POLITECNICO
MILANO 1863

Dipartimento di Elettronica, Informazione e Bioingegneria

Master Degree in Computer Science and Engineering

Tranformer Networks for the modelling of Jazz Harmony

by:
Giovanni Agosti

matr.:
928366

Supervisor:
Augusto Sarti

Co-supervisor:
Clara Borrelli

Academic Year
2020-2021

Abstract

Music is a multi-layered form of art and a language of its own.
It can in fact be considered one of the most ancient forms of communication between humans and can therefore be studied with the same tools we use to study natural languages. In order to define a clear linguistic framework and to properly apply it to the musical phenomenon we need to separate music into its atomic components, which can be considered as melody, harmony and rhythm, and then take into account their mutual interactions.

In this work we mainly focus on harmony and on its interaction with rhythm and musical structure which creates the so-called harmonic rhythm. Harmonic rules and praxis are widely recognized to be very much culturally dependent and it is well known that some chord progression can sound very weird for someone with a specific cultural background and at the same time very familiar for someone with a different one. In this thesis we specifically focused on Jazz harmony, which is a harmonic framework mostly based on the traditional western music harmonic rules.

In this context we investigated how to define the perceived complexity of an harmonic sequence and we tried to relate it to its unpredictability. Predictable sequences should be the ones for which the listener could easily guess in advance the next chord thanks to the presence of some previously heard common pattern. In this framework we can define as complex a sequence which is hard to predict and which creates in the listeners a sense of unsatisfied expectation. On the other hand, simple sequences are the ones which completely fulfill the expectation for the next chords and which should be simple to predict, both for a human listener and for a language model. Furthermore, we investigated the existence of a correlation pattern between the perceived complexity annotated from a set of listeners and the ability of a deep learning model to predict the next chord in a sequence.

For the purpose of this work we trained a Neural Network (NN) model based on the Generative Pretrained Transformer (GPT) architecture proposed by OpenAI in 2019. This architecture is a state of the art model for Natural Language Processing (NLP), a field of study which investigates how to model natural languages using NNs. The innovative aspect

of the Transformer resides in its specific attention mechanism which allows it to capture long term dependencies within the input sequences without the use of recurrency. We trained the model with two versions of a novel dataset containing more than 100 000 chord annotations taken from the well known Real Book of Jazz, a collection of lead sheets of the so-called Standards of jazz music. Furthermore, we validated the model complexity estimates exploiting perceptual complexity ratings by means of a listening test.

Even though a strong correlation between the cross-entropy calculated from the model and the perceptual ratings of a group of listeners was shown in [1], we actually did not observe this correlation in our experiments. This could be due to the high level of musical sophistication implied in the repertoire that we used for the training, resulting in difficulties for most non music trained listeners in decoding the sequences by ear and in properly evaluating their complexity. Furthermore, the Jazz vocabulary is not as widely diffused as the Pop or Rock ones, so listeners, on average, lack the necessary amount of experience of this particular music genre that is needed in order to properly identify what is a common jazz sequence and what is a very uncommon one.

As far as chord prediction is concerned, we can confirm that a GPT based architecture can produce coherent sequences of chords and that can learn harmonic rhythm patterns as well, a feature which can be used in interesting ways as a composition assistant tool.

Sommario

La musica è una forma d'arte che si esprime su vari livelli e rappresenta un linguaggio a sé stante.
Infatti, può essere considerata una delle più antiche forme di comunicazione tra essere umani e può quindi essere studiata con gli stessi strumenti con cui si approcciano i linguaggi naturali. Per definire un chiaro approccio di studio linguistico alla musica, è necessario separarla nei suoi aspetti fondamentali, che possiamo considerare come melodia, armonia e ritmo, e studiare le rispettive interazioni.

In questo lavoro ci concentriamo principalmente sull'armonia e sul suo legame con il ritmo. Le regole e le prassi dell'armonia sono universalmente riconosciute come essere in gran parte dipendenti dalla cultura di appartenenza. E' infatti risaputo che la stessa sequenza di accordi possa suonare come assolutamente banale per un individuo e contemporaneamente come completamente imprevista per un altro appartenente a una diversa cultura. In questa trattazione ci siamo concentrati sull'armonia Jazz, una prassi armonica che può a grandi linee essere inscritta all'interno dell'insieme della cultura musicale occidentale, ma che presenta comunque alcune caratteristiche molte specifiche e peculiari.

In particolare, abbiamo investigato come definire il concetto di complessità armonica associata a una sequenza di accordi e abbiamo cercato di legarlo alla sua imprevedibilità. Infatti, sequenze prevedibili dovrebbero essere percepite come poco complesse, mentre sequenze molto improbabili dovrebbero essere percepite come estremamente complesse. Inoltre, abbiamo investigato la presenza di una correlazione tra la complessità percepita di una sequenza e l'abilità di un modello informatico di predirla.

Per la scrittura di questa tesi abbiamo implementato un modello basato sull'architettura GPT-2 proposta da OpenAI nel 2019. Questo modello rappresenta una delle ultime proposte nel campo del NLP, una branca dell'informatica che studia i linguaggi naturali. Durante il presente lavoro abbiamo allenato il modello con un database originale di nostra proposta trascritto dall'applicazione iRealBook creata da Massimo Biolcati nel 2010. Il database utilizzato contiene più di 100 000 sequenze di accordi in tutte le tonalità tratte dai vari volumi del noto Real Book, uno storico archivio di trascrizioni dei cosiddetti Standards della musica Jazz. Inoltre abbiamo valutato la capacità del modello di

predire la complessità percepita delle sequenze di accordi tramite un test di ascolto.

Anche se una forte correlazione negativa tra capacità predittiva del modello e complessità percepita era stata dimostrata in [1], non abbiamo trovato la suddetta correlazione all'interno dei nostri dati. Questo può essere dovuto a varie ragioni, tra cui il più alto grado di sofisticazione del repertorio incluso nel database usato e la minore diffusione del Jazz rispetto ad altri generi come il Pop o il Rock.

Per quanto invece riguarda l'obiettivo di modellare le regole e le prassi dell'armonia jazz possiamo confermare che il modello GPT-2 produce sequenze di accordi coerenti con il database con cui è stato allenato. Inoltre abbiamo evidenziato come il modello abbia efficacemente imparato anche il concetto di ritmo armonico, caratteristica che potrebbe essere efficacemente sfruttata come strumento di composizione assistita.

Acknowledgments

I would like to thank my supervisor Prof. Augusto Sarti and my co-supervisor Clara Borrelli for their guidance in the making of this project and for pushing me in investigating such a beautiful field of study.

My biggest thanks goes to my parents for their endless support during these many years of studies in conservatories and universities around the globe.

A sweet thank you goes to my girlfriend Martina for her everyday support and love.

A special thanks goes out to Gianluca Elia for being one of the most brilliant individuals I know and for being a friend, always willing to help and share his knowledge.

Contents

List of Figures

List of Tables

Glossary

AR Auto-Regressive. 11

ARIMA Auto-Regressive Integrated Moving Average. 12

ARMA Auto-Regressive Moving Average. 11

BPTT Back Propagation Through Time. 18

CNN Convolutional Neural Network. 22

CV Computer Vision. 18

DAW Digital Audio Workstation. 28

DBN Dynamic Bayesian Network. 27

ES Exponential Smoothing. 11, 12

FFNN Feed Forward Neural Network. 15, 17–19, 34, 35

Gold-MSI Goldsmiths Musical Sophistication Index. ix, 46, 53

GPT Generative Pretrained Transformer. i–iv, 1, 2, 22, 29, 31, 33–35, 43, 45, 55, 56

GRU Gated Recurrent Unit. 2, 27, 28

HMM Hidden Markov Models. 11–13, 18, 27, 28

LSTM Long-Short Term Memory. viii, 2, 14, 18, 19, 26–29

MA Moving Average. 11

NLP Natural Language Processing. i, iii, 1, 2, 4, 14, 17, 18, 23, 30–32

NN Neural Network. i, 14, 15, 17, 27, 28

PPM Prediction by Partial Matching. 14, 26, 28

RNN Recurrent Neural Network. viii, 2, 14, 17–19, 26–29

1 Introduction

In this thesis we focus our attention on the analysis of jazz chord sequences, specifically on chord prediction and perceived complexity estimation. First, we want to evaluate the ability of the GPT-2 model for NLP to correctly predict future tokens within an harmonic sequence in the jazz context. Moreover, we approach the problem of complexity estimation of an harmonic progression and we aim at evaluating the ability of our model to correctly estimate what would be the perceived complexity of a sequence of chords. Furthermore, we investigate the hypothesis proposed in [1] that these two tasks can be related, as they demonstrated that the perceived complexity of a sequence can be linked to the ability of a NLP model to correctly predict it.

Automatic composition of chord sequences and the estimation of their complexity are tasks which belong to the field of study known as Music Information Retrieval (MIR). MIR is a branch of music engineering born in order to extract high level information that can be descriptive of some musical content, directly from the data. This approach is made possible because of the existence of huge databases of music, both in audio format and in symbolic representation. Manually extracting information from this kind of sources would be nearly impossible, therefore it is necessary to provide tools able to automatically analyze these datasets and extract useful features. In this context, estimating the complexity of a chord sequence can be a tool for music recommendation systems and genre classification while chord prediction can be used for automatic composition.

Giving a formal definition of harmonic complexity is a challenging

task and can be approached both on a musicological level and on an information theory level. The first approach consists in relying on a prior theoretical musical knowledge and exploits the rules of harmony to evaluate how much and in which ways two sequences can be compared based on their complexity. Following this approach we can exploit traditional western musical concepts such as tonality, functionality and cadences to assess the value of the complexity of a sequence [3, 5, 6, 7, 8, 9].

The second approach links the problem of harmonic complexity estimation to information theory, in particular to the concept of *cross-entropy*, considering as more complex sequences with high cross-entropy values. These sequences, in fact, bring a higher level of new or unlike information to the listening context and should then result in a more complex musical progression. Following this approach we aim at extracting the rules which govern music composition and perception straight from the data [1, 10, 11].

For what concerns the problem of chord prediction, we can define it as the task of estimating the most likely next token in a sequence of chords, given all the previous ones. Specifically, we can base our choice of the next token on its cross-entropy value. Chords which are very likely will exhibit a low cross-entropy while chords with high cross-entropy values are less prone to be chosen for a correct forecasting. In this framework, at each time step we can sample the next token with a k-top sampling algorithm, which means choosing the next chord based on its cross-entropy value. Tokens which have the lowest cross-entropy within the context of each prediction would correspond to a value of $k = 0$, while increasing the value of k would lead to choosing a token with increasing cross-entropy. This technique could lead to either sequences which have a constant-through-time cross-entropy, and sequences which follow particular *cross-entropy envelopes*.

State of the art solutions proposed in literature to tackle the chord prediction problem are based both on machine learning techniques, such as n-gram models [1, 12, 13, 10, 14] and Hidden Markov Models [1, 12, 15, 16], and on Deep Learning algorithms such as RNNs. Typically employed RNNs for this task are the LSTM and the Gated Recurrent Unit (GRU) [1, 17, 11, 18, 19, 20], which anyway, even if lead to good results, are still limited to a short time context for the evaluation of the prediction. Extremely better performances have been recently obtained exploiting the newly introduced Transformer model [21, 22, 23] which can capture dependencies from tokens which are very far from each other in the input sequence.

In this thesis we propose a data-driven method that uses a Transformer based architecture proposed by OpenAI in 2019 called GPT-2 which is a state of the art model in the field of NLP. As a matter of fact we model harmony as a language of its own where each chord represents

a word and each chord sequence represents a sentence in the language. In this framework, we approach the harmonic complexity estimation problem as described in [1], comparing the model's estimates of cross-entropy to a set of perceptual ratings collected by means of a listening test. Our goal is to evaluate the capability of our model to correctly estimate the perceived complexity of a set of chord progressions.

As far as chord prediction is concerned we evaluated the performance of our model both using objective metrics and experimenting in generating sequences using various orders of k-top sampling.

For our experiments we propose a newly retrieved database of jazz chord annotations collected from the iRealPro application. The iRealPro is a digitalization of the many volumes of the well known Real Book of jazz which contains chord annotations of the so called Standards of Jazz Music.

Both for evaluating our prediction results and for the problem of complexity estimation, we propose an *information-theory* approach based on cross-entropy, where low cross-entropy is associated to a good ability of the model in predicting the next chord in the sequence, and accuracy, defined as the number of chords that are correctly predicted by our model.

Although a very strong correlation between perceived complexity and cross-entropy is shown in [1], our result did not show a clear correlation. This can be due to many reasons. One of them can definitely be the fact that the experiment proposed in [1] was specifically designed to deal with *tonal* complexity while we proposed a test which contained progressions in all the twelve keys and that were highly modulating as well. A second reason can be the higher level of sophistication present within the repertoire included in the database. In fact, jazz harmony is mainly characterized by chords extended at least until the seventh, which might be difficult to decode for a not musically trained listener. Furthermore, Jazz music is not as widely diffused as Pop or Rock music, so listeners might have had a hard time discerning what is more complex or uncommon and what is a typical jazz sequence. Anyway, the sequences produced by the model have shown to be appropriate within the jazz vocabulary and some well known progressions have been clearly learned by the model such as II-V-Is progression or blues progressions.

The outline of this work is the following.
In Chapter 2 we introduce the fundamental concepts implied in the rest of the work. Specifically in Section 2.1 we introduce some basic harmonic notions, with a specific stress on the extension of the chords until the 7^{th} as being a widely employed praxis in the jazz context, in Section 2.2 we introduce the fundamentals or Time Series Forecasting and in Section 2.3 we introduce the most important Deep Learning models.

In Chapter 3 we offer an overview of some of the state of the art methods employed for the tasks of chord prediction and harmonic complexity

estimation both in the Machine Learning and Deep learning fields.

In Chapter 4 we formally introduce the methodologies that we employed for the chord prediction and complexity estimation tasks and we present the NLP model that we used for our experiments.

In Chapter 5 we describe our experiments, their outcomes and the metrics that we used for evaluating their results along with some qualitative observations.

In Chapter 6 we summarize the contribution of our work and propose further development for this study.

2
Background

In this Chapter we present the concepts which are necessary to understand the discussion presented in this work. In section 2.1 we introduce the basic concepts of Western music harmony with a particular regard to the extension of the chords until the 7^{th}. In Section 2.2 we present some classical techniques for time series forecasting while in Section 2.3 we introduce the most fundamentals Deep Learning architectures.

2.1 Harmony Fundamentals

2.1.1 Pitch and Pitch Classes

Pitch is a perceptual property of sounds that allows their ordering on the frequency scale. Informally we can define pitch as the quality that makes possible to distinguish two notes played on the same instrument. Pitch is a major auditory attribute of sound, along with duration, loudness, and timbre. An octave is the distance between a pitch and another one which has either the double or the half of the frequency of the first. Due to its mathematical and geometrical properties, the octave can be considered as a pure natural interval. Different cultures divide the octave in a variable number of pitch classes which can be arbitrarily placed within the octave itself. In the contemporary Western tuning system, called *Equal Temperament System*, each octave is divided into 12 logarithmically spaced tones, following the formula:

$$f_p = 2^{1/12} f_{p-1}. \tag{2.1}$$

Because the human hearing system perceives as equivalent two notes played at the distance of one octave, the Equal Temperament System

Table 2.1: Pitch classes and some of their related frequencies on different octaves

Pitch Class	**Octave 1**	**Octave 2**	**Octave 3**	**Octave 4**
C	66 Hz	132 Hz	264 Hz	528 Hz
C#/Db	70 Hz	140 Hz	280 Hz	560 Hz
D	74 Hz	148 Hz	296 Hz	592 Hz
D#/Eb	78 Hz	156 Hz	312 Hz	624 Hz
E	83 Hz	166 Hz	332 Hz	664 Hz
F	88 Hz	176 Hz	352 Hz	698 Hz
F#/Gb	93 Hz	186 Hz	372 Hz	744 Hz
G	98 Hz	196 Hz	392 Hz	784 Hz
G#/Ab	104 Hz	208 Hz	416 Hz	832 Hz
A	110 Hz	220 Hz	440 Hz	880 Hz
A#/Bb	117 Hz	234 Hz	468 Hz	936 Hz
B	124 Hz	248 Hz	496 Hz	992 Hz

defines 12 pitch classes within an octave frequency range. Each pitch class is related to different frequencies corresponding to the same note at different octaves.

The distance between two notes its called interval and the smallest possible interval is called semitone or half step. Musical objects like scales and chords are defined by particular interval patterns. These patterns create specific sounds which can be recognized as unique even when the pattern is translated over the pitch class domain.

Since the most common scale patterns in Western music have seven notes, even though 12 pitch classes are defined, note names are only 7 and special symbols such as the *flat* and *sharp* are used to refer to the remaining 5 pitch classes. This issue creates an ambiguity in the proper name to use to refer to the same pitch class which is called *enharmony*. In this framework, the correct name to assign to a pitch is determined by the current tonal context. Furthermore, performers who play instruments which can reproduce a continuous set of pitches, such as strings and horns, can slightly detune some notes from their assigned frequency, even within the context of Equal Temperament, in order to better adjust to the harmonic context and to better fit with the overtones produced by the other players.

2.1.2 Scales, Chords and Keys

A scale is a subset of pitches that span the range of an octave. In Western music the most important and widely used scales are composed of 7 notes. Since, as said above, we have only 7 note names to cover 12 pitch classes, it is common practice to enumerate the note names of a scale in such a way that we use all note names in each scale.

Table 2.2: Interval pattern of the major scale

T	T	S	T	T	T	S

The most fundamental and used scale in western music is the *major scale*. This scale is so important that it determines the logic of construction of the keyboard of a piano: by playing all white keys from a C note to the consequent one we are playing a C major scale. The interval pattern of the major scale is the one displayed in Table 2.2.

By translating this pattern on all the pitch classes we can generate all the major scales. Because of enharmony, some scales could be named in 2 different ways and it is common practice to choose the naming that minimizes the alterations needed for that scale. Furthermore, if we display the scales by 5^{th} intervals, we can observe that sharps increase clockwise and that flats increase anti clockwise. This representation is known as *Circle Of Fifth* and displays a sort of map of the tonal world. Keys which are close in the circle are tonally close to each other even if they are not on a keyboard, because they share all the notes except one.

In Western culture the major scale is commonly perceived with a clear sense of direction toward the first note of the scale, which is called *tonic* and which gives a sense of stillness and satisfaction.

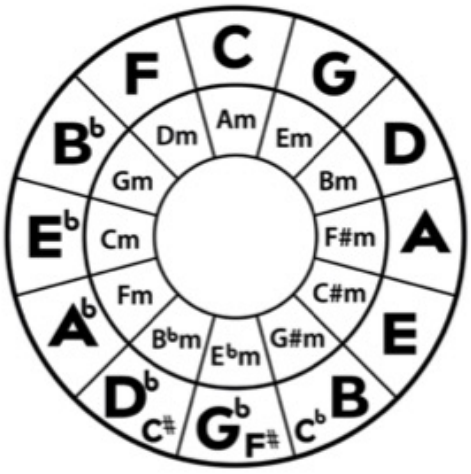

Figure 2.1: Circle of fifths.

Given a major scale it is possible to build chords on it. Chords are groups of three or more notes that follow specific patterns. In particular, in Western culture, it is mainly used the so-called *tertial harmony*, which defines a chord as a superposition of 3rd intervals. So, given a major scale, we can construct 7 chords by superimposing 2 or more 3rd intervals on each note of the scale. As we can see in Table 2.3, by harmonizing a major scale we define 3 kinds of chords: *major(maj7)*, *minor(min7)*, *dim(half diminished)*.

Within these 7 chords, the one built on the 5th degree is commonly

Table 2.3: C major scale harmonization with triads and seventh chords.

Root	**Triad**	**Seventh**
C	C	Cmaj7
D	Dmi	Dmi7
E	Emi	Emi7
F	F	Fmaj7
G	G	G7
A	Ami	Ami7
B	Bdim	Bmi7(b5)

recognized as being the most tense and far from the tonic, especially if extended with its seventh. This tension and the expectation for its release on the tonic creates the well known V-I progression also known as *Perfect Cadence*, representing the most important chord sequence in the context of Western culture as it fully describes a tonal centre. Combining the

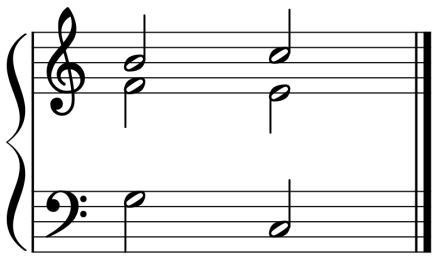

Figure 2.2: Voice leading of the perfect cadence in C major: the 4^{th} resolves on the 3^{rd} and the 7^{th} resolves on the tonic.

major scale interval pattern and the perfect cadence it is possible to define 3 kinds of minor scale which will all have their own chord harmonization. The *natural minor* scale is defined as relative to a major scale and has the same set of notes starting from its 6^{th} degree. For instance the relative minor scale of *C major* is *A minor*.

Because the 2 scales share the same notes, except for the fact that they are shifted, they share the same harmonization as well. This determines the fact that on the 5^{th} degree of a natural minor scale a minor triad arises, instead of a major one like in the major scale. Having a minor triad on the 5^{th} degree is not suitable for creating the perfect cadence within the natural minor scale, which would require a major triad with a minor seventh to create the so-called *Dominant* chord which plays a key role in the V-I progression. In order to guarantee the existence of the perfect cadence even in the minor key, an artificial adjustment of the natural minor scale is needed, which can transform the chord on the 5^{th} degree into a dominant one. This adjustment is done by raising by a semitone the7^{th} note of the natural minor scale, creating the so-called *harmonic minor* scale. This scale is called harmonic because it is defined

Table 2.4: A minor natural harmonization with triads and seventh chords.

Root	**Triad**	**Seventh**
A	Ami	Ami7
B	Bdim	Bmi7(b5)
C	C	Cmaj7
D	Dmi	Dmi7
E	Emi	Emi7
F	F	Fmaj7
G	G	G7

for a harmonic reason: guaranteeing the existence of the perfect cadence in the minor key.

In this way we are, as a matter of fact, building a previously unheard interval pattern which leads to a new set of chords harmonizing the scale. As we can see in Table 2.5, by harmonizing the harmonic minor scale we

Table 2.5: A minor harmonic harmonization with triads and seventh chords.

Root	**Triad**	**Seventh**
A	Ami	Ami(maj7)
B	Bdim	Bmi7(b5)
C	Caug	Cmaj7(#5)
D	Dmi	Dmi7
E	E	E7
F	F	Fmaj7
G#	G#dim	G#dim7

define a new kind of triad, the augmented one, and 3 new kinds of 7th chord: *min(maj7)*, *maj7(sharp5)* and *diminished 7*.

Even if the minor harmonic scale solves the problem of guaranteeing the existence of the perfect cadence, allowing the composers to properly express the minor tonality, it presents some issue in its melodic interval pattern. In fact by raising the 7^{th} note of one semitone a new melodic interval arises as well between the 6^{th} and the 7^{th} note of the scale. This interval, which is composed of one tone and a half, is present between 2 consecutive notes in the scale and so it is still named as a 2^{nd} instead of a minor 3^{rd}. This particular kind of 2^{nd} interval its called *augmented second*, and it is particularly difficult to sing and generally sounds weird for a western listener. In order to solve this melodic issue, the *minor melodic* scale is defined, which is constructed by raising of one semitone the 6th note of the minor harmonic scale. This solution guarantees both the existence of the perfect cadence in the minor key and a certain

Table 2.6: A minor melodic harmonization with triads and seventh chords.

Root	**Triad**	**Seventh**
A	Ami	Ami(maj7)
B	Bmi	Bmi7
C	Caug	Cmaj7(#5)
D	D	D7
E	E	E7
F#	F#dim	F#mi7(b5)
G#	G#dim	G#mi7(b5)

melodic smoothness and singability. The harmonization of the minor melodic scale generates a new pattern of consequent chord kinds with respect to the previous scales, but does not define any new chord kind of its own. The 4 types of scale described above are the most widely used in the western musical culture and fully define the so-called *Tonal Harmonic Framework* which is mainly characterized by a tension/release movement between the dominant and the root of a key or of their substitutes. Other harmonic frameworks are also in use, such as the modal one, but are most of the time implanted in some kind of tonal or functional context.

2.1.3 Harmonic Progressions

Even though the concept of western harmony firstly arose as a consequence of a polyphonic form of writing, i.e. a composing style based on melodies which would eventually be played or singed together and so needed to be in harmony with each other, it is legit to say that in most contemporary western music we can clearly separate harmony and melody. Harmony is displayed by means of *chord symbols* or roman numbers which are referred to a subregion of the time of the piece, like for instance one bar or half a bar. This means that the harmonic content of a relatively long subregion of time can be described with just one symbol.

These symbols are called chord symbols and represent a compressed version of the harmonic information which is implied by the notes contained in a specific time subregion. A sequence of chord symbols is called harmonic progression or chord progression. Widely used sequences of 2 or 3 chords are called *cadences* and represent pre-constituted sentences in the language of harmony. As already mentioned, the most important cadence in western music is the Perfect one (V-I), because it fully defines a tonal centre. Another widely used cadence is the plagal one (VI-I), which exhibits a smoother behaviour with respect to the perfect one.

Chord progression can also be longer than just 2 chord symbols. A very well known progression in jazz harmony is the II-V-I progression, which represents an expansion of the perfect cadence and which can be

encountered in almost every jazz tune.

Chord progressions can also be formed using chords which belong to different keys. We refer to this phenomenon as *modulation* from one key to another one. Usually, if the two keys involved in the modulation are close on the circle of fifths, the modulation is perceived as plain and natural, while, on the other hand, if the keys are far from each other, the modulation sounds more complex or unexpected.

2.2 Time Series Forecasting Fundamentals

Time series forecasting is the process of analyzing time series data using statistics and mathematical models in order to make predictions. Forecasting has a wide range of practical applications including economy, weather forecasting, healthcare, social studies and more. In this section we briefly present some of the most important time series forecasting techniques such as Auto-Regressive (AR) Models, Exponential Smoothing (ES) Models, Hidden Markov Models (HMM) and Finite Context Models.

2.2.1 Auto-Regressive Models

Auto-regressive (AR) models consists of a representation of a type of random process used to describe time-varying phenomenons in a number of fields such as physics, economics, nature and more. The AR model specifies that the output variable depends linearly on its own previous values and on a stochastic term.

The simplest AR model defines the prediction of a variable as a linear combination of its past values. Mathematically we can define the prediction at time t of a variable y as:

$$y_t = \alpha_0 + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} ... + \alpha_p y_{t-p} + \epsilon_t \tag{2.2}$$

where α_0, α_{01}, ..., α_P are the linear regression coefficients, ϵ_t is the random error, y_t, y_{t-1}, y_{t-p} are past observations of the variable to predict and p is the order of the model.

By expressing the forecast as a linear function of the past prediction errors we obtain a variation on the basic model called Moving Average (MA). In mathematical terms we can write:

$$y_t = \phi_0 + \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + ... + +\phi_q \epsilon_{t-q} \tag{2.3}$$

where ϕ_0, ϕ_1, ..., ϕ_q are the moving average coefficients and ϵ_{t-1}, , ϵ_{t-q} are the prediction errors from the past, ϵ is the random error variable and q is the order of the model.

To further expand the predictive performance of the model, AR and MA can be combined obtaining an Auto-Regressive Moving Average

(ARMA) model that can be described as a linear combination of either past prediction values or past prediction errors.

Moreover, in order to make our model capable of working with data which present particular trends during history, the so-called Auto-Regressive Integrated Moving Average (ARIMA) implementation has been introduced [24].

2.2.2 Exponential Smoothing Models

Exponential smoothing is a technique used for smoothing time series data using an exponential window function. Whereas in the simple moving average the past observations are weighted equally, exponential functions are used to assign exponentially decreasing weights over time so that events which are more far in the past get a smaller weight. Exponential smoothing is often used for analysis of time-series data and represents one of the many functions commonly applied to filter data in signal processing.

Simple ES models compute predictions as a weighted average of past data applying an exponential decay of the coefficients as the observations go more far in the past. This weighting approach ensures to give less importance to very old events while keeping a high attention on the most recent ones.

We can define the prediction for time $(t+1)$, given previous t demands $\{d_1, d_2, , d_t\}$, as:

$$f_{t+1} = \alpha d_t + \alpha(1-\alpha)d_{t-1} + \alpha(1-\alpha)^2 d_{t-2} + ... \tag{2.4}$$

where α is in the range $0 \leq \alpha \leq 1$, and it is called smoothing constant.

Simple ES techniques do not perform well when there is a seasonality in the data. In such situations, several methods were devised under the name "double exponential smoothing" or "second-order exponential smoothing," which consist in the recursive application of an exponential filter [25, 26].

2.2.3 Hidden Markov Models

A HMM is a Markov model in which the system being modeled is assumed to be a markovian process with non-observable states. The definition of a HMM requires that there should be an observable process Y whose outcomes are influenced by the outcomes of a hidden process X in a known way. HMMs also require that the outcome of Y at time $t = t_0$ can be influenced only by the outcome of X at $t = t_0$ and that the outcomes of X and Y at $t < t_0$ must not affect the outcome of Y at $t = t_0$. This requirement is known as limited horizon assumption.

HMMs applications span across various fields such as thermodynamics, statistical mechanics, physics, chemistry, economics, finance, signal

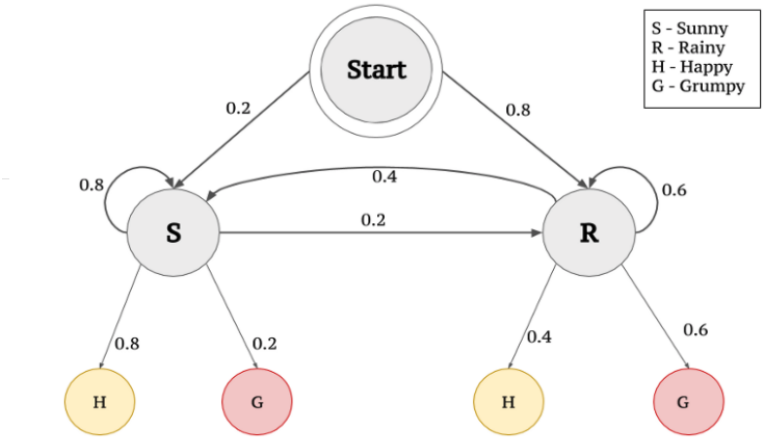

Figure 2.3: Hidden Markov Model with two states $S = \{\text{Happy, Grumpy}\}$ and two hidden states $Q = \{\text{Sunny, Rainy}\}$.

processing, information theory, pattern recognition and musical score following.
In order to fully describe a HMM the following elements are needed:

- N, which represents the number of states in the model. Individual states can be denoted as $\mathbf{q} = \{q_1, q_2, .., q_n\}$ and the state at time t as z_t.
- M, is the discrete alphabet size of the problem. It corresponds to the number of distinct observations for each symbol per state.
- The state transition probability distribution $A = \{a_{ij}\}$ where:

$$a_{ij} = P(z_{t+1} = q_j | z_t = q_i) \tag{2.5}$$

for:

$$1 \leq i, j \leq N \tag{2.6}$$

- The observation symbol probability distribution in state j denoted as $B = \{b_j(k)\}$, with:

$$a_j(k) = P(x_t = v_k | z_t = q_j) \tag{2.7}$$

for:

$$1 \leq j \leq N, 1 \leq k \leq M \tag{2.8}$$

- The probability distribution of the initial state $\pi = \pi_i$, with:

$$\pi_i = P(z_1 = q_i) \tag{2.9}$$

for:

$$1 \leq i \leq N \tag{2.10}$$

2.2.4 Finite Context Models

In the fields of NLP and probability, an n-gram is a contiguous sequence of n items from a given sample of text or speech. The items can be events, syllables, letters or words according to the application. An n-gram model is a type of probabilistic language model for predicting the next token in a sequence in the form of a (n-1) order Markov model. N-gram models are widely employed in probability, communication theory, NLP tasks, computational biology and data compression. Two benefits of n-gram models are simplicity and scalability. Considering a single-event prediction e_i, the probability of the event, given only the previous n elements, can be expressed as:

$$p(e_i|e_0^n) = p(e_i|e_{i-n}^{i-1}) \tag{2.11}$$

where e_i^j is a sequence of symbols from i to j.

The main limitation of n-grams is that high-order models often suffer from zero-frequency probability, which means that some n-grams do not appear in the training dataset. In order to get around this issue, Prediction by Partial Matching (PPM) algorithms have been introduced. PPM is an adaptive statistical data compression technique based on context modeling and prediction. PPM models use a set of previous symbols in the uncompressed symbol sequence to predict the next symbol in the stream. PPM algorithms can also be used to cluster data into predicted groupings in cluster analysis. Specifically, to solve the zero-frequency probability issue which affects n-gram models, PPMs employ the so-called back-off smoothing technique, which allows to deal with the sparsity of the context. Formally, the probability distribution of an event e_i can be written as:

$$p(e_i|e_{(i-n)+1}^{i-1}) = \begin{cases} \alpha(e_i|e_{(i-n)+1}^{i-1}), & \text{if } c(e_i|e_{(i-n)+1}^{i-1}) > 1 \\ \gamma(e_{(i-n)+1}^{i-1}) p(e_i|e_{(i-n)+2}^{i-1}), & \text{otherwise} \end{cases} \tag{2.12}$$

where $\alpha(e_i|e_{(i-n)+1}^{i-1})$ is an estimate of the probability of an already seen

n-gram and $\gamma(e_{(i-n)+1}^{i-1})$ is the probability weight assigned to all the novel symbols in the current context in the training set [9].

2.3 Neural Networks Fundamentals

In this Section we introduce some of the most common deep learning algorithms used to address the problem of the modelling of sequential data, such as RNN, LSTM and Transformers.

NNs are networks composed of artificial neurons or nodes which aim at simulating the connection present in a biological brain in order to solve artificial intelligence problems. The connections of the biological neurons

are modeled in an artificial NN as weights between nodes. All inputs are modified by a weight and summed. This activity is referred to as a linear combination. Finally, an activation function, which is usually non-linear, controls the amplitude of the output. NNs, as machine learning models, rely on a training process to learn and improve their performances.

2.3.1 Feed Forward Neural Networks

Feed Forward Neural Networks (FFNNs) are an artificial NN implementation where connections between the nodes follow strictly a forward fashion. FFNNs were the first and simplest type of artificial NNs ever introduced. In this networks, the information moves in only one direction: from the input nodes, through the hidden nodes and to the output nodes.
A FFNN is composed of:

- Input layer, the first layer of the network.
- One or more Hidden layers placed in between the input and the output ones and sequentially connect with each other through sets of weights $W = \{w^l_{ji}\}$.
- Output layer, which is the last layer of the network and provides the prediction for the input variable.

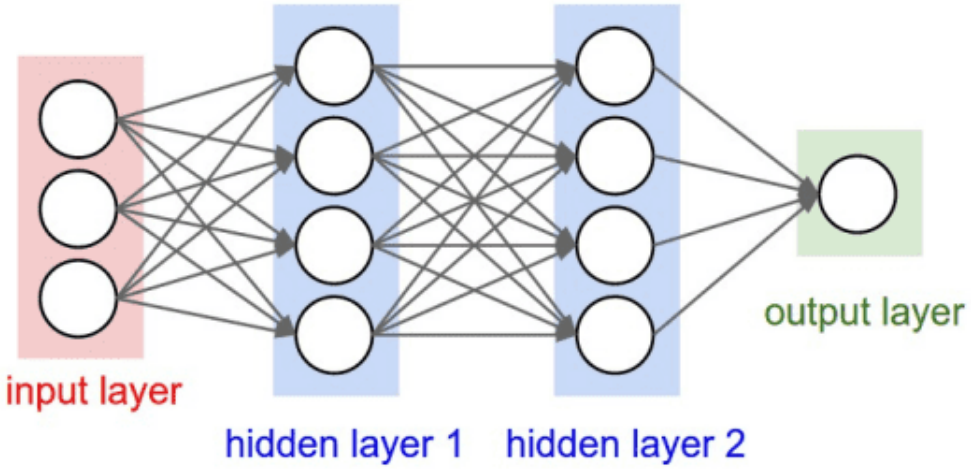

Figure 2.4: Feed Forward Neural Network with two Hidden Layers.

Given a set $\mathbf{x}$ of input variables, the activation values for each hidden neuron can be written as a linear combination of the input variables:

$$a_j = \sum_{i=1}^{N} w^{(1)}_{ji} x_i + b^{(1)}_j \tag{2.13}$$

where N is the dimension of the input vector, $j = \{1, , M\}$ and M is the number of neurons in each layer, $w_{ji}^{(1)}$ is a set of weights that connect the j^{th} neuron in the hidden layers to the i^{th} input variable and $b_j^{(1)}$ is the bias term of the j^{th} hidden neuron.

Neurons in each layer apply a specific non linear function to the linear combination of the inputs. Nonlinear activation functions allow such networks to compute nontrivial problems using only a small number of nodes, and such activation functions are called nonlinearities.

Some of the most used activation functions are the *ReLU* for regression tasks, *sigmoid*, *hyperbolic tangent* and *softmax* for binary and multiclass classification problems (Figure 2.6).

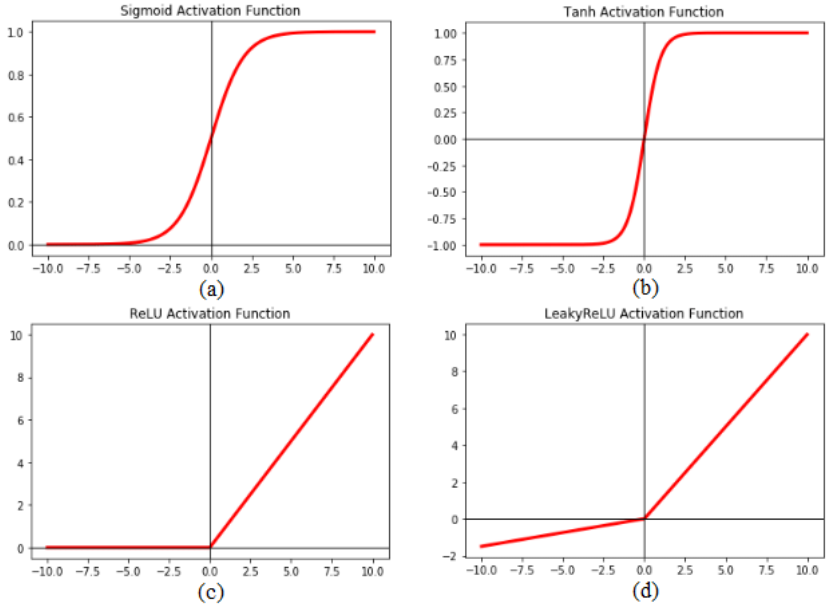

Figure 2.5: Some of the most common activation functions used in Neural Networks

The output of each node in the hidden layer can be expressed as:

$$h_j = H_j(a_j) \tag{2.14}$$

where H_j represents the activation function for the j^{th} neuron. The output of the output layer can then be expressed as a superposition of the linear combination and of the activation function.

$$y_k = G_k(\sum_{j=1}^{M} w_{ki}^{(2)} x_i + b_k^{(2)}) \tag{2.15}$$

where $k = \{1, .., K\}$, G_k are the activation functions of the output layer, $w_{jk}^{(2)}$ are the weights interconnecting the output layer and the previous one and $b_k^{(2)}$ are the bias terms.

It is possible to extend this model with an arbitrary number of hidden layers. Specifically, we refer to the number of layers as depth, and to the number of neurons in each layer as width of the model.

Optimization of FFNN is usually approached with a gradient descent technique. Gradient descent is an algorithm that aims at finding the minimum of a given cost surface by descending said surface following the direction opposite to the gradient itself. Optimization can became particularly difficult for cost functions which display multiple secondary minima. This situations are particularly effected from the starting point of the gradient descent algorithm.

Moreover, a number of generalization techniques can be employed in order to achieve better performances on previously unseen data. Some of the most common regularization methods are:

- *L1* and *L2* norms, which add a regularization term to the cost function which aims at keeping small the value of the weights.
- *Dropout*, which consist of randomly turning off some neurons in the network.
- *Early Stopping*, which automatically stops the training when there is no improvement in validation loss or accuracy.
- *Residual Connections*, which helps improve performances for very deep networks by adding shortcut connections between distant layers.

2.3.2 Recurrent Neural Networks

RNNs are a class of artificial NNs where connections between nodes form a directed or undirected graph along a temporal sequence. This allows the NN to display a temporal dynamic behavior. Derived from FFNNs, RNNs can use their internal memory state to process variable length sequences of inputs. This makes them applicable to tasks such or speech recognition, handwriting recognition and token prediction within a NLP framework. The process of optimization of RNNs is very similar to the one of FFNNs. Formally, given an input vector $\mathbf{x} = \{x_n, , x_n\}$, the hidden vector $\mathbf{h} = \{h_1, , h_j\}$ and the output vector $\mathbf{y} = \{y_1, , y_k\}$ are computed as shown in equations (2.16) and (2.17) for all time steps t from 1 to T

$$h_t = H(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{2.16}$$

$$y_t = W_{hy}h_t + b_y \tag{2.17}$$

where W represents the weight matrices of the input-to-hidden connections (W_{xh}), (W_{hh}) and (W_{yh}) represent respectively hidden-to-hidden recurrent connection and hidden-to-output connection, while b_h and b_y are bias vectors and H is the hidden layer activation function.

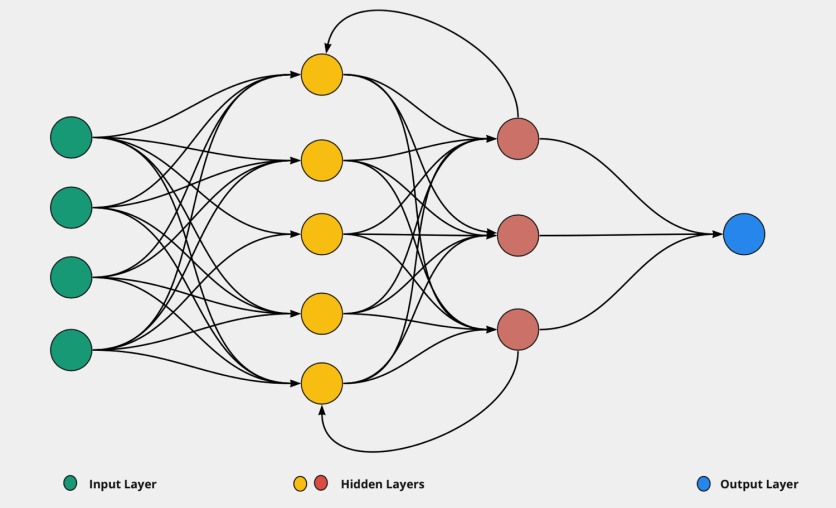

Figure 2.6: RNN with two Hidden Layers.

Even if RNN could theoretically deal with a context which spans over the whole history of inputs, they are in practice limited to a memory of a few steps because of the so-called *vanishing gradient* issue. This problem is due to Back Propagation Through Time (BPTT) as by going back through time steps the gradient converges to zero, discarding possible useful information.

2.3.3 Long-Short Term Memory

LSTMs are an RNN architecture introduced to overcome the vanishing gradient problem of RNNs. Unlike standard FFNNs, the LSTM architecture has feedback connections. It can process not only single data points, but also entire sequences of data. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. A detailed visualization of the LSTM is proposed in Figure 2.7.

LSTMs are well-suited for classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in time series. Relative insensitivity to gap length is an advantage of LSTMs over RNN, HMMs and other sequence learning methods in numerous applications.

2.3.4 Transfomer

The Transformer is a deep learning architecture introduced by Vaswani et al. in [2] in 2017 that adopts the mechanism of self-attention, weighting the significance of each part of the input data. It is used primarily in the field of NLP and in Computer Vision (CV).

Like RNNs, Transformers are designed to handle sequential input

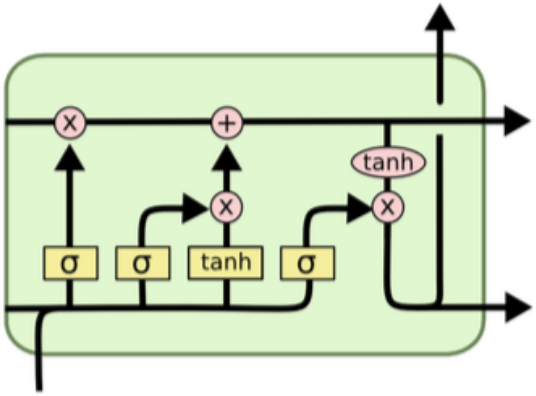

Figure 2.7: An LSTM unit with its internal gating mechanism.

data, such as natural languages, for tasks such as translation and text summarization. However, unlike RNNs, transformers do not necessarily process the data in order. In fact, the attention mechanism provides context for any position in the input sequence. For example, if the input data is a natural language sentence, the transformer does not need to process the beginning of the sentence before the end. Rather, it identifies the context that confers meaning to each word in the sentence. This feature allows for more parallelization than RNNs and therefore reduces training times.

Architecture

The Transformer model is composed of two main blocks:

- The Encoder, which is meant to encode the symbolic representation of the input sequence into a sequence of continuous representations.
- The Decoder, which produces an output sequence of symbols given the information present in the output of the encoder layer and translates the input sequences into a second language. The Transformer is an auto-regressive model so symbols in the output sequence are generated one at a time.

The generic stack of encoders and decoders of the transformer network is displayed in Figure 2.8.

Specifically, each encoder layer is composed of:

- A Multi-Head Self-Attention block, that aims at computing attention scores on each token in the input.
- A FFNN, that applies two linear transformations to its input with a ReLU activation function placed in between.

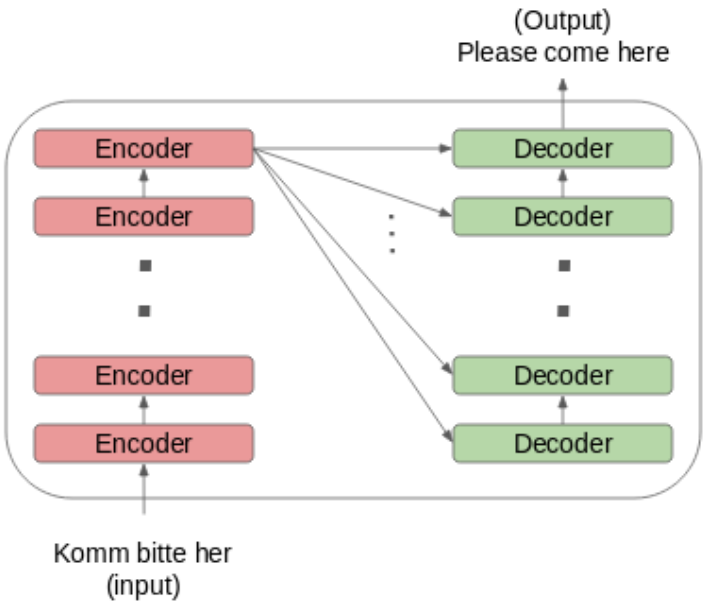

Figure 2.8: Encoder-Decoder architecture of a Transformer for language translation.

- A set of regularization techniques including residual connections, dropout and early stopping.
- Layer normalization, which aims at reducing training time by normalizing the activity of the neurons.

The decoder layers follows the same structure of the encoder but present two main differences:

- It uses the standard Multi-Head Self-Attention layers to compute attention scores on the output, while it applies a Masked Multi-Head Self-Attention block to the input of the decoder, to prevent the flow of information from future tokens.
- A linear transformation and a softmax activation function are used at the end of the decoder stack, to convert the output of the decoder to a probability distribution.

Scaled Dot-Product Attention

The transformer building blocks are scaled dot-product attention units. When a sentence is passed into a transformer model, attention weights are calculated between every token simultaneously. The attention unit produces embeddings for every token in context that contain information about the token itself along with a weighted combination of other relevant tokens each weighted by their attention coefficients.

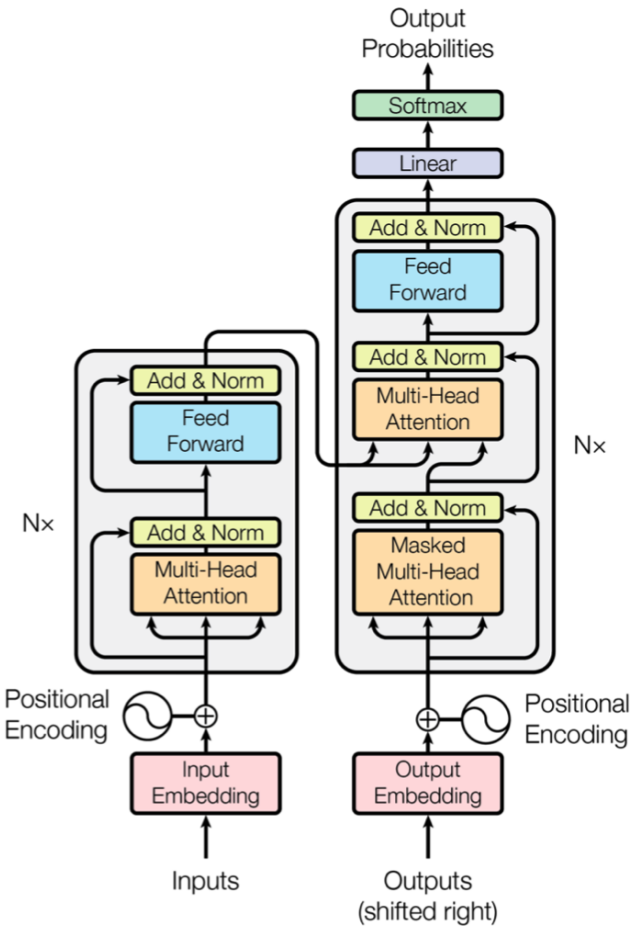

Figure 2.9: Stack of layers within the Transformer. (Figure taken from [2])

Every token is projected onto three vectors: *query*, *key* and *value*. Respective weight matrices W are learned during training in order to implement said projections. While calculating the attention on a particular word, a dot-product operation is calculated between the query vector and the key vector of each word [2]. Dot-product attention is scaled with $\frac{1}{\sqrt{d_k}}$ to compensate large dot-product values. The value vectors are weighted with weights from the dot product and then summed. In detail, the attention function implemented within the transformer model is computed as:

$$attention(\mathbf{q}, \mathbf{k}, \mathbf{v}) = softmax(\frac{\mathbf{q}\mathbf{k}^T}{\sqrt{d_k}})\mathbf{v} \tag{2.18}$$

where $\mathbf{q}$ is the vector of queries, $\mathbf{k}$ and $\mathbf{v}$ are vectors of keys and values and d_k is the number of dimensions of the model. The $softmax$ is applied to the dot product in order to obtain the weights that correspond to each input value.

For better results and for parallelization, multi-head attention is used. Each head learns a different attention distribution, similar to having mul-

tiple filters in a Convolutional Neural Network (CNN). Furthermore, a multi head and multi layer attention mechanism produces unique attention patterns for each layer and head, allowing to display how the model learns and interprets the input tokens at each level of the stack.

Embedding and Positional Encoding

Attention layers see their input as a set of vectors, with no sequential order. This model also doesn't contain any recurrent or convolutional layers. Because of this a *positional encoding* is added to the model to give it information about the relative position of the tokens in the sentence. The positional encoding vector is added to the embedding vector. Embeddings represent a token in a d-dimensional space where tokens with similar meaning will be closer to each other. But the embeddings do not encode the relative position of tokens in a sentence. So after adding the positional encoding, tokens will be closer to each other based on the similarity of their meaning and their position in the sentence, in the d-dimensional space. The positional encoding functions proposed by Vaswani et al. are computed as sine and cosine of different frequencies:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \tag{2.19}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \tag{2.20}$$

where pos is the position in the input sequence and i is the dimension. The result of the encoding is a vector of size d_{model} where each dimension corresponds to a sinusoid therefore the resulting vector implies a unique encoding for each word's position within a sentence.

2.4 Conclusive Remarks

In this chapter we briefly introduced all the main topics implied in the following discussion with a particular focus on functional harmony, classic time series forecasting techniques and the most important and widely used deep learning models for natural language processing. For the purpose of this thesis in fact, we trained a GPT-2 model and exploited its generative characteristics two approach a chord prediction task and a complexity estimation task.

3
State of the Art

In this chapter we introduce some of the most important studies proposed in literature regarding harmonic complexity estimation and chord prediction. In our work we in fact investigated both these fields implementing a Transformer algorithm for NLP with the purpose of employing it in a chord prediction task and we evaluated its performance by means of cross-entropy and accuracy. Furthermore, we investigated the relationship between the cross entropy and perceived complexity of a sequence of chords, under the hypothesis proposed in [1] that sequences that show a high cross entropy level are also perceived as more complex or unfamiliar by the human ear. Within this framework, we present in Section 3.1 some definitions of harmonic complexity drawn both from the information theory field and the functional harmony theory. In Section 3.2 we present some Machine Learning models which have been proposed to deal with the chord prediction task and in Section 3.3 we focus on Deep Learning algorithms applied to the same problem.

3.1 Harmonic Complexity

The definition and computation of complexity of a given piece of music refers to the research field known as Music Information Retrieval. As proposed in [27], in order to assess complexity on the various levels of which music is composed of, we can apply the so-called *divide at impera* approach, aiming at investigating the atomic components of music one at a time. We can identify these constitutive components as being melody, harmony and rhythm. Many definitions have been proposed concerning the concept of complexity and specifically of harmonic complexity. In this regard we can distinguish two main approaches proposed in litera-

ture: one based on music rules and one based on information theory. The first is related to the historical praxis that characterize a given musical culture and recognize as complex musical pieces the ones which exhibit uncommon characteristics with respect to the said cultural environment. The second one aims at defining complexity using a number of mathematical tools among which the most commonly used is information theory's *cross-entropy*.

Some examples regarding these two approaches are discussed in Section 3.1.1 and Section 3.1.2 respectively.

3.1.1 Functional Harmony Definition

Many studies have been conducted in the music field to give a definition of harmonic complexity. The most straightforward definition of harmonic complexity can be based on the tonal or functional harmonic framework which is the most widely spread within the western musical culture. With regard to this framework, tonal complexity can be analyzed by calculating the distance on the circle of fifths of each subsequent tonal context. This is the approach followed in [28], where the author introduces the concept of harmonic complexity by computing scores based on traditional musical rules. Specifically, he focus on:

- The rate at which chords are changed, also known as harmonic rhythm.
- The number of chords on the weak beats.
- The number passing notes which connect notes belonging to the same chord or to subsequent chords.
- The tonal distance of consecutive harmonies.

These scores are then employed to design a software model able to output a complexity score for each given harmonic sequence.

A similar approach is used in [6] where the authors propose a definition of harmonic complexity which relies on simulating the same process that a trained musician would employ to analyze a piece of music. Each transition between two consecutive harmonies is rated and then these values are combined to obtain the overall harmonic complexity of the sequence. Results show that this feature can be effectively used for the music genre classification task.

Weiss et al. in [3] again propose to link the complexity of each chord to the distance from the previous key centre using a visualization method based on the circle of fifths. A change in the tonal context of a small number of fifths can be associated with smooth changes, while large distances correspond to abrupt tonal changes.

In [29] Pachet addresses the problem of modelling jazz harmony. Specifically, even if he never explicitly mentions harmonic complexity, the

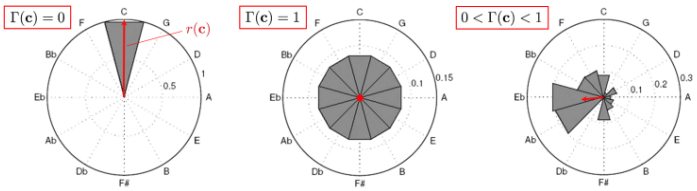

Figure 3.1: Complexity measure based on the circle of fifths. Figure taken from [3]

author investigates the concept of musical surprise. In fact, he defines surprise as linked to the presence or absence of previously heard peculiar harmonic patterns such as II-V-Is, turnarounds and blues forms. He also added to his scheme a set of substitution rules, thanks to which each chord can be converted into another one which brings the same kind of functional information. He then applies the Lempel-Ziv data compression algorithm and derives the so-called LZ tree, which is a representation where each node corresponds to possible continuation of the sequence. The results show that the system is able to learn jazz chord patterns and create tension in accordance with the jazz harmonic theory.

3.1.2 Information Theory Definition

Information theory is the scientific study of the quantification, storage, and communication of digital information. According to its rules, the amount of information necessary to transmit a a sequence of data x from a source to a receiver is formally quantified as:

$$I(x) = -log_2 p(x) \tag{3.1}$$

where $p(x)$ is the probability of the event and $I(x)$ is measured in bits. If an event is almost deterministic, the amount of information needed to transmit it would be close to zero. On the other hand, for rare or almost random events we need a lot of information in order to transmit it. A key measure in information theory is *entropy* which mathematically defines the amount of uncertainty in a probability distribution. Formally, its definition can be written as:

$$H(x) = -E[log_2 p(x)] \tag{3.2}$$

Distributions which are almost deterministic have very low entropy while distributions close to the uniform have high entropy. We can also define the *cross-entropy* of two distributions p and q over the same random variable x as the average number of bits needed to identify an event drawn from the set. Cross-entropy is formally defined as:

$$H(p, q) = -\sum_x q(x) log_2 p(x) \tag{3.3}$$

In most applications the distribution of q is not accessible so we have to rely on a Monte Carlo estimation of cross-entropy [30] as:

$$H_p(T) = -\frac{1}{T}\sum_{x \in T} log_2 p(x) \tag{3.4}$$

where T represents the test data and H_p is measured in bits/symbol.

Di Giorgi et al. exploited in [1] the measure of the cross-entropy to compute the complexity of harmonic sequences. Specifically, they define the complexity of the harmonic sequence as its average Monte Carlo cross-entropy. The authors also exploited the use of cross-entropy to evaluate the performance of their models on a chord prediction task, assuming that high cross-entropy would be related to highly complex sequences.

The same approach is used in [11], where the authors also exploit cross-entropy to measure how well their models are able to predict the continuation of chord sequences taken from their dataset.

3.2 Machine Learning for Chord Prediction

Several techniques have been proposed during the last years for addressing the task of chord prediction and automatic composition. Here we introduce the most relevant ones in the machine learning context.

3.2.1 Finite Context Models

A number of studies have been carried out in modelling harmony with n-grams.

In [13] the authors researched on a model able to predict 12-bars blues harmonic forms using controlled Markovian prediction. In particular, they trained a Markovian model on a collection of blues transcribed from Charlie Parkers Omnibook with an overall alphabet of 36 chords.

In [10] Sears evaluates finite context models with the use of PPM algorithms for automatic composition. The author computed the performance of his model as the average cross-entropy across the entire dataset. The results show that fixed-length context models generate the lowest average cross-entropy with respect to other models like LSTMs and RNNs, suggesting that context models are better suited for musical tasks.

Di Giorgi et al. in [1] trained a fixed-context model with PPM on a large database of chord annotations retrieved from *UltimateGuitar.com*. The authors used cross-entropy to evaluate the performance of the model on a chord prediction task.

3.2.2 Hidden Markov Models

Here we introduce some well known studies that address chord prediction using HMM.

In [1] Di Giorgi et al. trained a number of HMMs, varying the number of hidden states of the models. The results show that deeper models lead to a decrease of the overall cross-entropy and to more flexibility.

In [31] the author built a HMM to generate music sequences based on their probability distributions. Specifically, he trains his model with a custom database of chord annotations and generates synthetic sequences by prompting the model with the first chord. Results have shown that the model effectively outputs chord progressions which resemble the ones present in the training dataset.

In [12] the authors experimented with different architectures aiming at modelling complex harmonies. Specifically, they compared standard HMM with different numbers of hidden states and a feature-based Dynamic Bayesian Network (DBN) and showed that the overall predictive power of the HMM is worse than the one of n-gram models but HMMs are more robust to the problem of overfitting.

In [15] the authors worked as well on the chord prediction task using HMM, training their model on a corpus of 66 Beatles songs in an audio file format. In particular, they used HMM to model chord progression dependencies on the metrical structure.

3.3 Deep Learning for Chord Prediction

In this section we present some examples of studies in the field of chord prediction using NNs Models.

3.3.1 Recurrent Neural Networks

Here we present the main works conducted using RNNs with the aim of predicting the next chord in an harmonic sequence. The most widely used models to address this task are LSTM and GRU.

In [32] the authors trained a RNN model with a dataset retrieved from the Yamaha e-Piano Competition which contains MIDI files of piano performances of more than 1400 professional performers. Being this MIDI dataset generated from actual human performances introduces a great amount of dynamics and timing realism into the database. Results show that the model is able to produce piano performances which sound like coherent improvisations. Furthermore, the user can condition the output in a number of ways such as textural density, key, scale and dynamics.

In [1] Di Giorgi et al. used RNNs too to model harmony and predict the next token in a sequence of chords. Specifically, the authors trained a number of RNNs with the LSTM extension. They trained their models

experimenting in varying the number of neurons per layer and varying the number of hidden layers. The results show that bigger models lead to an overall decrease in the cross-entropy evaluated over the whole dataset. They also demonstrated that RNN models outperform PPMs and HMMs because of the larger size of their hidden space.

In [33] the author created a browser 2-D game based on a RNN model which automatically generates music in real time while the user is playing the game. Furthermore, the user can control two avatars at the same time in order to produce counterpoint-like musical results.

In [18] the authors propose XiaoIce Band, an algorithm able to predict both melody and harmony. They implemented an encoder-decoder GRU architecture trained with a dataset of more than 14000 pop songs transcribed in a MIDI file format which include various categories of instruments.

In [34] the author proposes an encoder-decoder architecture for the purpose of creating improvised-like piano performances controlled with a device with only 8 keys. A bidirectional LSTM encoder maps a sequence of piano notes to a sequence of controller buttons. A unidirectional LSTM decoder then decodes these controller sequences back into piano performances. After training, the encoder is discarded and controller sequences are provided by user input. The model is trained with the same set of piano performances used in [32] and results show that the algorithm is able to output performances which closely mimic the melodic contours entered with the controller.

Hild et al. in [20] introduced HarmoNet, an algorithm built as a compound of NNs for the purpose of generating the harmonization of a given melody in the style of Bach chorales. The models are trained with two separate sets of chorales containing either major or minor compositions. Results have been evaluated by a group of experts as being on the level of an improvising performer.

In [35] the authors propose a bundle of Ableton Live plugins based on various NN models among which RNNs. Specifically, the first release includes 5 apps called *Generate*, *Continue*, *Interpolate*, *Groove*, and *Drumify* with which the user can generate MIDI files straight into the Digital Audio Workstation (DAW). Specifically, *Continue* uses an RNN based structure to extend a melody or drum pattern prompted by the user, and *Interpolate* can combine features of from the user's inputs to produce new ideas or create musical transitions between phrases.

3.3.2 Transformer

In this section we provide some examples of studies which have been done in the field of chord prediction and sequence generation using transformer based neural networks.

OpenAI introduced in [22] a transformer-based model called MuseNet. MuseNet is a GPT-2 model which is able to create minutes long chorent

composition in a MIDI file format with 10 instruments and in a number of styles. The authors trained their model with hundreds of thousands of midi files retrieved from different sources such as ClassicalArchives, the MAESTRO dataset and BitMidi.

In [36] the authors created a custom Transformer based model for the modelling of piano performances. Specifically, they employed the concept of *relative self-attention* which explicitly modulates attention based on how far apart two tokens are. Relative self-attention also allows the model to generalize beyond the length of the training examples, which is not possible with the original Transformer model. Results show that this model outperforms the RNN and LSTM in producing coherent long sequences which convey a clear sense of structure.

In [37] the author proposes a GPT-2 architecture for the modelling of folk music. Specifically, he trained his network on the *Session* dataset including more than 200 000 music pieces. Results show that the model can produce satisfying music pieces including lyrics.

In [21] the authors propose an upgrade of the algorithm introduced in [38] with the aim of generating minute long coherent piano compositions. They trained their model with a database of 48 hours of piano music transcribed in a MIDI file format.

In [39] the author trained a Mini GPT-2 model on the Doug McKenzie Jazz Piano dataset with the purpose of generating improvised-like performances. He processed the MIDI files transforming them into a piano roll representation which includes a maximum of 128 MIDI notes for time intervals not shorter than a 16^{th} note. Results show that the model efficiently learned the basics of the jazz idiom.

In [23] the authors present the Jazz Transformer, a GPT model able to mimic jazz lead sheets. The aim of the project is to design a model which is able to generate both harmonic and melodic content at the same time as it is presented in common lead sheets. To evaluate the results, the authors proposed a subjective listening test to collect information about the perceptual qualities of the compositions and a set of quantitative metrics.

3.4 Conclusive Remarks

In this chapter we proposed an overview of the corpus of studies proposed to address the task of chord prediction and harmonic complexity estimation using a number of models and different datasets including audio files, lead sheets and MusicXML files. Specifically we distinguished between approaches based on information theory and approaches based on prior musical knowledge. For the purpose of this work we addressed the problem of chord prediction by training a GPT-2 model exploiting the chord symbols annotation contained in the dataset of the iRealPro app. In this framework we consider each chord symbol as a word in the jazz harmonic language and a sequence of chords as a sentence in

the language. This approach allows us to model the problem using NLP techniques. We then assessed the quality of our model by measuring the accuracy of the chord prediction with respect to the actual chord sequences. Furthermore, we linked the chord prediction problem to the one of complexity estimation and investigated the correlation between the ability of the model to predict the next chord in a sequence and the perceived complexity of that sequence.

4
Problem Formulation and Methods

As previously mentioned, Di Giorgi et al. in [1] demonstrated a close link between the perceived complexity of a sequence of chords and the cross-entropy of the sequence computed using a NLP model. In this thesis we replicated some of the experiments proposed in [1] with the aim of testing a more recent algorithm for language modelling on a newly proposed dataset of jazz chord annotations. Moreover, we assessed our model performance in terms of cross-entropy and accuracy. For the purpose of our work, we propose a data driven approach linking the problem of complexity estimation to the chord prediction task. Specifically, we trained a GPT-2 model on two versions of a newly proposed database of jazz chord annotations retrieved from the iRealPro app including more than 100 000 jazz chord sequences considering each chord as a word in the language of jazz harmony and each sequence of chords as a sentence. Our goal is to model the rules of jazz harmony directly from the data, without any prior harmonic assumptions. Specifically, we employed a version of the dataset including only chord changes and a second version where we included information about the duration of each chord by repeating the chord symbols for each quarter note it belonged to in the original music sheets.

Moreover, we analyzed how perceived harmonic complexity is influenced by the quality of the chord prediction computed by our model. In this chapter we describe how we approached the chord prediction task, the problem of harmonic complexity estimation and the model that we employed for our experiments.

4.1 Chord Prediction

Chord prediction is the task of estimating the most likely future chord in a sequence. Because of the inherent multi-layered nature of music the next chord in a harmonic sequence could be determined from the previous chord symbols, from the present and past melodic content and also from the duration of the chords themself. For the purpose of our experiment we decided to consider the harmonic information only, discarding any melodic information but we tried nonetheless to include information about the duration of each chord. As a matter of fact, we consider jazz harmony as a language of its own, thus we can think of chords as words in a language and to chord sequences as sentences in the language. This framework allows us to use NLP tools to model the underlying rules of jazz harmony.

In our experiments we trained our model on two versions of a newly proposed database of jazz chord sequences. Specifically, in one version we included only the chord changes, while in the other one we tried to include information about the duration of each chord by repeating its symbol for all the quarter notes it belonged to in the original scores.

In particular, we evaluated the results of our training both with objective metrics and with perceptual evaluations based on our prior knowledge of jazz harmony.

Figure 4.1: Simple outline of the chord prediction problem. Being the transformer an auto-regressive model, it produces one token at a time given all the previous ones and than add the generated symbol to the input in order to further generate new tokens.

4.2 Harmonic Complexity Estimation

The task of harmonic complexity estimation aims at evaluating the capability of the model to correctly estimate the perceived complexity of a given sequence of chords. This problem is extensively discussed in [1] where the authors validate the results with a perceptual listening test. In our work we adopt the same approach proposed in [1] evaluating the harmonic complexity estimated over a set of test sequences. First, the

goal of the experiment is to compute the harmonic complexity of a set of chord progression generated by the model. In particular, we prompt our model with the first chord of each sequence and let it predict the next 3 chords. Then, the harmonic complexity of each progression is computed as its average cross-entropy:

$$H_p(M) = -\frac{1}{M}\sum_{i=2}^{M} log_2 p(c_i) \tag{4.1}$$

where M represents the length of the progression and $p(c_i)$ corresponds to the models estimated probability for each chord c_i in the original sequence.

4.3 Neural Network Model

For the purpose of this thesis we implemented the original GPT-2 model. GPT-2 is a Transformer based algorithm presented by openAI in 2019 and it is a natural language model which aims at predicting the next word in a sentence given all the previous ones. This model differs in its structure from the traditional Transformer because it lacks the encoder side of the original model, as its purpose is to generate synthetic sentences by means of k-top sampling instead of translating sentences from a language to another. It has been demonstrated in [40] that GPT-2 outperforms other language models by learning different tasks in an unsupervised manner. OpenAI released in 2019 four types of GPT-2 characterized by a different number of parameters. In July 2021 OpenAI released a new model called GPT-3 which has a higher number of trainable parameters. In this work we propose an implementation of the small version of the GPT-2 model which we think is anyway big enough to address the chord prediction problem on our dataset.

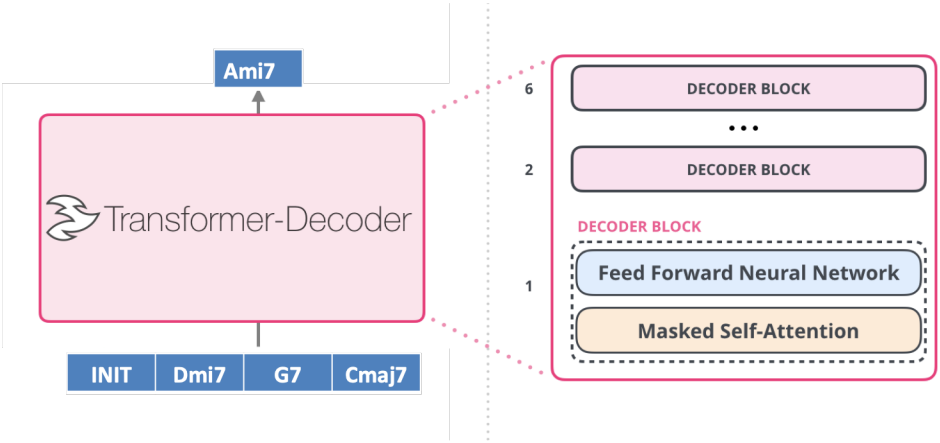

Figure 4.2: GPT-2 token generation based on past tokens. Each decoder block consists of a masked self-attention block and a FFNN layer. Figure taken from [4] and modified according to our case of study.

Structure

The structure of the GPT-2 is very similar to the one of the Transformer presented in [2] but it differs from it as being constituted of only decoder stacks. In fact, this model aims at generating new sentences in a given language while the original transformer is meant to translate sentences from a language to another one.

The structure of our model is displayed in figure 4.2. Each decoder layer is composed of:

- A masked self attention block, which allows the model to compute attention scores only for the previous tokens in the input sequence. This avoids the possibility for the model of accessing future tokens.
- A FFNN, which, as in the original Transformer, applies two linear functions with a ReLU activation in between.
- Residual connections applied both around the attention head and the FFNN block, followed by layer normalization.
- Input tokens are converted into vectors of d_{model} dimension through an embedding matrix. Moreover, positional information is incorporated into the input embeddings to give the model some information about the relative position of each token.

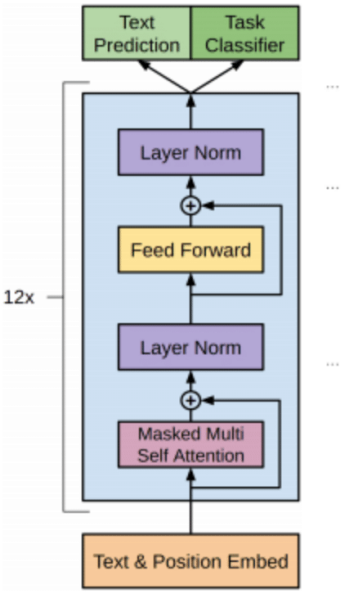

Figure 4.3: Visualization of a stack of decoder layers.

Masked Self-Attention

The main innovative aspect of GPT-2 with respect to the classic Transformer architecture relies on its attention mechanism. In fact, while both in [2] and in [41] the authors present a left to right self attention, in [40] the authors exploit the so-called *masked self attention* in each decoder layer. Masked Self-Attention is used in order to prevent the flow of information from future tokens while calculating the attention scores on the input sequence. A simple visualization of the difference between Self-Attention and Masked Self-Attention is illustrated in Figure 4.4.

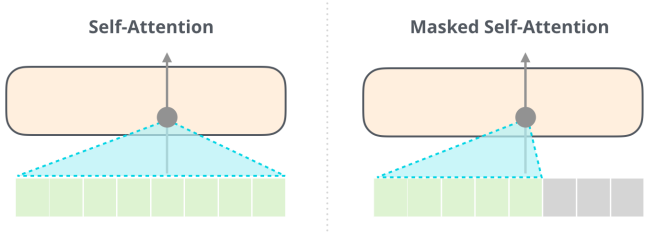

Figure 4.4: Self attention as used in the original Transformer model and masked self-attention as used for the Generative Pretrained Transformer.

k-top Sampling

Being GPT-2 an auto-regressive model, it generates one token at each time step on the basis of the previous tokens in the input sequence. The generated token is then added to the input for the generation of the next one and so on. When GPT-2 is used in a generative manner, the choice of the next token is governed by k-top sampling. This technique is used to inform the model on the subset of tokens among which the next one should be selected. If $k = 0$, the model will always select the most probable token. The choice of the optimal value of k can be difficult as small values of k can lead to the generation of generic and bland text, while large values could result in a loss of coherence and meaning [42]. Furthermore, a time varying value of k could be employed to generate sequences with a time varying cross-entropy profile.

Parameters

In this work we propose exactly the same implementation described in [43] thus we built our model as a stack of 12 decoder layers. The FFNNs in each decoder layer are characterized by two hidden layers, containing respectively 3072 and 768 hidden units. Furthermore, we computed attention with 12 distinct attention heads. Input embeddings and positional encodings are learned by the model during training using two embedding layers which are tuned to minimize a specific loss function with the chosen optimization method.

4.4 Conclusive Remarks

In this chapter we proposed a detailed overview of the experiments that we engaged for chord prediction and for complexity estimation and we described the model that we used to tackle those tasks.

5 Experiments and results

In this chapter we propose a detailed description of the database that we used and of the experiments that we carried out regarding the problems of chord prediction and harmonic complexity estimation.

5.1 Dataset

For the purpose of our work we propose a new database of jazz chord annotations retrieved from the iRealPro application. The iRealPro is an application created by Technimo and Massimo Biolcati in 2010 which as a matter of fact is a digitalization of the numerous volumes of the well known Real Book of Jazz, a collection of handwritten lead sheets of the so called Standards of jazz music. The iRealPro app is widely used within the jazz community all over the world and beside the chord annotations of a wide number of Jazz, Soul, Pop and Blues songs includes a number of features for practicing such as automatic comping in a variety of styles and automatic chords transposition. For the purpose of this thesis we download all and only the transcriptions provided by the author of the application himself in order to avoid using songs transcribed by other users which would probably contain more annotation errors. Files were exported in a MusicXML format from the application and a total number of more than 2500 songs were collected. We proceeded at processing the database in two ways: first we defined a finite set of allowed chord symbols based on our prior knowledge of the jazz harmony rules, then we separated each tune in its constitutive sections and transposed each obtained sequence in all the twelve keys.

5.1.1 Harmonic Framework Definition

Jazz harmony is mainly characterized by the extensive use of seventh chords. Within the context of western harmony chords are built as a superposition of 3rd intervals, so seventh chords are composed of three 3rd intervals. Using the canonical naming for intervals definition we can observe that within a seventh chord we can have 2 kinds of thirds (major and minor), 3 kinds of fifth (diminished, perfect, augmented) and 2 kinds of seventh (minor and major) for a total of 2 x 3 x 2 = 12 possible seventh chords built as a superposition of 3rd intervals. By enumerating the 12 possible tetrads constructed over one root note we obtain:

1. C Eb Gb Bb = half-dim —-> half-dim
2. C Eb Gb B = NONE
3. C Eb G Bb = mi7 —-> mi7
4. C Eb G B = mi(maj7) —-> mi
5. C Eb G# Bb = NONE
6. C Eb G# B = NONE
7. C E Gb Bb = 7(b5) —-> 7
8. C E Gb B = maj7(#11) —-> maj7
9. C E G Bb = 7 —-> 7
10. C E G B = maj7 —-> maj7
11. C E G# Bb = 7(#5) —-> 7
12. C E G# B = maj7(#5) —-> maj7

Three of these chords do not have any functional interpretation and can be immediately discarded while the remaining chords can be associated with five main chord families. Specifically, these five chord families are strictly the ones necessary to express the II-V-I cadence in both the major and minor keys. To these five categories we also had the diminished triad also associated with the diminished seventh chord obtaining in total 6 possible chord kinds:

1. major or major seventh (related to a I or IV degree in a major key)
2. minor or minor sixth (related to a I or IV degree in a minor key)
3. diminished or diminished seventh (related to the VII degree of a minor key)

4. dominant seventh (related to the V degree both in major and minor)

5. minor seventh (related to the II degree of a major key)

6. half diminished chord (related to the II degree of a minor key)

By using the letters from A to F in correspondence of the chord kinds we just defined, we can define the equivalence with the MusicXML chord kinds as displayed in Table 5.6.

5.1.2 Data Extraction

In order to create a database of coherent and short enough sequences we splitted the song at each rehearsal mark. For example the tune *Autumn Leaves* will generate three lines in the database. Furthermore we transcribed each song converting it to a 4/4 meter, thus writing a chord symbol for each quarter note in order to study dependencies due to harmonic rhythm as well. This was possible because even if the dataset contains tunes in 5/4 or 7/4 they actually never exhibit more than four different chords per bar.

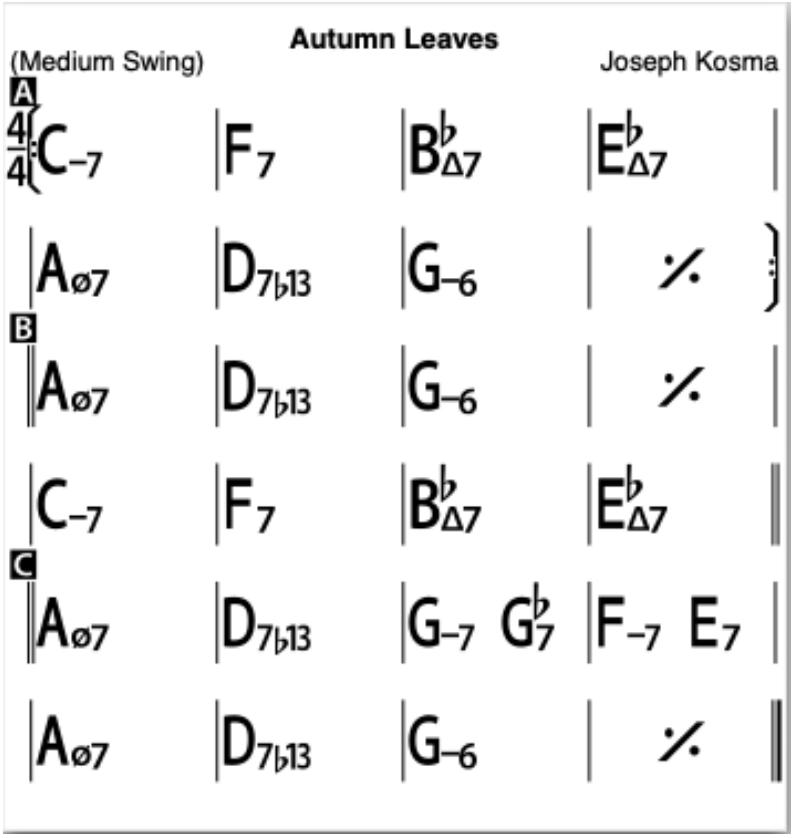

Figure 5.1: Chord changes of the tune "Autumn Leaves" as visualized in the iRealPro application.

The resulting database is a table of more than 8000 data where each line corresponds to a section of each tune and its columns are shown in Table 5.1.

Table 5.1: Columns of the database used during training.

TITLE	COMPOSER	STYLE	KEY	SECTION	CHORDS

The primary key of the database is then a combined key of TITLE + SECTION. Moreover, in the original database each tune is associated with either a major or minor key which we transcribed as always associated with the relative major key. For instance, Autumn Leaves is transcribed as being in B flat major even if it is actually in G minor. This key signature is anyway associated with the whole composition and not with the specific section. This creates an issue if we would like to transpose all the sequences in one key only, like for instance C major, because most of the tunes in the database have B sections which modulate with respect to the primary key of the tune.

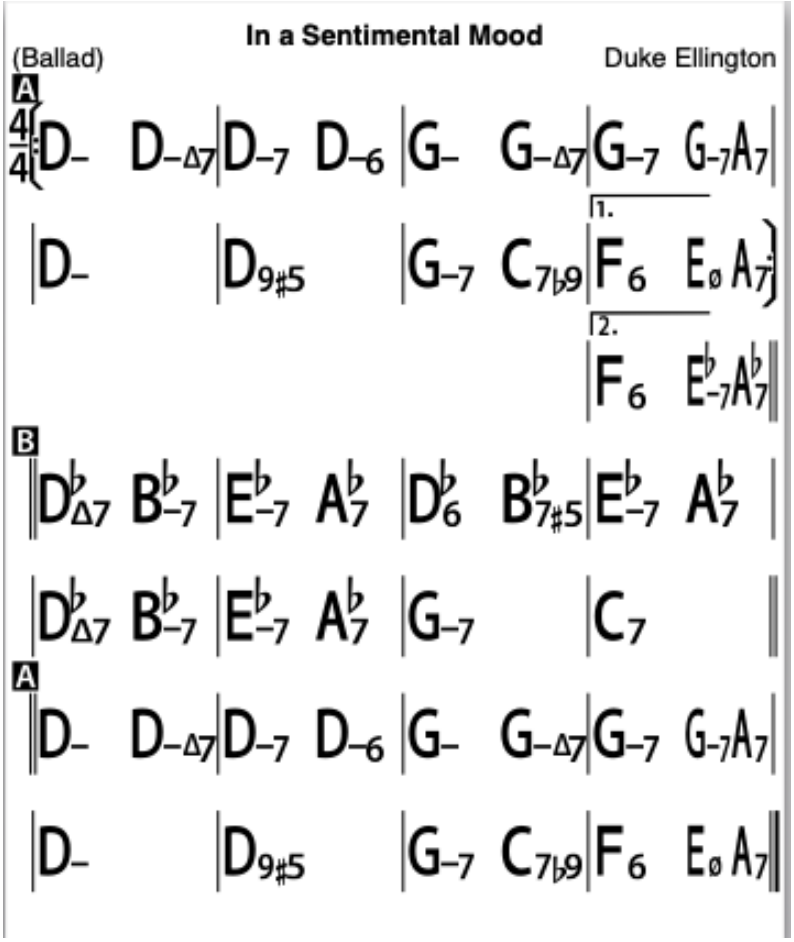

Figure 5.2: Chord changes of the tune "In A Sentimental Mood" as visualized in the iRealPro application.

For instance, *In A Sentimental Mood* by Duke Ellington and Billy

Strayorn is associated with the key of D minor or F major but its B section is clearly in D flat major. Furthermore, within the repertoire of jazz music are present also tunes affected by frequent modulation, like *26-2* by John Coltrane, or tunes which are constructed with a modal approach rather than a tonal one, like *Infant Eyes* by Wayne Shorter.

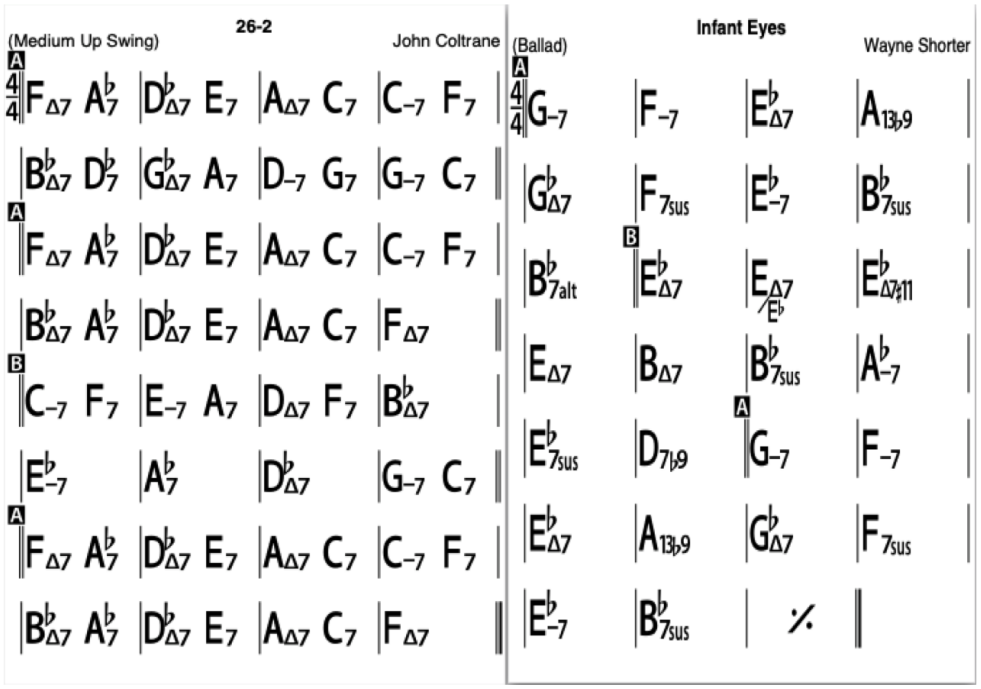

Figure 5.3: Chord changes of the tune "26-2" by John Coltrane and "Infant Eyes" by Wayne Shorter.

To overcome this issue and to mimic also the praxis of training of a professional musician, we transposed the whole database in all the 12 keys, thus obtaining more than 100 000 chord sequences.

Moreover, we interpolated the obtained database in order to have two different versions of it: one including the information about harmonic rhythm and one without. Since, as mentioned above, we transcribed the chords belonging to each quarter note in the original sequences we obtained the version without the harmonic rhythm information by simply removing chord symbols which are sequentially repeated.

5.2 Metrics

In order to assess the performance of our model in the chord prediction and complexity estimation tasks we employed, as already partially discussed in Chapter 4, cross-entropy and accuracy. Specifically, we applied the cross-entropy definition given in Equation 3.3, where q is the distribution in any corpus of chords and p is the distribution predicted by our model. Since we do not have access to q we employed the Monte Carlo estimation defined in Equation 3.4. For what concerns accuracy

we considered the number of chords that the algorithm is capable of correctly predicting when prompted with an initial chord. Formally, given the original M-chords progressions $\mathbf{c} = \{c_1, .., c_T\}$ in the test set T, and the sequences of predicted chords $\hat{\mathbf{c}} = \{\hat{c}_1, .., \hat{c}_T\}$, the accuracy can be computed as:

$$a = \frac{1}{M * T} \sum_{c \in T} \sum_{i=1}^{M} C(\hat{c}_i, c_i) \tag{5.1}$$

where $M * T$ is the overall number of predicted chords, c_i are the original chords in the test harmonic progressions T, $\hat{c}_i$ are the chords predicted by the model and $C(\hat{c}_i, c_i)$ is the indicator function defined as:

$$C(\hat{c}_i, c_i) = \begin{cases} 1, & \text{if } \hat{c}_i = c_i \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

Furthermore, to evaluate the nature of the relationship between either the complexity estimates produced by the model and the perceptual ratings obtained with the listening test or the relationship between cross-entropy and accuracy of the predictions we employ the Pearson correlation coefficient defined as:

$$r = \frac{\text{cov}(\mathbf{x}, \mathbf{y})}{\sigma_x \sigma_y} \tag{5.3}$$

where $\mathbf{x}$ and $\mathbf{y}$ represent the two measures we want to estimate the correlation of, cov is the covariance of the two variables and σ_x, σ_y are the respective standard deviations. The Pearson coefficient evaluates if a linear relation is present between the data and it ranges in the interval [-1,1], where the extremes indicate either a positive or negative strong correlation while values that are close to zero are associated to a very low correlation between the two inputs.

Moreover, in order to assess the nature of the relationship between the complexity estimates given by our model and the subjective ratings, we consider the average of the ratings for each chord progression in the test set over $S = 21$ participants. Specifically, we calculate the model's estimates according to Equation 4.1 and the average over the ratings for the j^{th} sequence is defined as follow:

$$\text{cmp}_j = \frac{\sum_{i=i}^{S} \text{cmp}_j^i}{S} \tag{5.4}$$

where cmp_j^i represents the complexity rating given by the i^{th} subject to the j^{th} sequence. In the following presentation we refer to the average subjects' ratings related to all the chord sequences as $\mathbf{cmp} = \{\text{cmp}_1, ..., \text{cmp}_J\}$ where $J = 20$ is the number of evaluated harmonic sequences.

Furthermore, following the procedure proposed in [1] we performed polynomial regression over our data, following the authors assumption that the relation between the model's estimates and the perceptual ratings can be described as a polynomial function. Considering the complexity estimates $\mathbf{h}$ computed by the GPT model as inputs, the polynomial function that provides the estimation of the average perceptual ratings $\hat{\text{cmp}}$ provided by the subjects can be expressed as:

$$\hat{\text{cmp}}(\mathbf{h}, \omega) = \sum_{k=1}^{K} \omega_k \mathbf{h}^k \tag{5.5}$$

where $k = \{1, 2, ..., K\}$ is the order of the polynomial function and ω_j are the regression weights. We experimented with different orders of polynomials from 1 to 6. Since we aim at finding the polynomial model that best fits the relationship between cross-entropy and perceptual ratings, we evaluate the quality of polynomial regressions of different orders using the R^2 coefficient of determination, thanks to which we can asses the quality of the regression. Specifically the R^2 is defined as:

$$R^2 = 1 - \frac{RSS}{TSS} \tag{5.6}$$

where RSS represents the Residual Sum of Squares and it is computed as:

$$RSS = \sum_{i=1}^{N} (\text{cmp}_i - \hat{\text{cmp}}_i)^2 \tag{5.7}$$

where cmp are the observed ratings related to complexity, $\hat{\text{cmp}}$ are the estimates of the complexity ratings obtained using polynomial regression, and N is the overall number of observed ratings.

TSS, instead, refers to the Total Sum of Squares defined as:

$$TSS = \sum_{i=1}^{N} (\text{cmp}_i - \overline{\text{cmp}})^2 \tag{5.8}$$

where $\overline{\text{cmp}}$ represents the mean of the complexity ratings. We evaluated the R^2 coefficient over 200 iteration on shuffle and split cross-validation, each time using 20% of the examples in the test set.

5.3 Training Details

In this section we list some details regarding the training process of our model.

The input tokens are embedded using an Embedding layer, which transforms each input chord into a 768-dimensional vector, containing the tokens' continuous representation. A second Embedding layer is then used to provide information about the position of each input chord in the

harmonic progression. We employed Adam optimizer, characterized by the following parameters related respectively to the 1^{st} and 2^{nd} momentum exponential decay rate, and to a constant for numerical stability: $beta_1 = 0.9$, $beta_2 = 0.98$ and $epsilon = 1e - 9$.. We provided the Adam optimizer with a custom learning rate scheduler, according to the formula in [2].

$$lrate = d_{model}^{-0.5} * min(stepNum^{-0.5}, stepNum * warmupSteps^{-1.5}) \quad (5.9)$$

$WarmupSteps$ are set to 40 000. The rate of change of the defined learning rate within the training steps is shown in Figure 5.4.

We set the early stopping criterion to automatically stop the training loop if the validation error does not decrease within 5 consecutive epochs.

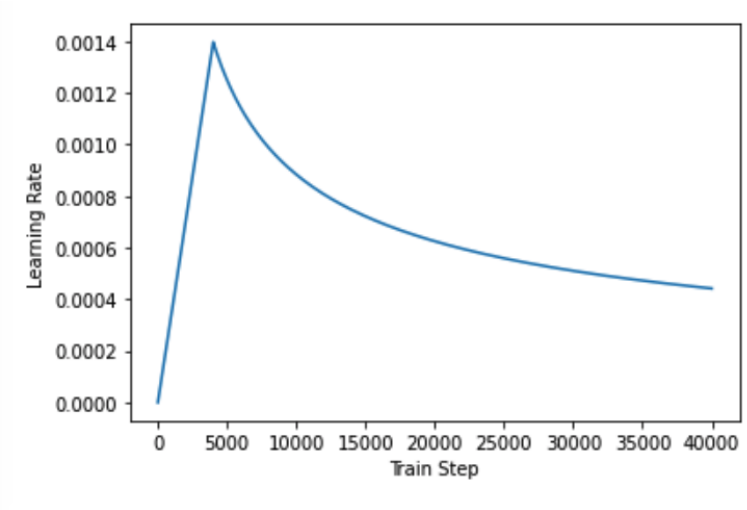

Figure 5.4: Custom learning rate scheduler.

Moreover, since our aim is to check whether the results obtained in [1] are valid also with our new database and since Di Giorgi et al. had a collection of harmonic progression which did not include information about the harmonic rhythm we choose to consider two separate versions of our database.

Specifically, we trained two models, the first using the originally transcribed version of the database which includes the information regarding harmonic rhythm and a second one which we trained with an interpolated version of the database which excludes harmonic rhythm. These two versions simply differ from each other for the fact that in the one containing the harmonic rhythm information each chord is repeated as many times as the quarter notes that it belongs to in the original music piece, while the second one considers a new chord symbol only when the harmony actually changes. We used the version without harmonic rhythm for repeating the same experiment conducted in [1] while we used both the versions of the database for our chord prediction experiments.

5.4 Experiments setup

In these next sections we introduce how we organized the perceptual test, how we generated the sequences employed during the listening experiment and the profile of the participants.

5.4.1 Sequences Generation

As mentioned in Chapter 4, the GPT-2 is an autoregressive model which generates the tokens of a sequence one after the other. Furthermore, each next token is generated according to the k-top sampling principle. This means that sampling using $k = 0$ will result in producing the most probable next token and that increasing the value of k would lead to the generation of sequences that are more unlikely as the value of k increases. Since we followed the same guidelines that the authors applied in [1] we generated our sequences based on their cross-entropy.

Specifically, we employed the model trained with the database with no harmonic rhythm and we generated a total of 20 sequences of tokens, each one produced with a value of k ranging from 0 to 4 so as to have different levels of cross-entropy. More precisely, we produced 4 sequences of 4 tokens for each value of k prompting the sequence with 4 different chord kinds out of the 6 that we defined each time in a random key.

We then generated a MIDI file for each sequence employing a custom voice-leading algorithm which given a sequence of token produces a 4 parts harmony midi file, trying to minimize the movement of the soprano part and fitting the remaining chordal notes in a close voicing fashion like shown in Figure 5.5.

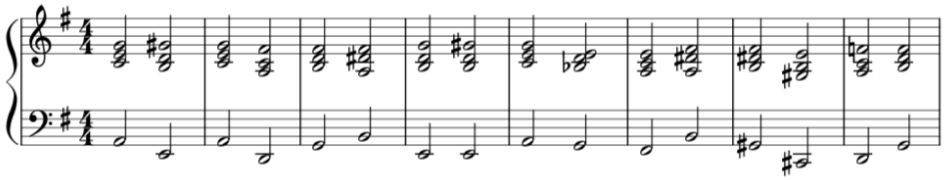

Figure 5.5: Example of voice leading of a sequence taken from the dataset where the movement of the soprano part is minimized.

We then employed a commercial piano library for generating the actual audio files.

5.4.2 Participants

Since we aim at verifying the relation discovered in [1] between cross-entropy and perceived complexity of an harmonic sequence we collected the perceptual complexity ratings from a set of subjects by means of a listening test. Specifically, we asked each participant to evaluate 20 chord sequences on five discrete levels of perceived complexity.

Furthermore, in order to better understand the musical background of the subjects involved in the test, we asked them to answer a specifically designed questionnaire which aims at evaluating their Gold-MSI. The questionnaire consists of 38 questions with 7 possible answers each. The results are then combined into 5 features [44]:

- Active musical engagement, which represents the degree to which the participants prioritize music-related activities.
- Perceptual abilities, which represents the accuracy of the musical listening skills.
- Musical training, which indicates the amount of formal music training received.
- Singing abilities.
- Emotional engagement, which represents the conscious use of music to alter emotional states.

The musical expertise of the subjects involved in the listening test is shown in Figure 5.6. The average Gold-MSI value for the subjects who participated in the experiment is 63.

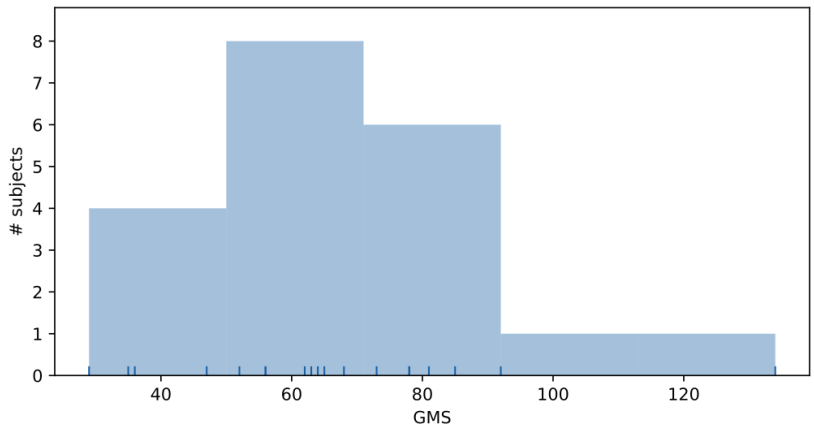

Figure 5.6: Gold-MSI of the subjects involved in the listening test..

5.5 Results

In this Section we present the outcome of our experiments regarding both the chord prediction problem and the complexity estimation task.

5.5.1 Chord Prediction

Here we present the results obtained using both versions of the dataset that we employed using objective metrics and by means of examples and observations on the generated chord progressions.

5.5.1.1 Evaluation of chord prediction task

Here we present the outcome of our training with objective metrics referred to both the versions of the database employed.

Dataset without harmonic rhythm

To evaluate the results of our training on previously unseen datas we calculated the average cross-entropy and accuracy on a test set obtaining the results shown in Table 5.2, which prove that the training had an effective result. Furthermore we evaluated the Pearson correlation coefficient between cross-entropy and accuracy of the sequences in the test set. Results show a strong negative correlation between these two measures as $r = -0.903$ for a $p-value \leq 0.001$, which implies that sequences with higher cross-entropy values lead to proportionally less accurate results.

Table 5.2: Loss, Accuracy and Pearson correlation coefficient over the test set for the training of the dataset without harmonic rhythm.

RESULTS ON TEST SET		
LOSS	ACCURACY	PEARSON COEF.
0.1121	0.9824	-0.903

Dataset with harmonic rhythm

For what concerns the dataset containing the harmonic rhythm information we exploited the same model evaluation techniques mentioned above. Specifically, to evaluate the model's predictive performance on previously unseen datas we calculated the measures of the loss and of the accuracy on a test set of about 1700 sequences obtaining the results displayed in Table 5.3. We again evaluated the correlation between loss and accuracy of the sequence in the test set employing the Pearson coefficient and noticing again a strong negative correlation.

Table 5.3: Loss, Accuracy and Pearson correlation coefficient over the test set for the training of the dataset with harmonic rhythm.

RESULTS ON TEST SET		
LOSS	ACCURACY	PEARSON COEF.
0.2738	0.9001	-0.8973

5.5.1.2 Perceptual Evaluation

Here we introduce some observations on the outcome of the chord prediction experiment for both the datasets employed based on our previous musical background.

Dataset without harmonic rhythm

In order to assess the musical quality of the predictions of our model we prompted the algorithm with various chord types letting it generate sequences of variable length. Results show that by sampling with a value of $k = 0$ the model efficiently generates common jazz harmonic patterns such as the II-V-I pattern, the turnaround, or the blues. Furthermore, the generated sequences present a certain degree of periodicity, as the one reported below:

Table 5.4: Examples of sequences generated with k-top sampling where k=0 and prompted with one random chord.

Cmaj7	Ami7	Fmaj7	Abmaj7	G7	Cmaj7
Fmi7	Bb7	Ebmaj7	Cmi7	Fmi7	Bb7
Abmaj7	Gmi7	Fmi7	Gmi7	Abmaj7	Gmi7
Dmi7(b5)	G7	Cmi6	Fmi7	Ebmi6	Dmi7(b5)

Moreover, it is particularly interesting to notice that by prompting the algorithm with a C7 chord and letting it generate the next tokens the model outputs exactly the chords of a C major blues. By plotting the probability distribution of each next token shown in Figure 5.7 we can clearly see that these chords are clearly much more likely than other ones. This could be due to the fact that the blues harmony is implied in many sequences contained in the database, even in those which are extracted from tunes which are not proper blues.

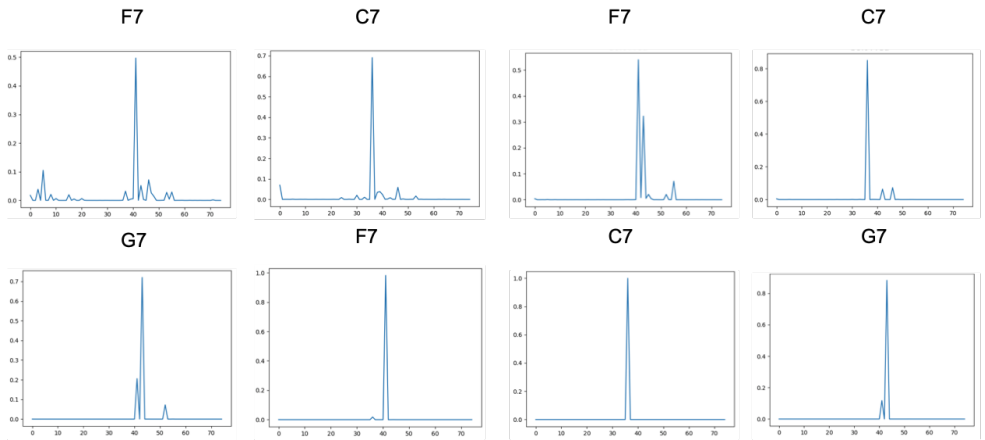

Figure 5.7: Probability distributions of each next token when prompting the model with a C7 chord and sampling with k=0

It is also interesting to notice that as the sequence goes on the token corresponding to $k = 0$ becomes more and more likely against the others as the more a sequence is composed the more it's direction becomes clear.

Dataset with harmonic rhythm

In this section we present some observations meant to understand if the model efficiently learned the concept of harmonic rhythm by letting it generate sequences prompted with various kinds of chords. The results show that by using a value of $k = 0$ throughout the whole progression the algorithm produces a sequence composed by only one chord. This was actually expected as in the database each chord is repeated multiple times for each quarter note that it belongs to. Anyway, by plotting the distributions of each next token shown in Figure 5.8 we can see that the tokens corresponding to a value of $k = 1$ become more likely in correspondence of an even number of tokens.

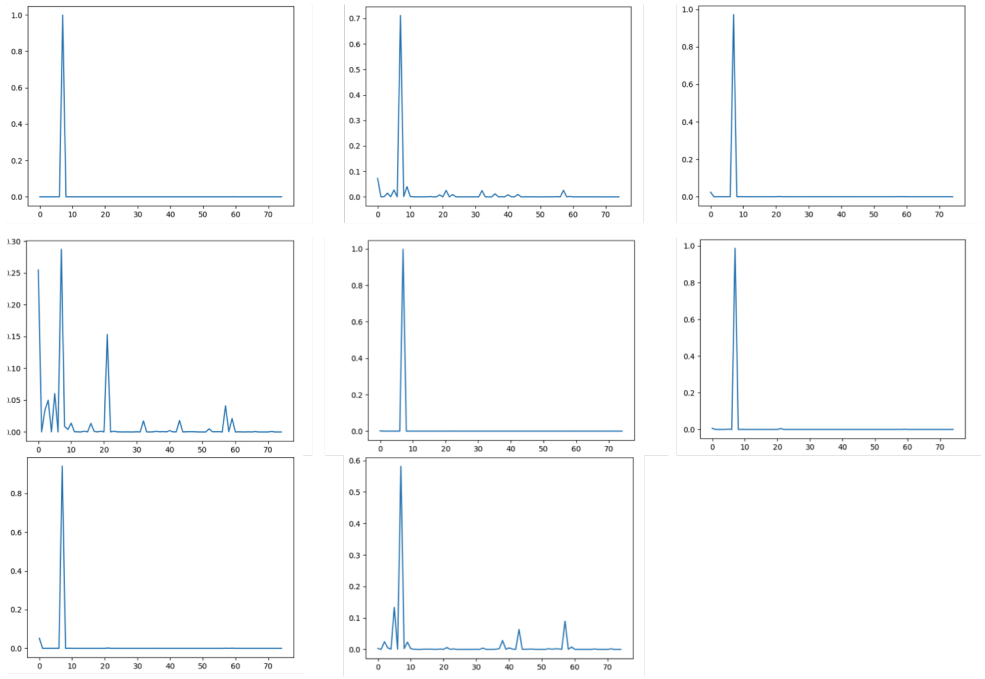

Figure 5.8: Probability distributions of each next token when prompting the model with a Gmaj7 chord and sampling with $k = 0$

From the Figure above we can see that even if the token corresponding to a value of $k = 0$ is always Gmaj7, the token related to the value of $k = 1$ change its likelihood through time, specifically increasing the value of its probability in correspondence of an even number of chords. From this observation we can suppose that our model learned effectively the concept of harmonic rhythm.

To further explore this aspect we prompted the model with the first two bars of a C major blues, i.e. C7,C7,C7,C7,F7,F7,F7,F7 and let the

model predict the continuation of the sequence by picking tokens corresponding to the value of $k = 0$. Quite surprisingly the algorithm outputs the chord of a C major blues changing the chords exactly where the changes are supposed to be in a standard blues form. The output of the model with each token distribution is visualized in Figure 5.9.

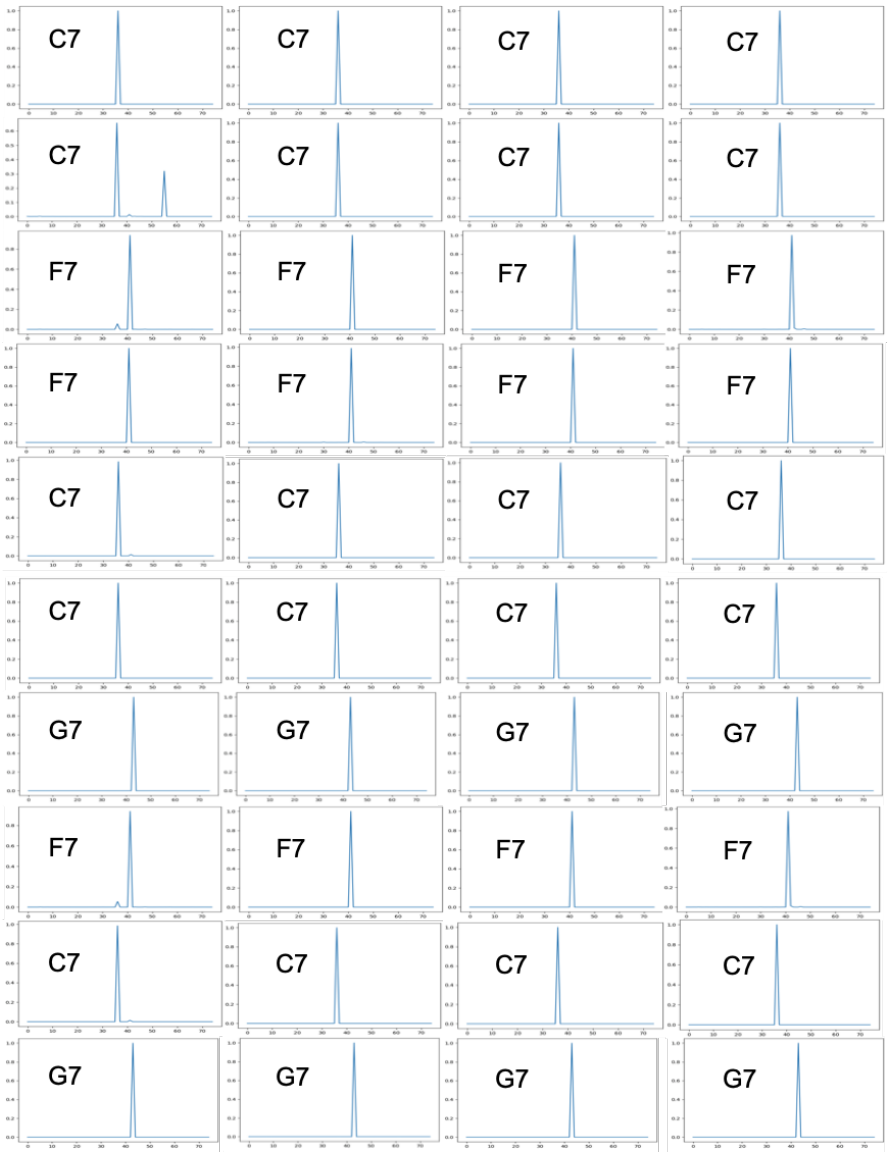

Figure 5.9: Probability distributions of each next token when prompting the model with the first two bars of a C major blues and sampling with $k = 0$

This result clearly shows that the model is able to produce sequences with a coherent harmonic rhythm behaviour even over a quite long context. Moreover, the F7 prompt as the second bar corresponds to the token related with a value of $k = 1$ in that specific point of the sequence. This suggests the possible existence of cross-entropy envelopes across the sequence which could be exploited for generating chord progressions which exhibit a similar complexity behaviour through time. This aspect could

be also exploited for studying whether chords with high cross entropy occur in specific points of some sequences, determining the narrative aspects of the sequences themself.

5.5.2 Complexity Estimation

Here we present the results obtained in the complexity estimation task and we propose some interpretations of the outcomes.

The obtained relationship between the model's estimates and the perceptual ratings is plotted in Figure 5.10 where the blue line represents the ratings mean and the green area represents the standard deviation.

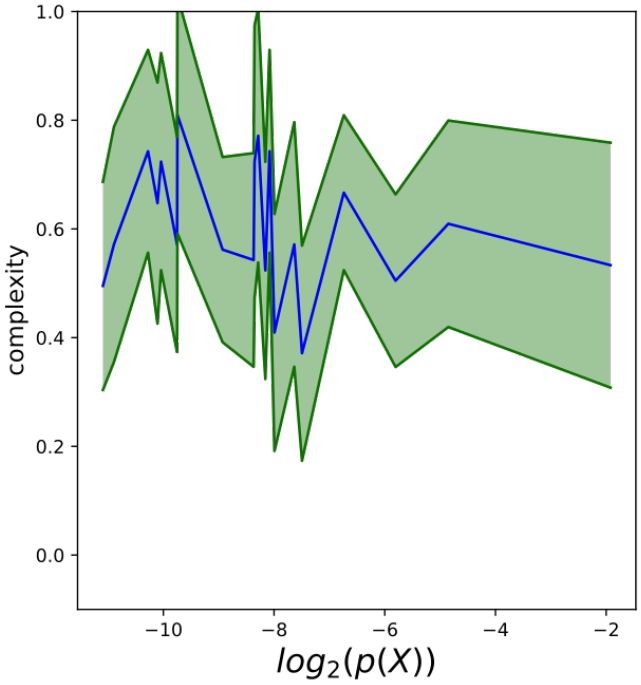

Figure 5.10: Relationship between complexity estimates and perceptual ratings obtained with our listening test.

From the plot we can see that we actually got some very noisy results which do not really overlap with the considerations made in [1]. Infact the supposed relationship should appear as a highly correlated monotonically decreasing curve, while our data look almost random.

To further investigate our results we calculated the Pearson correlation coefficient between the model's estimates and the perceptual ratings. The obtained value of the Pearson coefficient is displayed in Table 5.5 which mathematically shows that almost no correlation is present between the perceptual ratings and the model's estimates.

Moreover, following the procedure proposed in [1] we performed polynomial regression over our data, following the authors assumption that the relation between the model's estimates and the perceptual ratings

Table 5.5: Pearson coefficient of correlation between the model's estimates and the perceptual ratings.

PEARSON COEF.
-0.018

can be described as a polynomial function. The polynomial regression results are displayed in Figure 5.11 where emerges that the best linear model for fitting our data is the one of the first degree.

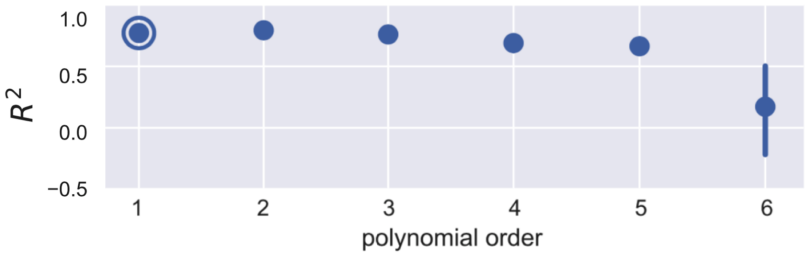

Figure 5.11: R^2 scores for different orders of polynomial regression models.

Considerations

As shown in the previous Section, our results do not match with the hypothesis theorized in [1]. This can be due to many different reasons, both related to how we implemented the listening test and to the dataset. Here we try to discuss some of them.

First of all, the experiments carried out by Di Giorgi et al. were meant to investigate how Tonal complexity is perceived while we extended the problem beyond the tonal context. In fact, as already discussed in Section 5.1, we proposed a newly retrieved database which is intrinsically non tonal and because of that we had to transpose it in all the twelve keys. This results in generated sequences which are both modulating within the sequences themself and that span over the whole possible keys domain. This can lead to a harder complexity evaluation task for non musically trained listeners both because of the modulating aspect of the sequences and because of their varying tonal or modal context.

Furthermore, the Jazz harmonic language is quite more sophisticated than the Pop and Rock ones, implying an extensive use of seventh chords which might result in difficulties for the listeners in decoding what is actually a common sequence against an uncommon one. Moreover, the Jazz genre itself is way less widely diffused than the Pop one, so listeners have, on average, less experience of listening to it.

Moreover, we generated the sequences with k-top sampling using values of k ranging from 0 to 4, which might have led to the synthesis of sequences which are not too different from each other. Using more spread

values of k could have led to generating sequences with more clear and distinct perceived complexity values.

Last, the evaluation of the Gold-MSI of the listeners showed that the average level of the participant of our test is lower than the one retrieved in [1]. This could have led, united with the previous considerations, to data which might not be reliable for such an investigation. Given our particular context, we presume that assessing the perceptual ratings of professionally trained musicians only would result in data which would show a higher correlation with the model's estimates.

5.6 Conclusive Remarks

In this chapter we presented the experiment that we conducted for the problems of chord prediction and complexity estimation. We analyzed the different chord prediction results that we obtained on two different versions of our database pointing out how our model effectively succeeded in learning the basic rules of jazz harmony and the basic rules of harmonic rhythm. Furthermore we evaluated the ability of our model to correctly predict the perceived complexity of a sequence of chords by means of a listening test. Despite a quite strong correlation between the model's estimate and the perceptual ratings was shown in [1] we didn't find the same results. We discussed why this could have happened within the context of our experiment and we suggested how to further explore this link in the field of jazz harmony.

Table 5.6: Equivalence between our alphabet and the MusicXML chord types.

Triads	
major (major third, perfect fifth)	A
minor (minor third, perfect fifth)	B
augmented (major third, augmented fifth)	A
diminished (minor third, diminished fifth)	C
Sevenths	
dominant (major triad, minor seventh)	D
major-seventh (major triad, major seventh)	A
minor-seventh (minor triad, minor seventh)	E
diminished-seventh (diminished triad, diminished seventh)	C
augmented-seventh (augmented triad, minor seventh)	D
half-diminished (diminished triad, minor seventh)	F
major-minor (minor triad, major seventh)	B
Sixths	
major-sixth (major triad, added sixth)	A
minor-sixth (minor triad, added sixth)	B
Ninths	
dominant-ninth (dominant-seventh, major ninth)	D
major-ninth (major-seventh, major ninth)	A
minor-ninth (minor-seventh, major ninth)	E
11ths	
dominant-11th (dominant-ninth, perfect 11th)	D
major-11th (major-ninth, perfect 11th)	A
minor-11th (minor-ninth, perfect 11th)	E
13ths	
dominant-13th (dominant-11th, major 13th)	D
major-13th (major-11th, major 13th)	A
minor-13th (minor-11th, major 13th)	B
Suspended	
suspended-second (major second, perfect fifth)	A
suspended-fourth (perfect fourth, perfect fifth)	+7E
Functional sixths	
Neapolitan	NONE
Italian	NONE
French	NONE
German	NONE
Other	
pedal (pedal-point bass)	NONE
power (perfect fifth)	NONE
Tristan	NONE

6
Conclusions and Future Works

In this thesis we investigated the ability of the GPT-2 language model to automatically compose chord sequences in the context of Jazz harmony and its capability of producing correct estimates for the perceived complexity of a given chord progression.

As a matter of fact, we modelled jazz harmony as a language of its own, where each chord represents a word in the language and each sequence is considered as a sentence. For training our model we used a newly proposed database of jazz chord annotations retrieved from the iRealPro application proposed by Technimo and Massimo Biolcati in 2010 which is a widely diffused practicing tool within the jazz community all over the world. We formalized how we transcribed this database using an alphabet of $12 * 6 = 72$ chord symbols retrieving more than 100 000 chord sequences.

We evaluated the performance of our model in the chord prediction task both by means of objective metrics and against our prior knowledge of harmony and in the complexity estimation task by means of a perceptual experiment. For what concerns the chord prediction problem, we compared the results obtained from the training of our model on two different versions of the dataset, one containing only chord annotations and one containing information on the harmonic rhythm as well. We showed that the model effectively learned the fundamentals of the Jazz harmonic language and of the harmonic rhythm patterns both by means of objective results and musical examples.

Considering the complexity estimation task, we devised a listening experiment in which a group of listeners rated the perceived complexity of 20 chord sequences generated by the model on the basis of their cross-entropy. We investigated the relationship between the perceptual ratings

and the estimates of the algorithm and, despite a quite strong negative correlation was demonstrated in [1], we didn't find any meaningful relation between these two quantities. We proposed some reasoning about the outcome of this test and suggested some future options for more experiments.

Even if the model proved to have learned to compose sequences that are quite resembling of the original sequences in the database some more options remain open for future investigations.

First, the cross-entropy envelopes which appear within each single sequence can be further explored investigating whether chords with high cross-entropy correspond to specific points in the harmonic grid. In particular this aspect could be exploited for generating sequences based on cross-entropy envelopes of a prompted sequence which is liked by the user. This could represent a quite interesting composition assistant tool.

Moreover, the cross-entropy measure could be exploited for a classification task aiming at classifying the composer of a given sequence of chords or at generating a new sequence in the style of a given composer. In our opinion, even if the database includes a style attribute for each tune, the classification task conducted on this attribute would not lead to good results, since the annotated style is mainly referred to the groove style of the original interpretation of the tune, rather than on the style of the song itself.

For what concerns the complexity estimation task, it would be interesting to repeat the test with a higher number of subjects with a stronger musical background. Furthermore, the experiment could be also conducted on professionally trained jazz musicians using chord annotation instead of audio files. In our opinion this would probably lead to a better correlation with the cross-entropy estimates, since the dataset used for our training is retrieved from sequences which are meant to be read rather than heard. In fact, the quality of the perceptual ratings using audio files can be highly biased from the particular voice-leading algorithm that we employed.

Finally, it would be interesting to repeat our work using the recently proposed GPT-3 model.

Bibliography

[1] B. Di Giorgi, S. Dixon, M. Zanoni, and A. Sarti, "A data-driven model of tonal chord sequence complexity," *IEEE/ACM Transactions on Audio, Speech, and Language processing*, vol. 25, pp. 2237–2250, 2017.

[2] A. Vaswani, N. Shazeer, J. parmar, L. Uszkoreit, A. Jones, L. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[3] C. Weib, J. Balke, J. Abeber, and M. Muller, "Computational corpus analysis: a case of study on jazz solos," *ISMIR*, pp. 416–423, 2018.

[4] J. Alammar, "The illustrated transformer," 2018.

[5] D. Temperley, "The cognition of basic musical structures," *MIT press*, vol. 25, pp. 2237–2250, 2004.

[6] L. Marsik, J. Porkorny, and M. Ilcik, "Towards a harmonic complexity of musical pieces," *Proceedings of the Annual International Workshop on Databases, Texts, Specifications and Objects (DATESO)*, 2014.

[7] C. Weiss and M. Muller, "Quantifying and visualizing tonal complexity," *Conference on interdisciplinary Musicology (CIM 2014)*, 2014.

[8] C. Weiss and M. Muller, "Tonal complexity features for style classification of classical music," *2015 IEEE International Conference on Acoustics, Speech and Signal processing (ICASSP)*, 2015.

[9] I. Witten and T. Bell, "The zero frequency problem: estimating the probabilities of novel events in adaptive text compression," *IEEE Transactions on information theory*, vol. 37, 1991.

[10] D. Sears, F. Korzeniowski, and G. Wildmer, "Evaluating language models of tonal harmony," 2018.

[11] F. Korzeniowski, D. Sears, and G. Widmer, "A large scale study of language models for chord prediction," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 91–95, 2018.

[12] M. Rohrmeier and T. Graepel, "Comparing feature-based models on harmony," *Proceedings of the 2008 ACM conference on Recommender systems*, pp. 179–186, 2012.

[13] F. Pachet and P. Roy, "Markov constraints:steerable generation of markov sequences," *Constraints*, vol. 10, pp. 148–172, 2011.

[14] R. Whorley, A. Wiggins, C. Rhodes, and M. Pearce, "Multiple viewpoint systems: time complexity and the construction of domains for complex musical viewpoints in the harmonization problem," *Journal of New Music Research*, vol. 42, pp. 237–266, 2013.

[15] H. Papadopoulos and G. Peeters, "Simultaneous estimation of chord progresion and downbeat from an audio file," *IEEE International Conerence on Acoustics, Speech and Signal Processing*, pp. 121–124, 2008.

[16] A. Van Der Merwe and W. Schulze, "Music generation with markov models," *IEEE Multimedia*, vol. 18, pp. 78–85, 2010.

[17] O. Peracha, "Improving polyphonic music models with feature-rich encoding," *IEEE/ACM Transactions on Audio, Speech, and Language processing*, 2019.

[18] H. Zhu, Q. Liu, N. Yuan, C. Qin, J. Li, K. Zhang, F. Zhou, F. Wei, Y. Xu, and E. Chen, "Xiaoice band: a melody and arrangment generation framework for pop music," *proceedings of the 24th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining*, pp. 2837–2846, 2018.

[19] H. Lim, S. Rhyu, and K. Lee, "Chord generation from symbolic melody using blstm networks," 2017.

[20] H. Hild, J. Feulner, and W. Menzel, "harmonet: a neural net for harmonizing chorales in the style of j. s. bach," *Advanced in Neural Information Processing Systems*, 1991.

[21] Y. Huang and Y. Yang, "Pop music transformer: generating music with rhythm and harmony," 2020.

[22] C. Payne, "Musenet," *OpenAI*, 2019.

[23] S. Wu and Y. Yang, "The jazz transformer on the front line: exploring the shortcomings of ai-composed music through quantitative measures," *IEEE/ACM Transactions on Audio, Speech, and Language processing*, 2020.

[24] R. Hyndman and G. Athanasopoulos, "Forecasting: principles and practice," 2018.

[25] C. Holt, "International journal of forecasting," *Managment Science*, vol. 20, pp. 5–10, 2004.

[26] P. Winters, "Forecasting sales by exponentially weighted moving averages," *Management Science*, vol. 6, pp. 324–342, 1960.

[27] F. Foscarin, "Chord sequences: evaluating the effect of complexity on preference," 2017.

[28] S. Streich, "Music complexity: a multi-faceted description of audio content," 2006.

[29] F. Pachet, "Surprising harmonies," *International Journal of Computing Anticipatory Systems*, 199.

[30] P. De Boer, D. Kroese, S. Mannor, and Y. Rubinstein, "A tutorial on cross-entropy method," *Annals of operations research*, vol. 134, pp. 19–67, 2005.

[31] A. Osipenko, "Markov chain for music generation," *Towards Data Science*, 2021.

[32] I. Simon and S. Oore, "Performance rnn: Generating music with expressive timing and dynamics," *Magenta Blog*, 2017.

[33] V. Thio, "Runn," *Magenta Blog*, 2017.

[34] C. Donahue, "Pianogenie," *Magenta Blog*, 2018.

[35] A. Roberts, C. Kayacik, C. Hawthorne, D. Eck, J. Engel, M. Dinculescu, and S. Nørly, "Magenta studio: Augmenting creativity with deep learning in ableton live," in *Proceedings of the International Workshop on Musical Metacreation (MUME)*.

[36] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, C. Hawthorne, A. M. Dai, M. D. Hoffman, and D. Eck, "Music transformer: Generating music with long-term structure," *arXiv preprint arXiv:1809.04281*, 2018.

[37] G. Branwen, "Gpt-2 folk music generation," *Magenta Blog*, 2016.

[38] C. Huang, A. Vaswani, M. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. Dai, D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[39] V. Pham, "Generating jazz music using gpt and piano roll encoding methods," *Towards Data Science*, 2021.

[40] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog*, 2019.

[41] J. Devlin, M. Chang, K. lee, and K. Toutanova, "Bert: pre-training of deep bidirectional transformers for language understanding," 2018.

[42] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious cas of neural text degeneration," *arXiv preprint arXiv:1904.09751*, vol. 25, pp. 2237–2250, 2019.

[43] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Imrpoving language understanding by generative pre-training," *IEEE/ACM Transactions on Audio, Speech, and Language processing*, 2018.

[44] D. Mullensiefen, B. Gingras, L. Stewart, and J. Musil, "Goldsmiths musical sophistication index (gold-msi) v1.0: technical report and documentation revision 0.3," *London: Goldsmiths, University of London*, 2013.