifically, the model explained 43.2% of the variance in the mean uncertainty for Chorale 61, [$R^2_{\text{adj}} =$



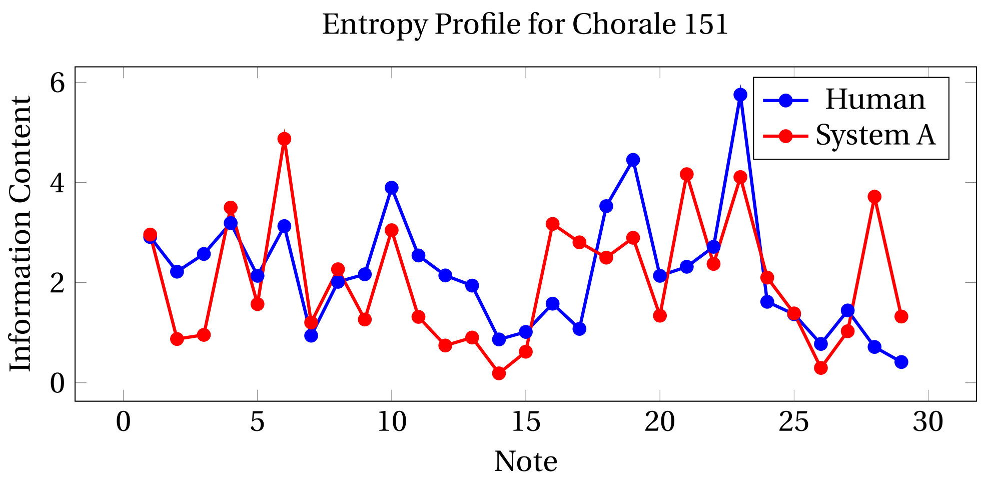
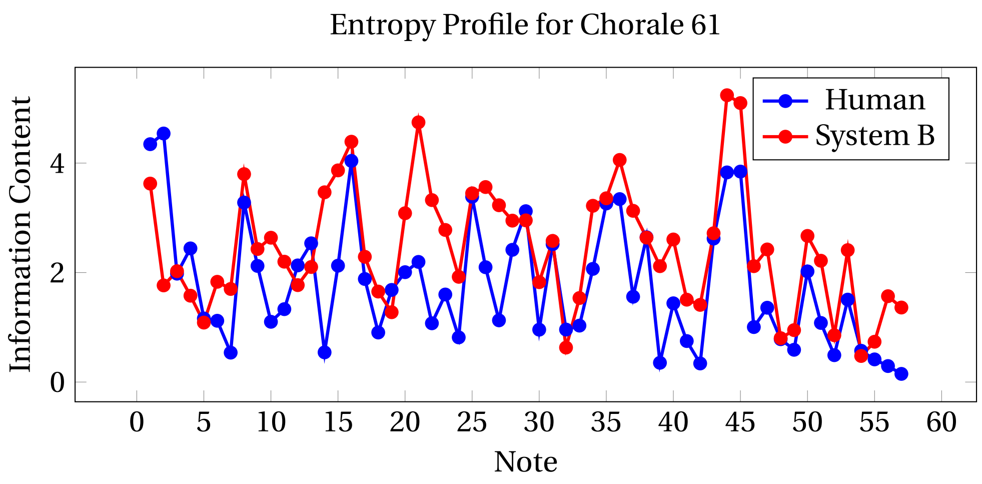


Figure 5.1: The entropy profiles for Chorales 61 and 151, averaged over the subjects of the Manzara experiment [MWJ92] and for System A.

Stage	**Viewpoint Added**	**Viewpoint Dropped**	**Chorale 61** R^2_{adj}	**Chorale 151** R^2_{adj}	**Mean** R^2_{adj}
1	cpitch	-	0.398	0.203	0.300
2	cpitch⊗dur	-	0.408	0.257	0.333
3	keysig⊗thrbar	-	0.426	0.304	0.365
4	mode⊗cpintfip	-	0.426	0.322	0.374
5	contour⊗referent	-	0.430	0.322	0.376
6	referent⊗cpintfip	-	0.430	0.348	0.389
7	cpintfref⊗fib	-	0.432	0.349	0.391

Table 5.2: Results of viewpoint selection for System B: fit entropy profiles of the Manzara experiment [MWJ92].

$0.432, F(1,55) = 43.63, p < 0.0001$], and 34.9% for Chorale 151, [$R^2_{\text{adj}} = 0.349, F(1,27) = 16.02, p < 0.001$]. Profiles for both System B entropy and human entropy are shown in Figure 5.2 for Chorales 61 and 151.



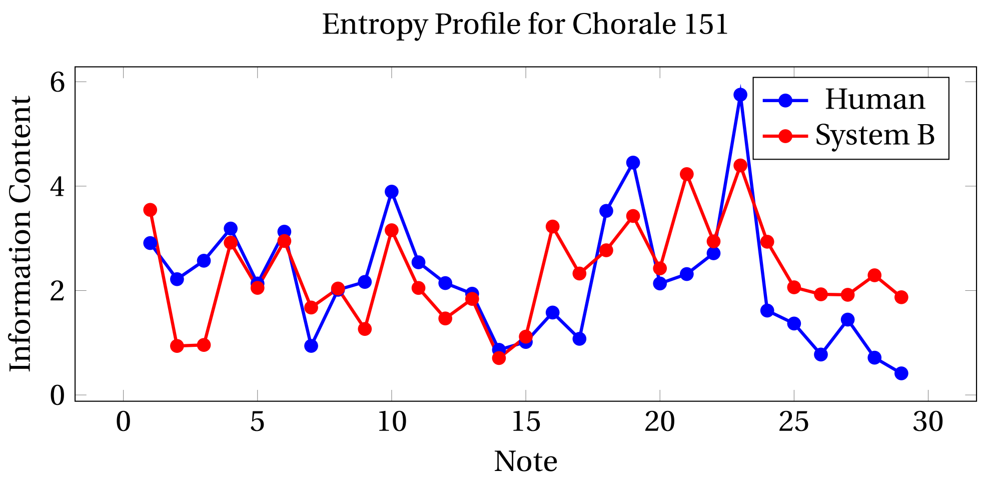


Figure 5.2: The entropy profiles for Chorales 61 and 151, averaged over the subjects of the Manzara experiment [MWJ92] and for System B.

5.2.4 Experiment 3: System C

The results of the viewpoint selection process for System C are presented in Table 5.3. This table displays the adjusted R-squared values for Chorale 61, Chorale 151, and their mean, obtained for the subset of viewpoints selected at each stage. The adjusted R-squared value, which ranges from 0 to 1, measures how well the variance in human uncertainty is explained by the model. Compared with System A and B, the last three stages are deletion of viewpoints.

Regarding the viewpoints selected in the experiment, we observe, similar to System A and B, that the majority of selected viewpoints are linked viewpoints. The three primitive types `cpitch`, `contour` and `cpintfib` were selected but both `cpitch` and `contour` were deleted during the last two stages. Furthermore, three of the selected viewpoints represent conjunctions of pitch and time-related attribute types. This again indicates strong correlations between pitch structure and rhythmic structure within the corpus of chorale melodies.

The model with the final subset of selected viewpoints accounted for 47.3% of the

Stage	Viewpoint Added	Viewpoint Dropped	Chorale 61 R^2_{adj}	Chorale 151 R^2_{adj}	Mean R^2_{adj}
1	cpitch	-	0.249	0.286	0.268
2	contour	-	0.331	0.293	0.312
3	cpintfib	-	0.331	0.302	0.316
4	cpitch⊗dur	-	0.395	0.344	0.370
5	cpitch⊗mode	-	0.413	0.348	0.381
6	cpitch⊗cpintfip	-	0.419	0.406	0.413
7	dur⊗cpintfref	-	0.428	0.409	0.419
8	dur⊗cpintfip	-	0.432	0.410	0.421
9	keysig⊗contour	-	0.441	0.435	0.438
10	contour⊗referent	-	0.446	0.456	0.451
11	-	keysig⊗contour	0.447	0.457	0.452
12	-	cpitch	0.451	0.480	0.466
13	-	contour	0.452	0.494	0.473

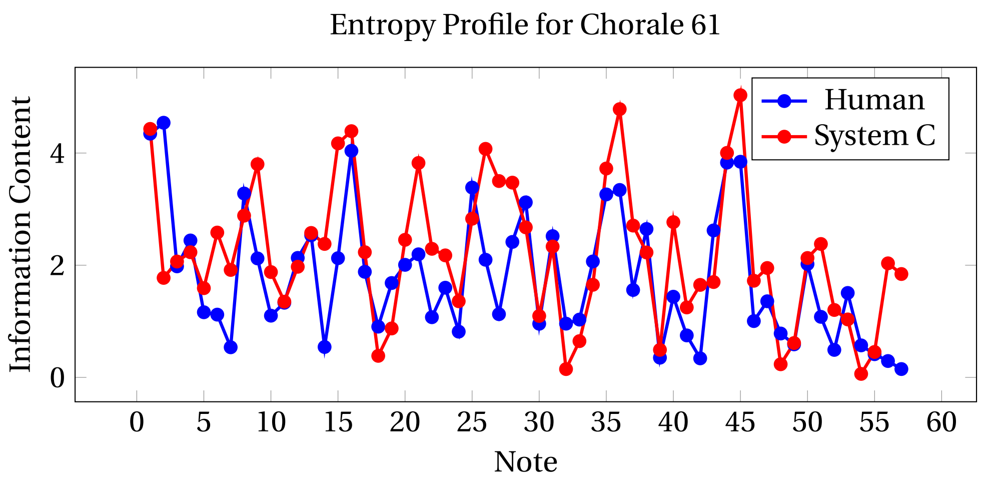
Table 5.3: Results of viewpoint selection for System C: fit entropy profiles of the Manzara experiment [MWJ92].

variance in the mean uncertainty estimates reported by Manzara. Specifically, the model explained 45.2% of the variance in the mean uncertainty for Chorale 61, [$R^2_{\text{adj}} = 0.452, F(1,55) = 47.18, p < 0.0001$], and 49.4% for Chorale 151, [$R^2_{\text{adj}} = 0.494, F(1,27) = 28.33, p < 0.0001$]. Profiles for both System C entropy and human entropy are shown in Figure 5.3 for Chorales 61 and 151.

5.2.5 Comparison Between Systems

We observe several commonalities between the viewpoints selections of System A, B and C. Indeed, `cpitch` is the only primitive viewpoint present in all three systems. All systems include viewpoints combining `cpitch` with other attributes, emphasizing the critical role of pitch in predicting chorale melodies. Each system selected linked viewpoints that combine pitch with time-related attributes.

We observe several differences between the systems. System A exclusively uses `cpitch`



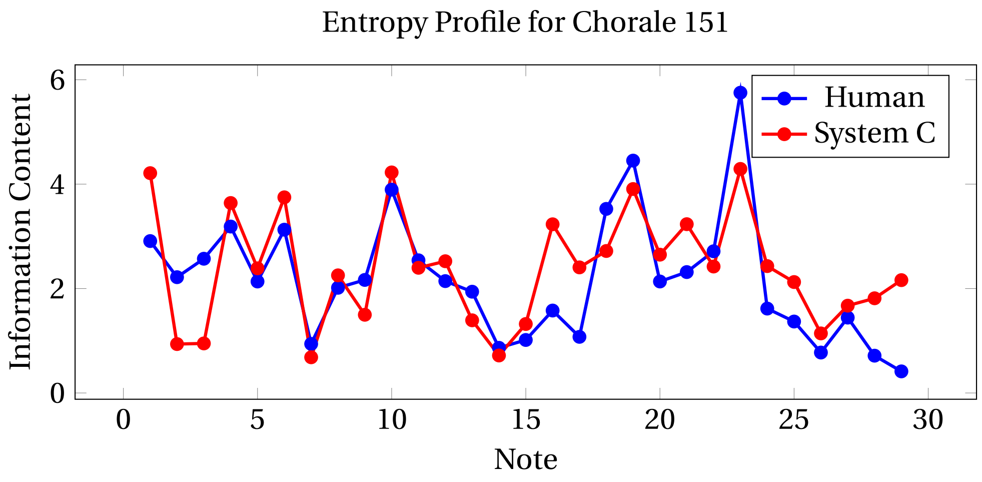


Figure 5.3: The entropy profiles for Chorales 61 and 151, averaged over the subjects of the Manzara experiment [MWJ92] and for System C.

in conjunction with other attributes, indicating either a low information content in the pitch or high information content in other types. Furthermore, System C, has 13 stages compared to 8 for System A and 7 for System B. This suggests that the augmentation technique implies a larger subset of viewpoints that can model human data effectively. Notably, System C is the only system with viewpoints that were deleted from the current set.

The focus of System A on `cpitch` combinations may offer a straightforward interpretation of the importance of pitch, but it also results in the worst fit to the behavioral data (see Table 5.4), potentially missing out on the complexity captured by Systems B and C. The variety of viewpoint selection in System C provides the best performance among the three systems (see Table 5.4), but also carries the risk of overfitting the entropy profiles derived from behavioral experiments. The variety of viewpoints in System C is a direct result of the augmentation technique applied. This technique was proven successful, enabling System C to outperform the non-augmented System B by 21%.

5.2.6 Comparison with State-of-the-Art Models

Having evaluated the performance of our implementation in fitting the uncertainty patterns reported by participants in a behavioral experiment, we can now compare our model with state-of-the-art models. Table 5.4 presents a comparison of the performances of Systems A, B, and C with the IDyOMpy [Gui18] model and the original IDyOM model [Pea05].

5.2.6.1 IDyOMpy

The main competitor to our implementation is IDyOMpy, another implementation of IDyOM in Python. IDyOMpy accounted for 12.5% of the variance in the mean uncertainty estimates reported by Manzara. Specifically, IDyOMpy explained 3.0% of the variance in the mean uncertainty for Chorale 61, $[R^2_{\text{adj}} = 0.030, F(1,55) = 2.730, p = 0.1]$, and 21.9% for Chorale 151, $[R^2_{\text{adj}} = 0.219, F(1,27) = 8.866, p < 0.01]$. The performance of IDyOMpy is significantly below that of any of the three systems A, B, and C. This represents a substantial advancement towards developing a robust statistical model of melodic expectancy in Python.

5.2.6.2 IDyOM

We compared the performances of the original IDyOM with the performances of our implementation. For this purpose, we ran the same process of feature selection than we did for our experiments to fit Manzara's data, with the same viewpoints set and the same dataset. The feature selection algorithm to fit Manzara's data is not provided is the IDyOM implementation, therefore we developed a Python wrapper around the original model for feature selection. The optimal subset of viewpoints returned by the algorithm is the following:

cpitch,	barlength⊗tessitura,	keysig⊗tessitura,
dur⊗cpitch-class,	cpitch⊗inscale,	mode⊗tessitura,

cpitch-class⊗posinbar,
referent⊗inscale,
referent⊗tessitura,
inscale⊗tessitura,
cpintfref⊗dur-ratio,
tessitura⊗posinbar,
tessitura⊗ioi,
tessitura⊗dur-ratio,
tessitura⊗tactus,
cpitch-class⊗tactus,
cpitch-class⊗fib,
inscale⊗posinbar,
inscale⊗tactus,
cpintfib⊗tessitura,
tessitura⊗fib,
tessitura.

With the selected subset of viewpoints, we found that IDyOM accounted for 56.4% of the variance in the mean uncertainty estimates. Specifically, the model explained 48.7% of the variance in the mean uncertainty for Chorale 61 [$R^2_{\text{adj}} = 0.487, F(1,55) = 54.11, p < 0.0001$], and 64.1% for Chorale 151 [$R^2_{\text{adj}} = 0.641, F(1,27) = 51.07, p < 0.0001$].

These results are 19.24% most performant than System C, which accounts for 47.3% of the variance in the mean uncertainty estimates. The discrepancy between these results may be due to several factors:

1. Our first observation concerns the frequency with which the tessitura viewpoint is selected in the viewpoint selection process with IDyOM. Specifically, tessitura is selected in half of the viewpoints, either as a linked type or primitive type. In contrast, with our system, tessitura is never selected. This leads us to hypothesize that the probability distribution returned by the tessitura viewpoint model over pitches is not relevant for predictions and is therefore discarded by our system. Given that tessitura has three possible values: $\{-1, 0, 1\}$, we further hypothesize that our method for converting this distribution into the alphabet of pitches is suboptimal and should be improved.

2. Since we use a dataset composed of melodies in the MIDI encoding, it contains less information by default than the **kern encoding used in the original implementation. For example, **kern contains information on the key signature of melodies when MIDI does not. Therefore we had to use a key signature detection algorithm from the `music21` Python library. The key signature algorithm should be accurate but might not capture the correct tonality of the Soprano voice that is originally part of a 4-part chorale. Correctly determining the key signature is essential to derive viewpoints information as the referent note, the interval from

the referent, and notes that are in scale.

3. Our implementation is not a perfect replication of the original implementation. The details published on the implementation date from 2005 and have not been updated since, even though the model benefited from continuous development up to this day.

4. Details about parameters used to train the IDyOM model in the experiments of the PhD thesis are not always given [Pea05]. We assumed that the model used in their experiments was trained with parameters presented as optimal in the implementation details. However, they might have employed a theoretically suboptimal parameter set that produced better results for the specific test pieces used in the experiment.

Model	**Chorale 61** R^2_{adj}	**Chorale 151** R^2_{adj}	**Mean** R^2_{adj}
IDyOMpy	0.030	0.219	0.125
System A	0.342	0.253	0.343
System B	0.432	0.349	0.391
System C	0.452	0.494	0.473
IDyOM	0.487	0.641	0.564

Table 5.4: The fit of each model to Manzara entropy profiles [MWJ92].

5.3 Melody Generation

Having developed and evaluated a Python implementation of IDyOM, we now assess an extension of the model for melody generation. The generation process, as described in Chapter 4, uses a random walk sampling method with temperature scaling. In this section, we will explore the effect of temperature on the mean success ratings of the generated melodies, using Systems A, B, and C designed in the previous section.

A prerequisite for this experiment is establishing a method to quantify the stylistic success of a melody, whether generated by the system or sourced from the dataset. This method is described in the following section.

5.3.1 Evaluation of Melodies

Given that the generated melodies belong to the chorale genre, the most accurate evaluation would come from genre-specific experts who could assess both the system-generated melodies and those from the dataset. Pearce used this approach to evaluate chorale melodies produced by the system [Pea05]. Unfortunately, gathering such experts for this thesis is not feasible. Rather than relying on non-expert evaluations, we have trained a model capable of assigning ratings to melodies based on expert observations and analyses documented in the literature for systems with similar objectives [Pea05].

The evaluation model employed is a multiple linear regression model. The dependent variables are the ratings assigned by judges in Pearce's study on modeling composition [Pea05]. The independent variables are predictors that capture key observations and analyses provided by the judges:

- **Pitch Centre**: Captures the typical pitch range of chorales in the dataset, which are written for soprano voice and generally span about an octave above and below C_5, with a preference for the center of this range.
- **Pitch Range**: Measures the absolute distance, in semitones, between the pitch range of a given melody and the mean pitch range of the dataset. This predictor distinguishes melodies that adhere to the typical pitch range of the chorale genre from those that extend beyond it.
- **Interval Size**: Represents the number of intervals greater than an octave. Such intervals are rare in the chorale genre, where melodies typically follow a stepwise pattern of conjunct motion.
- **Interval Dissonance**: Counts the number of dissonant intervals greater than a perfect fourth. Dissonant intervals are uncommon in chorales.
- **Chromaticism**: Indicates the number of tones outside the scale. While some chromatic notes are common in the chorale genre, an excessive number can disrupt the tonal structure of the piece, making it less representative of the genre.

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	6.0922	0.420	14.495	0.000	5.218	6.966
pitch_centre	-0.1855	0.118	-1.568	0.132	-0.431	0.061
pitch_range	-0.2233	0.106	-2.108	0.047	-0.444	-0.003
interval_size	-0.0622	0.301	-0.207	0.838	-0.687	0.563
interval_dissonance	-0.8474	0.247	-3.436	0.002	-1.360	-0.335
chromaticism	-0.1642	0.040	-4.106	0.001	-0.247	-0.081
harmonic_closure	0.0311	0.485	0.064	0.949	-0.977	1.040

Table 5.5: The results of the multiple linear regression analysis used to fit the ratings provided by judges in Pearce's experiment [Pea05]

- **Harmonic Closure**: Takes the value 0 if the melody ends on the tonic of the key. Chorales traditionally conclude with a perfect cadence, where the tonic is preceded by the dominant.

The results of the multiple linear regression analysis are shown in Table 5.5. While some predictors were not statistically significant, they were retained in the model.

5.3.2 Experimental Methodology

We randomly selected 10 base melodies from the dataset, with a mean rating of 5.15, as determined by the evaluation model. This value serves as a baseline for comparing the ratings of the generated melodies against those from the dataset. The base melodies were removed from the training dataset. For each system (A, B, and C) and each temperature $T \in \{0.02, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0\}$, 10 melodies were generated for each base melody. This results in 100 melodies generated per temperature per system, 1,000 melodies per system, and a total of 3,000 melodies across all systems. The mean success ratings of the 100 melodies generated for each system and temperature were averaged to provide an overall mean success rating.

5.3.3 Results

The mean ratings for each system across different temperatures, along with their corresponding standard deviations, are illustrated graphically in Figure 5.4.

The correlation analysis between temperatures and the mean ratings of each system revealed strong negative correlations. For system A, the correlation was $r = -0.969$ (p <

0.001). System B showed $r = -0.972$ ($p < 0.001$), while system C showed $r = -0.965$ ($p < 0.001$). The strong negative correlations indicate that combining temperature scaling and random walk methods for generating melodic sequences is effective. Specifically, as the temperature decreases, the ratings of the generated melodies tend to be higher, while an increase in temperature results in lower ratings for the generated melodies.

The correlation analysis between temperatures and the standard deviations from the mean rating of each system revealed strong positive correlations. System A showed a correlation of $r = 0.909$ ($p < 0.001$). For system B, the correlation was $r = 0.884$ ($p < 0.001$), while system C showed $r = 0.871$ ($p < 0.001$). Based on this observation, we can conclude that melodies generated at lower temperatures tend to fall within a range closer to the mean rating. However, it is important to note that the reduction in entropy at low temperatures may limit the variety of melodies produced, thereby restricting exploration within the space of possible generations.

System A achieved the highest overall rating of 4.89 at a temperature of 0.2. In comparison, System B's maximum rating is 4.37 at a temperature of 0.2, while System C's maximum rating is 3.72, at a temperature of 0.02. System A, designed to minimize uncertainty in unseen sequences, not only has the best overall rating but also maintains a higher mean rating across all temperatures when compared to the other systems.

The temperature scaling technique improves the random-walk process by up to 908.24% when comparing the performances of System A at temperatures 1.0 and 0.2. However, even the best-performing system has a lower rating than the mean ratings of melodies derived from the dataset, which stands at 5.15. This indicates that, based on our evaluation method, our system nearly succeeds in composing novel melodies comparable in quality to those in the dataset. Additionally, caution is needed when claiming such an improvement, as melodies generated by System A at low temperatures tend to be highly repetitive. This is because System A reduces entropy over unseen pieces, leading to the frequent repetition of patterns.

Consequently, while the melodies generated by the system receive high ratings according to the evaluation model, they may still deviate from the characteristics of the chorale genre. Despite the absence of certain undesirable traits typically associated with a traditional random walk, these generated melodies might not fully adhere to the constraints of chorale music (e.g. a lot of repetition). Examples of melodies generated by System A with different temperatures can be found in Appendix A.

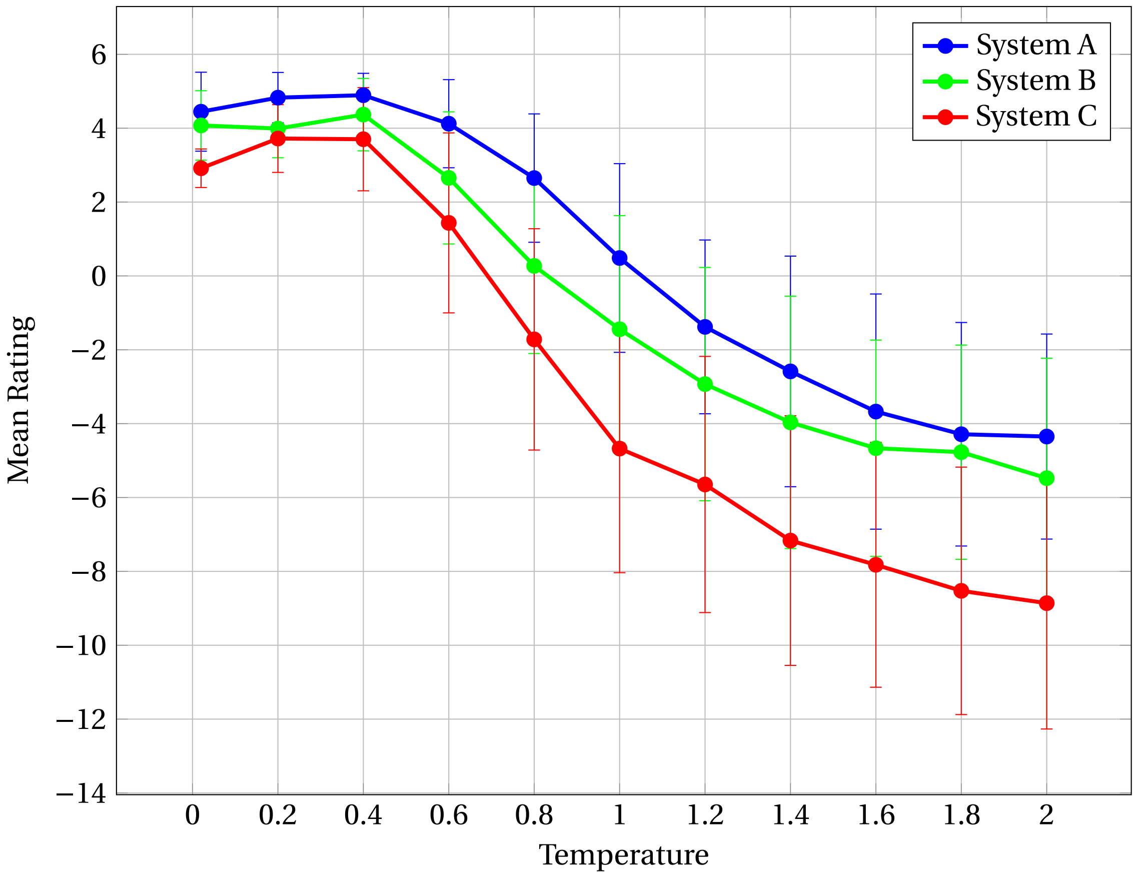


Figure 5.4: Mean rating and standard deviation for each system as a function of temperature.

6

Conclusion

In this master thesis, we investigated the cognitive processes involved in melodic cognition. Specifically, we studied the computational modeling of processes related to melodic perception. The computational model is associated with the problem of sequence prediction: given an ordered sequence of discrete events (the context), the model must determine the probability of any subsequent event, thereby predicting the identity of the next event. Furthermore, based on statistical models of melodic perception, we explored using the random walk technique for sequence generation, employing the temperature-scaling method.

The current leading computational model for auditory melodic perception is the Information Dynamics of Music (IDyOM) model [Pea05; Pea18]. IDyOM has achieved the best results in modeling melodic perception, particularly in predicting melodic expectancy. This machine learning algorithm can simulate the surprise experienced by a listener when hearing an unfamiliar melody. However, IDyOM has significant limitations: it is not easily accessible and is difficult for researchers in music cognition to extend. The model is implemented in Common Lisp, and its installation process is complex and unconventional. The primary objective of this thesis is to reimplement IDyOM in Python, a more accessible and extendable programming language for researchers in music cognition. The secondary objective is to demonstrate the extensibility of the developed model. To this end, we will extend the model to generate melodies using a modified random-walk technique with temperature scaling. This technique should enable the generation of melodies that more successfully adhere to a specific style compared to those generated without temperature scaling.

After developing and training our implementation on a corpus of Western chorales harmonized by J.S. Bach, we evaluated its performance in accounting for patterns of expectations demonstrated by humans in behavioral studies. Notably, each of the developed systems A, B and C, significantly outperformed IDyOMpy [Gui18], which is another IDyOM Python implementation that had the same objectives as the ones presented in this research. System C, which employs an augmentation technique to increase the number of transitions between events in the database, demonstrated better performance than Systems A and B. However, while the designed systems produced valuable outputs, they were still less effective than the original IDyOM implementation when tested under the same conditions, utilizing the same corpus of melodies and feature selection procedure.

In our experiments on melody generation using the random walk technique combined with temperature scaling, we observed a strong negative correlation between temperature and the ratings of the melodies. This indicates that as the temperature decreases, the system is more capable of generating melodies rated more successfully in terms of adherence to the musical style. Additionally, we found a strong positive correlation between temperature and the standard deviation of the melody ratings, suggesting that the system is more likely to produce a successful melody at lower temperatures. Overall, the melodies generated by the system were rated slightly below the average rating of the melodies in the dataset.

The Python implementation we developed offers several significant advantages for different user profiles. For researchers, it provides easier access and usability for studies in auditory and music cognition. The output of IDyOM has been widely used in research on melodic expectancy and uncertainty [Di +20; Pea18], emotional experience [Ege+13], recognition memory [AAP18], and phrase boundary detection [PW06], among others. Additionally, this implementation allows researchers to extend the model for future work, such as adapting it for polyphonic music. For composers, the implementation serves as a valuable tool for co-creativity. The extension we provide enables the generation of melodies, with the ability to fine-tune the process using the temperature parameter. These generated melodies can be used as a creative resource in the composition process.

Several limitations in our research should be addressed. First, while our implementation of IDyOM produced valuable results, they are not identical to those of the original model, nor are the viewpoint selections. This indicates that the two models, though similar, differ in their implementation details. The discrepancies in the results may stem from several factors, such as potential flaws in the conversion of probability distributions over pitches, the reliance on external libraries to extract musical elements not present in MIDI encoding, outdated implementation details in Pearce's thesis compared to the latest code release, as well as missing information about the training parameters (see Section 5.2.6.2). Second, the limited availability of data on human patterns of melodic expectancy restricted our feature selection, forcing us to optimize the model using only two data samples. As a result, there is a relatively high risk of overfitting to this specific dataset. Third, although the ratings for our generated melodies were promising, closer inspection revealed that many highly rated melodies contain excessive repetition, making them stylistically inaccurate. The evaluation method did not account for every style feature, leading us to avoid some incorrect characteristics while introducing others. Finally, our research focuses on monophonic melodies. The developed program is not capable of predicting pieces with vertical harmonies formed by multiple parts. Furthermore, the symbolic representation of the data does not incorporate timbral information, which could influence listeners' perception of music.

A

Melodies Generated By System A

In this appendix, we present five melodies generated by System A, as discussed in Chapter 5. Each melody is derived from Chorale 207, which serves as the base melody for the generation process. The generation was executed with different temperature values T: 0.02, 0.4, 0.6, 0.8, and 1.0.

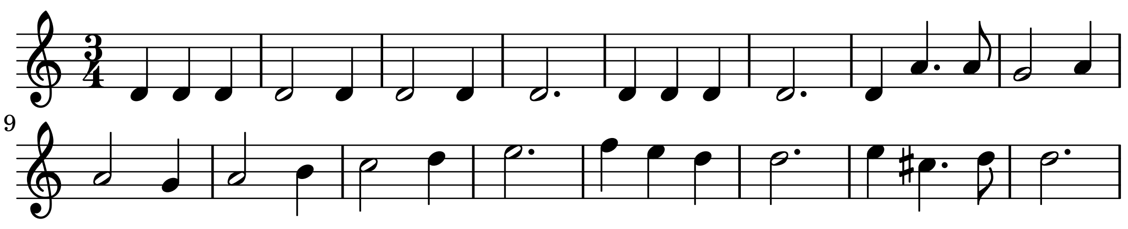


Figure A.1: A melody generated by System A with T = 0.02

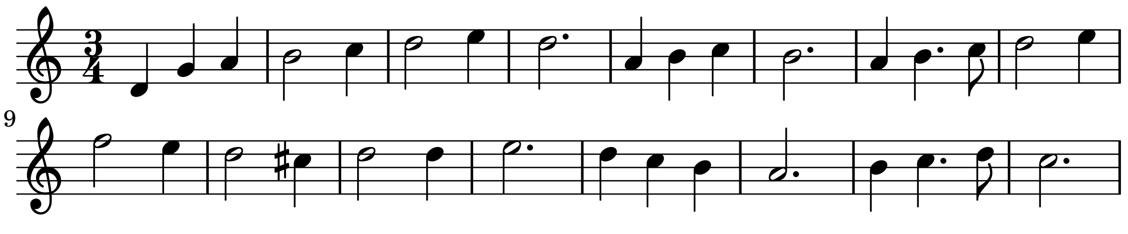


Figure A.2: A melody generated by System A with T = 0.4

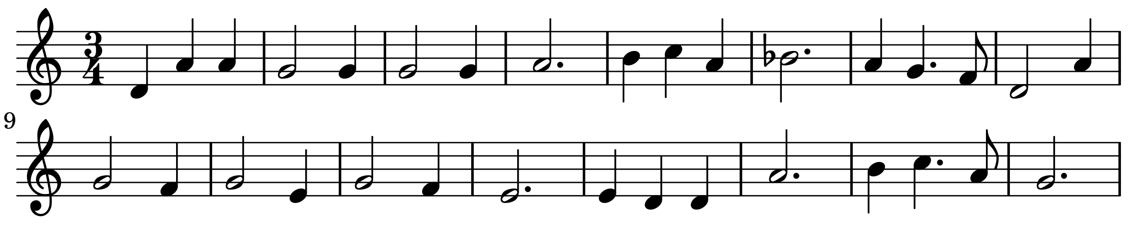


Figure A.3: A melody generated by System A with T = 0.6

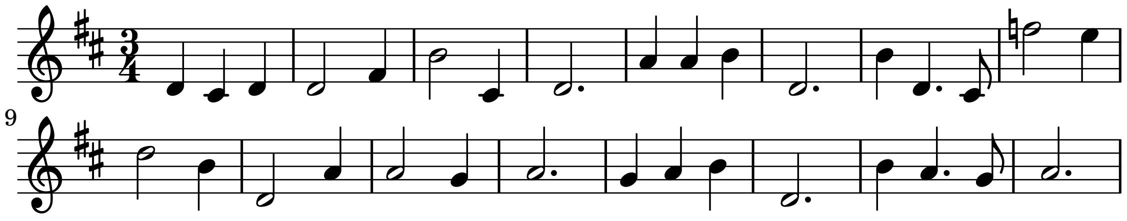


Figure A.4: A melody generated by System A with T = 0.8

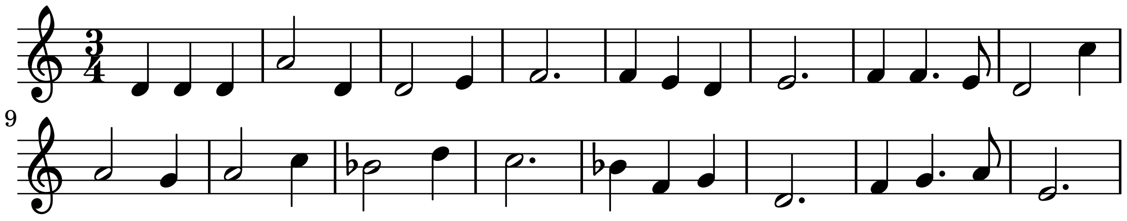


Figure A.5: A melody generated by System A with T = 1.0

B

Script for Dataset Preprocessing

```
#!/bin/bash

soprano_dir="soprano_krn"
midi_dir="midi_files"
mkdir -p "$soprano_dir"
mkdir -p "$midi_dir"

for file in *.krn; do
    base_name=$(basename "$file" .krn)

    # Extract only the soprano voice
    extract -i "*Isoprn" "$file" \
        > "${soprano_dir}/${base_name}_soprano.krn"

    # Convert to MIDI
    hum2mid "${soprano_dir}/${base_name}_soprano.krn" \
        -o "${midi_dir}/${base_name}_soprano.mid"
done
```

Bibliography

AAP18 Agres, Kat, Samer Abdallah, and Marcus Pearce (Jan. 2018). "Information-Theoretic Properties of Auditory Sequences Dynamically Influence Expectation and Memory". In: *Cognitive Science* 42.1. Epub 2017 Jan 25, pp. 43–76. DOI: 10.1111/cogs.12477.

AB95 Aha, David W. and Richard L. Bankert (Apr. 1995). "A Comparative Evaluation of Sequential Feature Selection Algorithms". In: Proceedings of Machine Learning Research R0. Ed. by Doug Fisher and Hans-Joachim Lenz. Reissued by PMLR on 01 May 2022., pp. 1–7. URL: https://proceedings.mlr.press/r0/aha95a.html.

Bro+57 Brooks, F. P. et al. (1957). "An experiment in musical composition". In: *IRE Transactions on Electronic Computers* EC-6.3, pp. 175–182. DOI: 10.1109/TEC.1957.5222016.

CG96 Chen, Stanley F. and Joshua T. Goodman (1996). *An Empirical Study of Smoothing Techniques for Language Modeling*. arXiv: cmp-lg/9606011 [cmp-lg]. URL: https://arxiv.org/abs/cmp-lg/9606011.

CW03a Cleary, John and H. Witten (Mar. 2003). "Data Compression Using Adaptive Coding and Partial String Matching". In: *IEEE Transactions on Communications* 32. DOI: 10.1109/TCOM.1984.1096090.

Con03 Conklin, Darrell (June 2003). "Music Generation from Statistical Models". In: *Journal of New Music Research* 45. DOI: 10.1080/09298215.2016.1173708.

CW03b Conklin, Darrell and Ian Witten (June 2003). “Multiple Viewpoint Systems for Music Prediction”. In: *J. New Music Res* 24. DOI: 10.1080/09298219508570672.

Cop89 Cope, David (1989). “Experiments in Musical Intelligence (EMI): Non-linear Linguistic-based Composition”. In: *Interface* 18.1–2, pp. 117–139. DOI: 10.1080/09298218908570541. URL: https://doi.org/10.1080/09298218908570541.

CL95 Cuddy, Lola L. and Carol A. Lunney (1995). “Expectancies generated by melodic intervals: Perceptual judgments of melodic continuity”. In: *Perception & Psychophysics* 57.4, pp. 451–462. DOI: 10.3758/BF03213071. URL: https://doi.org/10.3758/BF03213071.

Cut+92 Cutting, James et al. (Sept. 1992). “Selectivity, Scope, and Simplicity of Models: A Lesson From Fitting Judgments of Perceived Depth”. In: *Journal of experimental psychology. General* 121, pp. 364–81. DOI: 10.1037//0096-3445.121.3.364.

DE10 Davismoon, Stephen and John Eccles (Apr. 2010). “Combining Musical Constraints with Markov Transition Probabilities to Improve the Generation of Creative Musical Structures”. In: pp. 361–370. DOI: 10.1007/978-3-642-12242-2_37.

Di +20 Di Liberto, Giovanni M et al. (Mar. 2020). “Cortical encoding of melodic expectations in human temporal cortex”. In: *eLife* 9. Ed. by Jonathan Erik Peelle and Barbara G Shinn-Cunningham, e51784. ISSN: 2050-084X. DOI: 10.7554/eLife.51784. URL: https://doi.org/10.7554/eLife.51784.

Ege+13 Egermann, Hauke et al. (Sept. 2013). “Probabilistic models of expectation violation predict psychophysiological emotional responses to live concert music”. In: *Cognitive, Affective, & Behavioral Neuroscience* 13.3, pp. 533–553. DOI: 10.3758/s13415-013-0161-y.

FS01 Farbood, Morwaread and Bernd Schoner (Jan. 2001). “Analysis and Synthesis of Palestrina-Style Counterpoint Using Markov Chains”. In.

GRP22 Guan, Xinyi, Zeng Ren, and Claire Pelofi (2022). “py2lispIDyOM: A Python package for the information dynamics of music (IDyOM) model”. In: *Journal of Open Source Software* 7.79, p. 4738. DOI: 10.21105/joss.04738. URL: https://doi.org/10.21105/joss.04738.

Gui18 Guilhem (2018). “Investigating Statistical Models of Music Perception”. Master’s Thesis. Paris, France: Laboratoire des Systèmes Perceptifs, ENS.

Guo+17 Guo, Chuan et al. (2017). *On Calibration of Modern Neural Networks*. arXiv: 1706.04599 [cs.LG]. URL: https://arxiv.org/abs/1706.04599.

Har24 Harley, Nick (2024). *Idyoms*. https://github.com/nick-harley/Idyoms. Accessed: 2024-07-25.

HCC17 Herremans, Dorien, Ching-Hua Chuan, and Elaine Chew (Sept. 2017). “A Functional Taxonomy of Music Generation Systems”. In: *ACM Computing Surveys* 50.5, pp. 1–30. DOI: 10.1145/3108242. URL: https://doi.org/10.1145%2F3108242.

Hil59 Hiller, Lejaren A. (1959). “COMPUTER MUSIC”. In: *Scientific American* 201.6, pp. 109–121. ISSN: 00368733, 19467087. URL: http://www.jstor.org/stable/24941187 (visited on 02/12/2024).

Hur97 Huron, David (1997). “Humdrum and Kern: Selective feature encoding”. In: *Beyond MIDI: The Handbook of Musical Codes*. Ed. by E. Selfridge-Field. Cambridge, Massachusetts: MIT Press, pp. 375–401.

Kru95 Krumhansl, Carol L. (Mar. 1995). “Music Psychology and Music Theory: Problems and Prospects”. In: *Music Theory Spectrum* 17.1, pp. 53–80. ISSN: 0195-6167. DOI: 10.2307/745764. eprint: https://academic.oup.com/mts/article-pdf/17/1/53/6303948/17-1-53.pdf. URL: https://doi.org/10.2307/745764.

KD90 Kuhn, R. and R. De Mori (1990). “A cache-based natural language model for speech recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12.6, pp. 570–583. DOI: 10.1109/34.56193.

MWJ92 Manzara, Leonard C., Ian H. Witten, and Mark James (1992). “On the Entropy of Music: An Experiment with Bach Chorale Melodies”. In: *Leonardo Music Journal* 2.1, pp. 81–88. ISSN: 09611215, 15314812. URL: http://www.jstor.org/stable/1513213 (visited on 05/24/2024).

Mey57 Meyer (1957). *Meaning in Music and Information Theory*. Vol. 15. 4. [Wiley, American Society for Aesthetics], pp. 412–424. URL: http://www.jstor.org/stable/427154 (visited on 04/11/2024).

Mey73 Meyer, Leonard (1973). *Explaining Music: Essays and Explorations*. English. First Edition. Berkeley, CA: University of California Press, p. 284. ISBN: 978-0520022164.

Mof90 Moffat, A. (1990). “Implementing the PPM data compression scheme”. In: *IEEE Transactions on Communications* 38.11, pp. 1917–1921. DOI: 10.1109/26.61469.

MNW98 Moffat, Alistair, Radford M. Neal, and Ian H. Witten (July 1998). “Arithmetic coding revisited”. In: *ACM Trans. Inf. Syst.* 16.3, pp. 256–294. ISSN: 1046-8188. DOI: 10.1145/290159.290162. URL: https://doi.org/10.1145/290159.290162.

Nar90 Narmour, Eugene (1990). *The Analysis and Cognition of Basic Melodic Structures: The Implication-Realization Model*. URL: https://api.semanticscholar.org/CorpusID:195710276.

Omi+13 Omigie, D. et al. (Aug. 2013). “Electrophysiological correlates of melodic processing in congenital amusia”. In: *Neuropsychologia* 51.9, pp. 1749–1762. DOI: 10.1016/j.neuropsychologia.2013.05.010. eprint: 2013-05-23.

Pea18 Pearce, Marcus (May 11, 2018). “Statistical learning and probabilistic prediction in music cognition: mechanisms of stylistic enculturation”. In: *Annals of the New York Academy of Sciences* 1423.1. Epub ahead of print, pp. 378–395. DOI: 10.1111/nyas.13654.

PW06 Pearce, Marcus and Geraint Wiggins (July 2006). “Expectation in Melody: The Influence of Context and Learning”. In: *Music Perception* 23, pp. 377–405. DOI: 10.1525/mp.2006.23.5.377.

Pea05 Pearce, Marcus T. (2005). “The Construction and Evaluation of Statistical Models of Melodic Structure in Music Perception and Composition”. Doctoral Dissertation. Department of Computer Science, City University of London, UK: City University of London.

Pea+10 Pearce, Marcus T. et al. (2010). “Unsupervised statistical learning underpins computational, behavioural, and neural manifestations of musical expectation”. In: *NeuroImage* 50.1, pp. 302–313. ISSN: 1053-8119. DOI: https://doi.org/10.1016/j.neuroimage.2009.12.019. URL: https://www.sciencedirect.com/science/article/pii/S1053811909013068.

PP90 Perruchet, Pierre and Chantal Pacteau (Sept. 1990). “Synthetic Grammar Learning: Implicit Rule Abstraction or Explicit Fragmentary Knowledge?” In: *Journal of Experimental Psychology: General* 119, pp. 264–275. DOI: `10.1037/0096-3445.119.3.264`.

Pin56 Pinkerton, Richard C. (1956). “Information theory and melody.” In: *Scientific American* 194, pp. 77–87.

Reb67 Reber, Arthur S. (1967). “Implicit learning of artificial grammars”. In: *Journal of Verbal Learning and Verbal Behavior* 6.6, pp. 855–863. ISSN: 0022-5371. DOI: `https://doi.org/10.1016/S0022-5371(67)80149-X`. URL: `https://www.sciencedirect.com/science/article/pii/S002253716780149X`.

RR12 Rohrmeier, Martin and Patrick Rebuschat (2012). “Implicit learning and acquisition of music”. In: *Topics in Cognitive Science* 4.4, pp. 525–553. DOI: `10.1111/j.1756-8765.2012.01223.x`.

SAN96 Saffran, Jenny R., Richard N. Aslin, and Elissa L. Newport (Dec. 1996). “Statistical learning by 8-month-old infants”. In: *Science* 274.5294, pp. 1926–1928. DOI: `10.1126/science.274.5294.1926`.

Saf+99 Saffran, Jenny R. et al. (Feb. 1999). “Statistical learning of tone sequences by human infants and adults”. In: *Cognition* 70.1, pp. 27–52. DOI: `10.1016/s0010-0277(98)00075-4`.

Sch97 Schellenberg, E. (Apr. 1997). “Simplifying the Implication-Realization Model of Melodic Expectancy”. In: *Music Perception* 14, pp. 295–318. DOI: `10.2307/40285723`.

Sch+03 Schellenberg, E. et al. (Jan. 2003). “Expectancy in melody: Tests of children and adults”. In: *Journal of experimental psychology. General* 131, pp. 511–37. DOI: `10.1037//0096-3445.131.4.511`.

Sch96 Schellenberg, E. Glenn (Jan. 1996). “Expectancy in melody: tests of the implication-realization model”. In: *Cognition* 58.1, pp. 75–125. DOI: `10.1016/0010-0277(95)00665-6`.

Sha51 Shannon, C. E. (1951). “Prediction and entropy of printed English”. In: *The Bell System Technical Journal* 30.1, pp. 50–64. DOI: `10.1002/j.1538-7305.1951.tb01366.x`.

TC98 Teahan, W. and John Cleary (Feb. 1998). “Models of English text”. In: *Proceedings of the Data Compression Conference*.

Wag+12 Wagemans, Johan et al. (Nov. 2012). “A century of Gestalt psychology in visual perception: I. Perceptual grouping and figure-ground organization”. In: *Psychological Bulletin* 138.6. Epub 2012 Jul 30, pp. 1172–1217. DOI: `10.1037/a0029333`.

WB91 Witten, Ian and Timothy Bell (July 1991). “The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression”. In: *IEEE Transactions on Information Theory* 37, pp. 1085–1094. DOI: `10.1109/18.87000`.

WC90 Witten, Ian H. and Darrell Conklin (1990). “PREDICTION AND ENTROPY OF MUSIC”. In: URL: `https://api.semanticscholar.org/CorpusID:63091676`.