Reranking CCG Parser for Jazz Chord Sequences

ONG, Hong Xuan

School of Information Science
Japan Advanced Institute of Science and Technology
December, 2015

Master's Thesis

Reranking CCG Parser for Jazz Chord Sequences

1310251 ONG, Hong Xuan

Supervisor : Associate Professor Nguyen Minh Le
Main Examiner : Associate Professor Nguyen Minh Le
Examiners : Professor Satoshi Tojo
Associate Professor Kiyoaki Shirai

School of Information Science
Japan Advanced Institute of Science and Technology

November, 2015

Contents

This dissertation was prepared according to the curriculum for the Collaborative Education Program organized by Japan Advanced Institute of Science and Technology and Vietnam National University - Ho Chi Minh City.

Acknowledgements

At first, I would like to give thanks to my advisor, Associate Professor Nguyen Minh Le, who gives me direction through this research and completion of this thesis from theory to application. Second, I would like to thanks professor Satoshi Tojo, who supports me during this master course.

At the same time, I would like to give thanks to the teachers in the Information Science department at University of Science and Japan Advanced Institute of Science and Technology, who equipped me the necessary basic knowledge so that I can fulfill this thesis.

Finally, I thanks to my family and my friends who are always beside me, gives me a great inspiration.

List of Figures

List of Tables

Chapter 1

Introduction

In Artificial Intelligence (AI), we attempt to use mathematics models for representing and formalizing knowledge so that the computer able to adopt its computational ability to execute useful tasks. In recent year, with the advancement of science and technology, we have achieved a certain success in this field. Particularly, we have applications such as face recognition and image processing in Computer Vision area, voice synstudy or voice recognition in Signal Processing field, and do text mining with vast amount of unstructured data set for extracting practical information which is unable to handle by human.

Besides the efforts of making computer performed some abilities as human such as seeing, hearing, or understanding language in documents, music is also a compelling field that worthily consider to study. Achievements in music with combination of other fields such as psychology, linguistics, and cognitive science have been proven in practice [21], [22], [24], [28], [32], [33]. In addition, if we could generate cognitive model of how human conceive the music by using language of mathematics [3], [36], [35], [34], we could simulate mechanism of these processes so that we can make the computer able to handle more helpful functions.

1.1 Literature review

In music analysis, we attempt to answer the question of how the music works. Specifically, in this study, how we can formalize the chord sequences so that the computer able to analyse the structure of music. To answer this question, we need to understand the music so that we can analyse them. In this section, we take an overview of some researches related to formalism of music [10], [25], [17], [14], [13], [16], [12], [1].

A Generative Theory of Tonal Music (GTTM)

GTTM is a theory developed by Fred Lerdahl and Ray Jackendoff in 1983. This theory try to describe the musical intuitions of an experienced listener in musical idiom. GTTM provides two theories. The first one is grouping analysis, which attempt to group music

pieces in boundaries. The second one is the metric analysis, which identifies the rhythmic structures of music. Time-span analysis and Prolongational analysis are composed by two theories above. For time-span analysis, we will get the representation of metric structure. For prolongational analysis, we will get the representation of harmonic stability [17].

The Cognition of Basic Musical Structures

This is an interesting book talking about music cognition. David Temperley provides us the way to extract musical information, such as meter, phrase structure, counterpoint, pitch spelling, harmony, and key from music as we hear it. From this model, we can build computations on computer by generating aspects of musical structure [13].

Tonal Pitch Space

In this book, Lerdahl extended GTTM with multidimensional model of diatonic and chromatic spaces that interest to music theorists, musicologists, composers, computer musicians, and cognitive psychologists. This model quantifies listeners' intuitions of the relative distances of pitches, chords, and keys from a given tonic. Also, the model represents paths through the space for analyzing harmony.[16].

The language of music

In the book was published in 1959 written by Deryck Cooke, he provides various examples for expressing the characteristics of music. The same tonal basis, melodic phrases, harmonies, and rhythms can express and evoke the same emotions. In addition, this book provides the expressive function of musical form of two complete symphonies by Mozart and Vaughan Williams [9].

Implementing "A Generative Theory of Tonal Music"

In this research, the authors proposed a system which can generate automatic time-span tree analyser (ATTA). This study demonstrates the ability of implementing a music analyzing system from formalism theory called Generative Theory of Tonal Music. Although the system cannot cover all 26 GTTM rules, we hope this tool can be extended for further research area [19].

1.2 Challenge

In those approaches mentioned above, there exists one approach from AI capable of applying for analyzing music. If we considered music as language, we can implement some methods and techniques into parsing musical grammar structure. This is because music and language have something in common. When we listen to a music piece, we able to realize the meaning behind the transmission of musicians. Similarly, we can understand

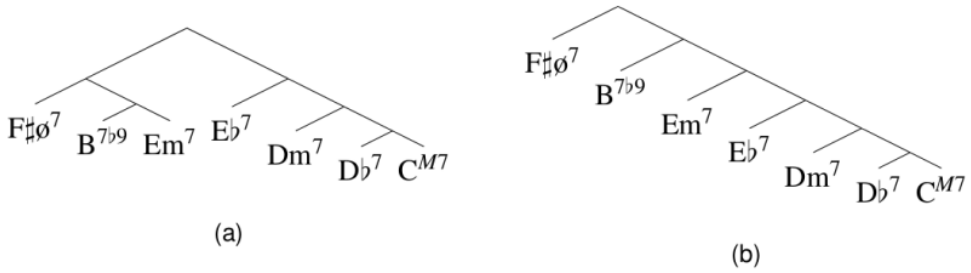


Figure 1.1: Two ways of interpreting a passage from Autumn Leaves: (a) as two separate cadences; (b) as a single cadence, in which the Em^7 has a dominant function [18].

each other by using language in communication. Besides, we both use utter in music and speaking to create sound into the air so that we can hear the transmit vibration.

Unfortunately, in analyzing music, we also get trouble of ambiguity when interpreting the structure of musical grammar. With the same chord progression, we can derive more than one musical structure. For instance, given a chord sequence from part of Autumn Leaves, we can analyse it in two ways as Figure 1.1. This is the challenge in improving the precision and accuracy when analyzing musical structure.

1.3 Goal

In a study of Mark Granroth-Wilding [27], he had proved that the methods of parsing grammar structure in Natural Language Processing (NLP) can efficiently analyse musical structure. With 92.29% in precision and 92.79% recall in PCCG and St+PCCG correspondingly, we can see the prospects in applying other improving methods of NLP into music.

However, in this approach, recall value in PCCG get only 88.78%, and precision value in St+PCCG only get 90.18%. The reason is that the final parse result of each chord progression is the highest parsing probability but not the correct candidate of n-best parse results.

In this paper, we aim to adapt Reranking model [5], [8], [2], [11] from NLP to improve the performance of CCG parsers [7], [20], [6], [37] for Jazz chord sequences [31], [23], [29], [4]. In addition, if we manually pick up the correct candidate for the final result of 5000-best, we can get 97.07% in f-score for PCCG parser, and 94.12% in f-score for St+PCCG parser.

1.4 Thesis organization

My thesis is organized in 5 chapters: Introduction, Background, Reranking model, Evaluation and Conclusion.

In Chapter 2, we introduce various background knowledge related to our research. At first, basic concepts about harmony are mentioned to help us understanding the relation-

ship between harmony and chord progression in analyzing music. Then, we cover some definitions of *Combinatorial Category Grammar* and show how we apply this grammar into interpreting chord progression.

In Chapter 3, we describe the Reranking model framework. Then, we will cover basic definitions and notations in the model. Next, we will define list of feature template that are used to extract features from the candidates of baseline parsers. Finally, we apply perceptron algorithm to find the optimal weighted vector for Reranking model.

In Chapter 4, we show the structure of Jazz corpus. Then, we talk about evaluation of model consist of calculating the precision, recall, and f-score. Finally, we analyze the current results of proposed method.

In Chapter 5, we summarize the current results in our study and mention some future works.

Chapter 2

Background

In this chapter, we introduce various background knowledge relating to our research. Basic concepts about harmony that used in music. The relationship between harmony and chord progression. Some visualization techniques to describe chord progression. How to apply Combinatory categorial grammar to interpret chord progression. In addition, table of lexicon is provided for helping analyze chord progression.

2.1 Harmony

Harmony, as its name implies, is a way to harmonize sounds together to form a song. Even sounds simple but this is a large part of the immensity and diversity of music. However, we will not go too deep in this section. Instead, we will cover some basic concepts of how harmony is.

Consonance

Consonance is the way that composers, musicians, the orchestra initiated on the tone that makes you feel consonant. This is because our brain can distinguish between the consonant and dissonant sounds. For example, when we try to hear one person playing wrong notes. No need to be a musician, you also easily recognize the feeling of "something is wrong" in the music. The interesting part is that this principle apply independently on every region of the world. Whether in European, African, or Asian, our brain "preferred" the way sound combines. Therefore, to analyze music, we seek to derive correctly musical structure of how a person conceive music piece.

Harmony

Harmony bases on principle mentioned above. The harmonic principle that we learn was found by the Gregorian monks centuries ago in Europe and still continue used these day. For instance, a symphony orchestra is composed of four groups of instruments, strings (violin, viola, violoncello, contrabass), instruments that made of wood was blown

when playing (flute, oboe, castanet, basso), instruments that made of brass (trumpet, trombone, tuba) and percussion (drums, gongs, percussion). A symphony requires an absolute consonance between all the instruments. That means at a certain moment, a hundred instruments were played in different musical notes, different tones from the bass to the treble notes that could combine together and make up a band.

2.2 Chord sequences

Chord is a set of notes that sound up together and form a harmonious timbre. This combination has rules and clear principles. Otherwise, we cannot generate consonant sound. The most important thing of a chord is quality of that chord. Each chord makes listeners different feelings and emotions. If we do not follow the rules, it will be very difficult to bring such feelings. The number of each group of chords are diverse. There can be three, four, five, six, etc., chords for each group but the principle of constructing them is only one. Fortunately, these group of chords are not combined randomly so we have very clear principles. Usually, a chord is a combination of three or four pitches.

Chord progression

In a song, to express the flow of melody, the musician have to use multiple chords and mix them together. Of course, a piece of music is not only one chord. Otherwise, it is similar to cook a soup with only salt. You need to combine many different spices, different types of ingredients to make a delicious dish with flavor.

In practice, we cannot mix the chords in an arbitrary way to create chord sequences. We need to follow groups of rules that we should obey. For example, the C major will come with A minor, F major and G7. These group of chords will form a block of harmony. In most music piece, we just play the chords in a block that can describe the songs or compositions.

2.3 Cadence

Cadence is a sequence of chords telling us the ending of the music piece. There are different types of cadence such as authentic cadence, half cadence, plagal cadence, interrupted cadence, inverted cadence, etc. As we can see in the Figure 2.1, half cadence is the progression from $I(tonic)$ to $V(dominant)$ that makes us feel a tension, and authentic cadence is the progression from $V(dominant)$ to $I(tonic)$ that makes us feel a relaxation.

Cadence is the key component of harmonic structure which built from expectation-resolution patterns. The larger structure which made up of successive expectation-resolution patterns called *extended cadences*. The example of this extended cadences will show in Figure 2.11.

There are different types of cadences but the common cadences are *authentic cadence* and *plagal cadence*. The authentic cadence consists of a chord rooted a perfect fifth above

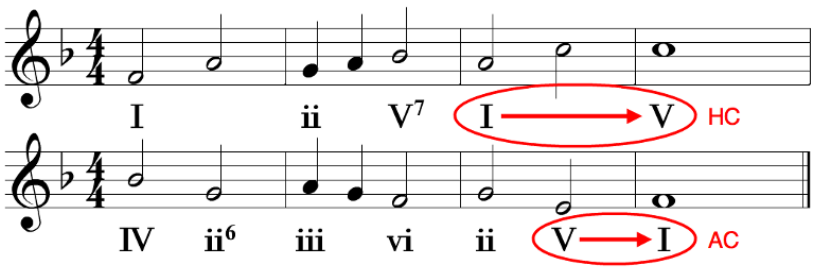


Figure 2.1: Half cadence (HC) and authentic cadence (AC).

its expected resolution. And plagal cadence consists of a tension chord rooted a perfect fourth above its resolution. The occurrence of particular chord shows its *function* on that occasion of use.

2.4 Harmonic Interpretation

In this section, we talk about some visualization techniques to describe chord progression. This is the way we define the musical syntax and semantics in tonal pitch space so that we can apply Combinatory categorial grammar in later section.

Just Intonation

This is the harmonic relation which is based on ratios of 2, 3, and 5 (the octave, perfect fifth, and major third). The interval between two notes are defined as the product $2^x \cdot 3^y \cdot 5^z$, where x, y, z are positive or negative integers. The harmonic relation can be visualized as an infinitely extending, discrete, three dimensional space with these three prime factors as generators [15].

In Figure 2.2, each point represent the tonal relation to the point labeled I which are degrees of the scale written in Roman numerals. The plus and minus sign are used for distinguishing the notational ambiguity of the syntonic comma. For example, the minor seventh interval ($\flat VII$) from the dominant seventh ($\flat VII^-$).

Longuet-Higgins Steedman (1971) [26] show that harmony can be calculated by a Manhattan distance metric over this space. In Figure 2.3, we see that major triad (CEG), minor triads ($CE\flat G$), and major seventh tetrad are close together in this metric space.

Equal Temperament

This is the tonal harmonic space which approximated by slightly equate all the positions that have the same names in Figure 2.4 and then by even further distorting the major thirds, to equate C with $B\sharp, D\flat\flat$. In the system of equal temperament, this is done by spacing the 12 tones of the diatonic octave evenly, so that all the semitones are (mis)tuned to the same ratio of $\sqrt[12]{2}$.

VI$^-$	III$^-$	VII$^-$	♯IV$^-$	♯I	♯V	♯II	♯VI	♯III$^+$	♯VII$^+$	♯♯IV$^+$
IV$^-$	I$^-$	V$^-$	II$^-$	VI	III	VII	♯IV	♯I$^+$	♯V$^+$	♯II$^+$
♭II$^-$	♭VI$^-$	♭III$^-$	♭VII$^-$	IV	I	V	II	VI$^+$	III$^+$	VII$^+$
♭♭VII$^-$	♭IV$^-$	♭I$^-$	♭V$^-$	♭II	♭VI	♭III	♭VII	IV$^+$	I$^+$	V$^+$
♭♭V$^-$	♭♭II$^-$	♭♭VI$^-$	♭♭III$^-$	♭♭VII	♭IV	♭I	♭V	♭II$^+$	♭VI$^+$	♭III$^+$

Figure 2.2: Part of the space of tonal relations [25].

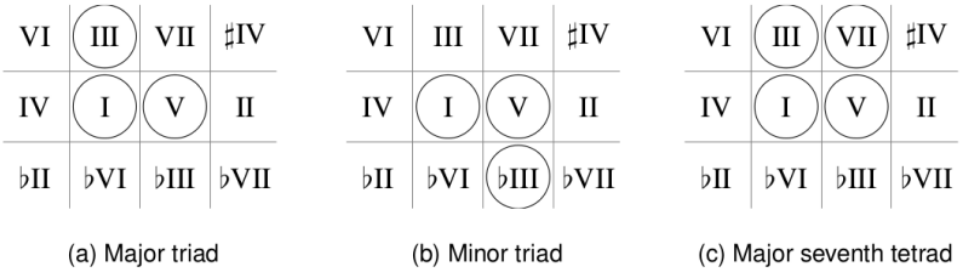


Figure 2.3: The tonal relations of the notes of the major (a) and minor (b) triads and the major seventh tetrad (c) [18].

E	B	F♯	C♯	G♯	D♯	A♯	E♯	B♯
C	G	D	A	E	B	F♯	C♯	G♯
A♭	E♭	B♭	F	C	G	D	A	E
F♭	C♭	G♭	D♭	A♭	E♭	B♭	F	C
D♭♭	A♭♭	E♭♭	B♭♭	F♭	C♭	G♭	D♭	A♭

Figure 2.4: Example of equal temperament space. The circled points form a C major triad. [18].

Interpretation of Tonics

By using a 4×3 enharmonic block, we can describe the logical form in coordination space as Figure 2.5. The notation $\langle x, y \rangle$ denotes an enharmonic coordinate which is different from vector (x, y) in full space. With these descriptions, the coordinate range is between $\langle 0, 0 \rangle$ and $\langle 3, 2 \rangle$.

Consider Figure 2.6, chord C^{M7} is represented as $[\langle 0, 0 \rangle]$ which is the root of the chord progression. Chord $A\flat^6$ is actually the chord $G\sharp$ so we can represent as $[\langle 0, 2 \rangle]$. When combine together, the chord sequence $C\ A\flat^{M7}$ can be represented as $[\langle 0, 0 \rangle, \langle 0, 2 \rangle]$ with $A\flat^6$ is the point underneath the $\langle 0, 0 \rangle$.

D♯	A♯	E♯	B♯	F♯♯	C♯♯	G♯♯	D♯♯	A♯♯	E♯♯
B	F♯	C♯	G♯	D♯	A♯	E♯	B♯	F♯♯	C♯♯
G	D	A	E	B	F♯	C♯	G♯	D♯	A♯
E♭	B♭	F	C	G	D	A	E	B	F♯
C♭	G♭	D♭	A♭	E♭	B♭	F	C	G	D
A♭♭	E♭♭	B♭♭	F♭	C♭	G♭	D♭	A♭	E♭	B♭
F♭♭	C♭♭	G♭♭	D♭♭	A♭♭	E♭♭	B♭♭	F♭	C♭	G♭

Figure 2.5: Enharmonic blocks at the centre of the space. A position within one of these 4×3 blocks is equated by equal temperament with the same position in every other block. [18].

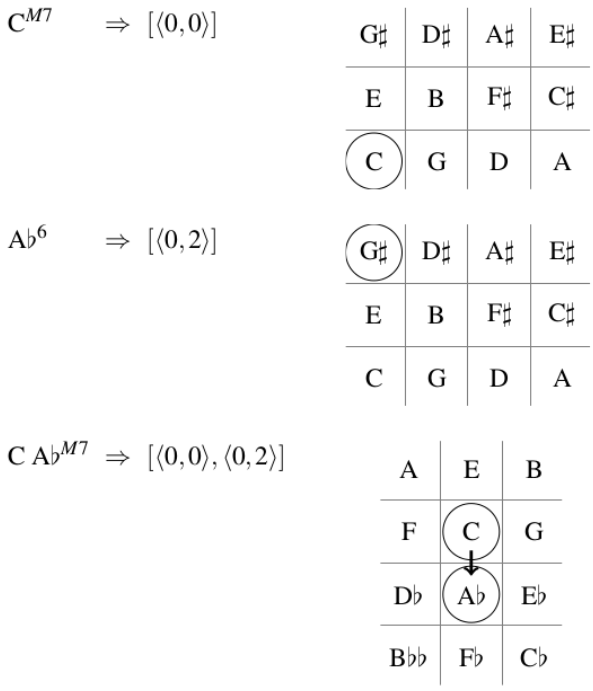


Figure 2.6: Some interpretations of chords as tonics. [18].

2.5 Combinatory Categorial Grammar (CCG)

Throughout the process of parsing musical structure, we use a grammar structure called Combinatory Categorial Grammar (CCG, Steedman, 2000) [30]. This grammar consists of two parts. First, the category used to associate words with specific categories which define syntactic behavior. Second, the combinator includes a set of rules defines how words and other constituents can be combined according to their categories.

2.5.1 Categories

The set of syntactic categories C is defined recursively as follows:

Atomic categories

Atomic categories are the grammar for each language is assumed to define a finite set of atomic categories. In language, we have Subject (S), Noun (N), Noun phrase (NP), ... $\in C$. In chord progression, we have Tonic (T), Dominant (Dom), Subdominant (Subdom), Diminished (Dim), ... $\in C$.

Complex categories

If $X, Y \in C$, then $X/Y, X\backslash Y \in C$. X is a result, Y is an argument, $X/Y, X\backslash Y$ are functors.

Given the following example, we want to see how to apply CCG from NLP into analyzing chord progression in music. In NLP, we have statement *flights to Boston* as example. In music, we have chord sequence $C\ Dm^7\ G^7\ C$.

Functors $X/Y, X\backslash Y$ have forward slash / represents directional variant of categorial grammar with arguments to the right of the functor, backslash \ represents directional variant of categorial grammar with arguments to the left of the functor.

For instance, a categorial lexicon for English of statement *flights to Boston* might contain the following entries:

$$\begin{aligned} flights &\vdash N \\ to &\vdash (N\backslash N)/NP \\ Boston &\vdash NP \end{aligned}$$

Meanwhile, the chord progression $C\ Dm^7\ G^7\ C$ might contain the following entries:

$$
\begin{aligned}
C &\vdash C^T \\
Dm^7 &\vdash D^D/G^{D|T} \\
G^7 &\vdash G^D/C^{D|T} \\
C &\vdash C^T
\end{aligned}
$$

As we can see, by using a finite set of these categories, CCG allows strong interaction between the lexical categories of a language and the combinatory rules allowed in its grammar.

Also, we can represent semantic interpretations in the language of the λ-calculus in each syntactic category. Each syntactic category has a type semantic interpretation correspond to that syntactic category. For complex functor category, we can use λ-term to represent the form of a semantic bound variable as examples below:

Predicate-argument structures for statement $flights\ to\ Boston$

$$
\begin{aligned}
flights &:= N : \lambda x.fight(x) \\
to &:= (N\backslash N)/NP : \lambda y.\lambda f.\lambda x.f(x) \wedge to(x,y) \\
Boston &:= NP : Boston
\end{aligned}
$$

Predicate-argument structures for chord progression $C\ Dm^7\ G^7\ C$

$$
\begin{aligned}
C &:= C^T : [\langle 0,0 \rangle] \\
Dm^7 &:= D^D/G^{D|T} : \lambda x.leftonto(x) \\
G^7 &:= G^D/C^{D|T} : \lambda x.leftonto(x) \\
C &:= C^T : [\langle 0,0 \rangle]
\end{aligned}
$$

2.5.2 Combinatory rules

There are four rules can be used to build derivations. Rules allow the combination between the syntactic categories and the logical forms simultaneously. We define a symbol for each rule to identify its use in derivations. The following rules below are written as productions. The left hand side of $\Longrightarrow$ are the inputs, the right hand side of $\Longrightarrow$ are the results of derivations.

The application combinator, often denoted by $>$ for forward application and $<$ for backward application. The composition combinator, often denoted by $>_{\mathbf{B}}$ for forward composition and $<_{\mathbf{B}}$ for backward composition.

Application rule

$$\begin{array}{lrrcl} Forward(>) & X/Y:f & Y-Z:x & \Longrightarrow & X-Z:f(x) \\ Backward(<) & X-Y:x & Z\backslash Y:f & \Longrightarrow & X-Z:f(x) \end{array}$$

The development rule joins together fully resolved passages. It requires its inputs to be atomic categories which need to be combined with something by function application before they can be considered resolved. This rule simply concatenate the two lists [18].

Development rule

$$(dev) \qquad V-W:l_0 \qquad X-Y:l_1 \implies V-Y:l_0 \oplus l_1$$

Composition rules permit complex categories to be combined before their argument is available. The final outcome is the same as function application. The different is this rule allows the outcome to be produced by a different order of combinations.

Composition rule

$$\begin{array}{lrrcl} Forward(>_{\mathbf{B}}) & X/Y:f & Y/Z:g & \Longrightarrow & X/Z:f\circ g \\ Backward(<_{\mathbf{B}}) & X\backslash Y:g & Z\backslash X:f & \Longrightarrow & Z\backslash Y:f\circ g \end{array}$$

The coordination rule combines two unresolved cadences to behave as a single unresolved cadence and requires that they are expecting the same resolution. The two cadences are required to be either both authentic (dominant) or both plagal (subdominant) [18].

Coordination rule

$$(\&) \qquad X^F/Y:f \qquad Z^F/Y:g \implies X^F/Y:f\wedge g \qquad where\ F\in D,S$$

2.5.3 Examples for applying combinatory rules

The following examples describe how combinatory rules derive based on given lexicons.

Application rule

First example 2.7 uses forward $>$ and backward $<$ application rule to derive simple statement $flights\ to\ boston$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\text{VI}^7}{\lambda x.L(x)} \quad \dfrac{\text{IIm}^7\ \text{V}^7}{\lambda x.L(L(x))}}{\lambda x.L(L(L(x)))}>\mathbf{B} \quad \dfrac{\text{IIm}^7\ \text{V}^7}{\lambda x.L(L(x))}}{\lambda x.L(L(L(x))) \wedge \lambda x.L(L(x))}\& \quad \dfrac{\text{I}}{[\langle 0,0\rangle]}}{[(\lambda x.L(L(L(x))) \wedge \lambda x.L(L(x)))(\langle 0,0\rangle)]}>$$

Figure 2.9: CCG derivation for chord sequence using composition and cooridnation rules. L stands for $leftonto$ predicate [27].

$$\dfrac{\dfrac{flights}{N : \lambda x.flight(x)} \quad \dfrac{\dfrac{to}{(N\backslash N)/NP : \lambda y.\lambda f.\lambda x.f(x) \wedge to(x,y)} \quad \dfrac{boston}{NP : boston}>}{(N\backslash N) : \lambda f.\lambda x.f(x) \wedge to(x, boston)}}{N : \lambda x.flight(x) \wedge to(x, boston)}<$$

Figure 2.7: CCG derivation for *flights to boston*

Application rule and development rule

Second example 2.8 uses forward > application rule and development rule *dev* to derive simple chord sequence $C\ Dm^7\ G^7\ C$.

$$\dfrac{\dfrac{\text{C}}{C^T : [\langle 0,0\rangle]} \quad \dfrac{\dfrac{Dm^7}{D^D/G^{D|T} : \lambda x.leftonto(x)} \quad \dfrac{\dfrac{G^7}{G^D/C^{D|T} : \lambda x.leftonto(x)} \quad \dfrac{\text{C}}{C^T : [\langle 0,0\rangle]}>}{G^D - C^T : [leftonto(\langle 0,0\rangle)]}}{D^D - C^T : [leftonto(leftonto(\langle 0,0\rangle))]}>}{C^T : [\langle 0,0\rangle, leftonto(leftonto(\langle 0,0\rangle))]}\mathbf{dev}$$

Figure 2.8: CCG derivation for chord sequence

Composition rule and coordination rule

The example 2.9 uses forward $>_{\mathbf{B}}$ composition rule and coordination rule & to derive simple cadence $VI^7\ IIm^7V^7\ IIm^7V^7\ I$.

Applying four rules

An example derivation using all four rules is shown in Figure 2.10

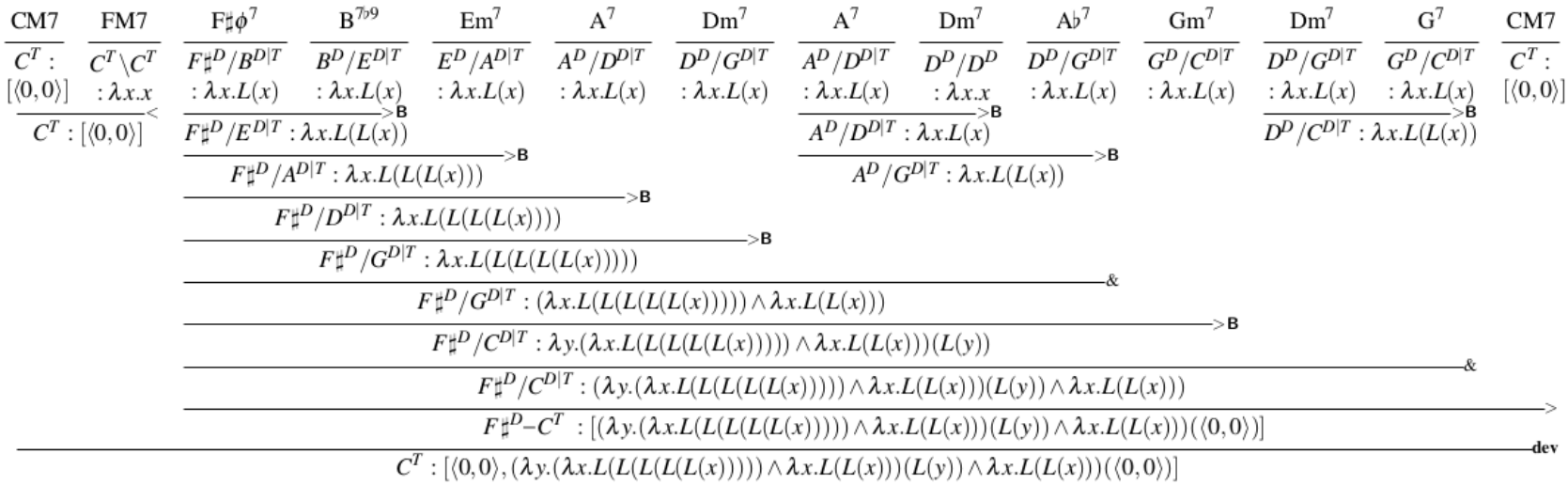


Figure 2.10: Semantic derivation of the extended cadence from *Alice in Wonderland* [27].

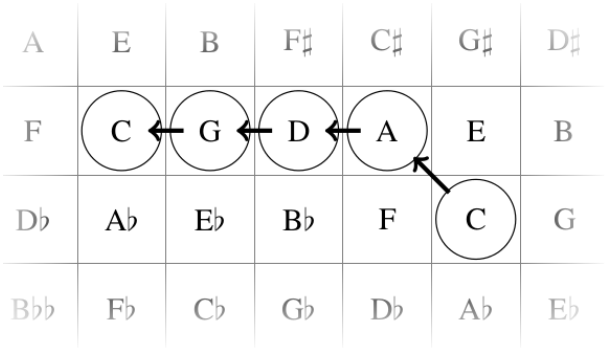


Figure 2.11: A tonal space path for the extended cadence: $CA^7D^7G^7C$

2.6 Domain for analysis

In this research, we use tonal space as the semantic domain of harmonic analysis. The harmonic interpretation of a piece is the path through the tonal space traced by the roots of the chords.
The relationship of expectation-resolution between two chords can be described as a single step to the left in the space. Meanwhile, the relationship of subdominant-tonic is a rightward step.
Figure 2.11 is an example of tonal space path for an extended cadence. The perfect fifth relationship between the dominants and their resolutions is reflected in the path.
By identifying the syntactic structure of the harmony, that is the recursive structure of expectation-resolution relationships between pairs of chords, we produce the path through the space that this dictates for the chord roots of the progression, including those that have been substituted for.

2.6.1 Extracting tonal space path

The logical form of categories are the semantics represent the path through the tonal space. The tonic points can be ambiguous in the representation but every point of a path can be inferred from a full logical form.
There are two types of predicates that used in this study. The left movement in space denoted in the semantics by $leftonto$ predicate and right movement in space denoted in the semantics by $rightonto$. If the point (x, y) is specified, the path $leftonto(leftonto((x, y)))$ is unambiguous.
From derivation of CCG analysis structure, we can visualize this result in tonal space as example 2.12. Given a chord sequence $Dm^7\ G^7\ C$, we can analyze this sequence by using CCG grammar. Then, we we can use this result to visualize as tonal space path.
A simple algorithm is constructed for recursive transformation of the logical predicates to produce the flat tonal space path represented by a logical form produced by parsing.

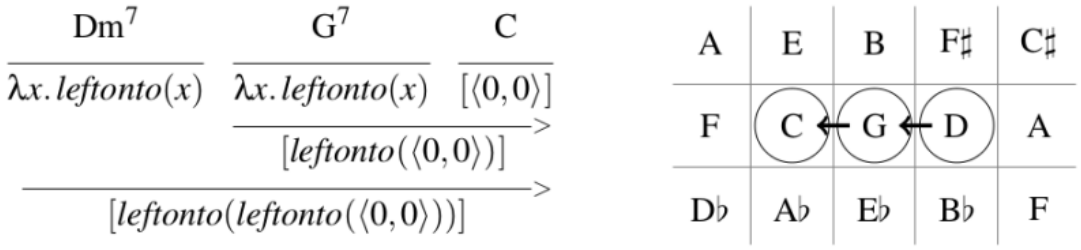


Figure 2.12: Interpretation of an extended cadence with two dominants the familiar $IIm^7\ V^7\ I$ derived from the logical forms of the individual chords. The resulting tonal space path movements are shown to the right. [18].

2.6.2 Harmony as dependency graph

In NLP, we often use dependency graph to represent the syntactic relations between words in a sentence. Similarly, when analyzing the harmonic relations between chords, we can use dependency graph to express the musical logical forms of sequences. Dependencies can be drawn from the tonal space path where exists logical form relations between them. Each tension chord is marked as dependent on its resolution. The arcs are used in this thesis is labeled with the name of the predicate ($leftonto$ or $rightonto$).
The dependency graph consists of root which is denote as $\langle 0,0\rangle$ symbol. In addition, each dependency is a pair (h,m) where h is the index of a head, m is the index of a modifier. From example 2.8 we can illustrate dependency graph for harmonic interpretation of CCG derivation

$$C^T : [\langle 0,0\rangle, leftonto(leftonto(\langle 0,0\rangle))]$$

as Figure 2.13. In this figure, we represent a dependency (h,m) by a directed edge from h to m.

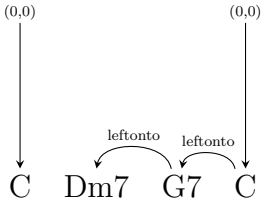


Figure 2.13: Harmonic interpretation of $C\ Dm^7\ G^7\ C$ represented as a dependency graph.

These graphs express the the logical forms (the *semantics*) of the harmonic interpretation. By using the dependency graph we can visualize the functional expectations and resolutions of the harmonic relations. Then evaluation metrics which is edit distance can be calculated.

Table 2.1: Definitions of the chord types included in each chord class, as used in the lexicon. This component of the grammar is flexible and may be altered to suit notational conventions and genre-specific chord function conventions. The definitions in this table are those that I have used to parse Jazz chord sequences [18].

Class	Chord types
X	$X, X^{M7}, X^{7}, X^{aug,M7}, X^{\flat 5,M7}, X^{sus4}, X^{sus4,7}$
Xm	Xm, Xm^{7}, Xm^{M7}
X^{7}	$X^{7}, X^{aug}, X^{aug,7}, X, X^{\flat 5,7}, X^{\flat 5}, X^{sus4}, X^{sus4,7}$
Xm^{7}	$Xm^{7}, X\phi^{7}, X\circ^{7}, Xm, Xm^{\flat 5}, Xm^{M7}$
$X\circ$	$X\circ^{7}, X\phi^{7}$

2.7 The lexicon

Table 2.2 summarize all lexicon of grammar of jazz chord sequences. First column is mnemonic label which indicates the identifier of lexical schema. Second column is category schema which generalize as X chord root and followed by syntactic type using Roman numeral notation. Table 2.1 shows generalized syntactic type and a logical form of harmonic interpretations.

Table 2.2: The lexicon for the jazz grammar. For each schema, a typical example is given of a chord in the key of C and the syntactic type of the category that would be assigned to that chord [18].

Mnemonic label	Category schema	Chord	Syntactic type
Ton.	$X(m) := I^T : [\langle 0,0\rangle]$	C^{M7}	C^T
Ton-III.	$Xm := \flat VI^T : [\langle 0,2\rangle]$	Em	C^T
Ton-bVI.	$X := III^T : [\langle 0,1\rangle]$	$A\flat^{M7}$	C^T
Dom.	$X(m)^7 := I^D/IV^{D\|T} : \lambda x.leftonto(x)$	G^7	$G^D/C^{D\|T}$
Dom-backdoor.	$X(m)^7 := VI^D/II^{D\|T} : \lambda x.leftonto(x)$	$B\flat^7$	$G^D/C^{D\|T}$
Dom-tritone.	$X(m)^7 := \flat V^D/VII^{D\|T} : \lambda x.leftonto(x)$	$D\flat^7$	$G^D/C^{D\|T}$
Dom-bartok.	$X(m)^7 := \flat III^D/\flat VI^{D\|T} : \lambda x.leftonto(x)$	E^7	$G^D/C^{D\|T}$
Subdom.	$X(m) := I^S/V^{S\|T} : \lambda x.rightonto(x)$	F	$F^S/C^{S\|T}$
Subdom-bIII.	$X := VI^S/III^{S\|T} : \lambda x.rightonto(x)$	$A\flat$	$F^S/C^{S\|T}$
Dim-bVII.	$X\circ := IV^D/\flat VI^{D\|T} : \lambda x.leftonto(x)$	$D\circ^7$	$G^D/C^{D\|T}$
Dim-V.	$X\circ := II^D/V^{D\|T} : \lambda x.leftonto(x)$	$F\circ^7$	$G^D/C^{D\|T}$
Dim-III.	$X\circ := VII^D/III^{D\|T} : \lambda x.leftonto(x)$	$A\flat\circ^7$	$G^D/C^{D\|T}$
Dim-bII.	$X\circ := \flat VI^D/\flat II^{D\|T} : \lambda x.leftonto(x)$	$B\circ^7$	$G^D/C^{D\|T}$
Pass-I.	$X\circ := I^T/I^T : \lambda x.x$	$C\circ^7$	C^T/C^T
	$X\circ := I^D/I^D : \lambda x.x$	$G\circ^7$	G^D/G^D
Pass-VI.	$X\circ := VI^T/VI^T : \lambda x.x$	$A\circ^7$	C^T/C^T
	$X\circ := VI^D/VI^D : \lambda x.x$	$E\circ^7$	G^D/G^D
Pass-bV.	$X\circ := \flat V^T/\flat V^T : \lambda x.x$	$G\flat\circ^7$	C^T/C^T
	$X\circ := \flat V^D/\flat V^D : \lambda x.x$	$D\flat\circ^7$	G^D/G^D
Pass-bIII.	$X\circ := \flat III^T/\flat III^T : \lambda x.x$	$E\flat\circ^7$	C^T/C^T
	$X\circ := \flat III^D/\flat III^D : \lambda x.x$	$B\flat\circ^7$	G^D/G^D
Aug-bII.	$X^7 := \flat VI^D/\flat II^{D\|T} : \lambda x.leftonto(x)$	$Baug$	$G^D/C^{D\|T}$
Aug-VI.	$X^7 := \flat III^D/\flat VI^{D\|T} : \lambda x.leftonto(x)$	$E\flat aug$	$G^D/C^{D\|T}$
Colour-IVf.	$X(m) := V^T/V^T : \lambda x.x$	F	C^T/C^T
Colour-IVb.	$X(m) := V^T\backslash V^T : \lambda x.x$	F	$C^T\backslash C^T$
Colour-IIf.	$X(m) := \flat VII^T/\flat VII^T : \lambda x.x$	Dm	C^T/C^T
Colour-IIb.	$X(m) := \flat VII^T\backslash\flat VII^T : \lambda x.x$	Dm	$C^T\backslash C^T$
Dom-IVm.	$Xm := II^D/V^T : \lambda x.leftonto(x)$	$Fm(6)$	G^D/C^T

Chapter 3

Reranking model

In this section we talk about reranking model, which are used in natural language processing to improve parsing performance. A key advantage of Reranking models is their flexibility. As we will see, they allow a very rich set of features to be used in a model, arguably much richer representations than the simple estimation techniques. In this section we will give basic definitions, and describe how parameters can be estimated in this model.

3.1 Basic Definitions

Definition 1 (Reranking Model) A Reranking model consists of the following components:

- A set X of possible input chord sequences.
- A set Y of possible dependency graphs. The set Y is assumed to be finite.
- A positive integer d specifying the number of features and parameters in the model.
- A function $f : X \times Y \rightarrow \mathbb{R}^d$ that maps any (x, y) pair to a feature-vector $f(x, y)$.
- GEN is a function that maps an input x to a set of candidates $GEN(x)$
- A parameter vector $w \in \mathbb{R}^d$.

3.2 Reranking framework

Consider Reranking framework in figure 3.1. For any given input chord sequence x we use a baseline parser to produce top N parses for each chord sequence in training and test data. That means we apply function GEN(x) to generate the top N parses for x under the baseline model (PCCG and Supertagging+PCCG parsers). Then for each candidates dependency parse y, we apply function feature $f(x, y)$ to extract the feature

for each candidate. Finally, we choose the highest scoring candidate as the most plausible structure which satisfied $F(x) = \text{argmax}_{y \in GEN(x)}\, w \cdot f(x, y)$.
In more details, Reranking model only considers a small number of parses per sequence so it is not necessary to use dynamic programming. For each sequence x the n-best dependency graph from baseline parsers returns n highest probability dependency graphs $Y(x) = y_1(x), ..., y_n(x)$.
A feature extractor is a vector of m functions $f = (f_1, ..., f_m)$, where each f_j maps any (x, y) pair to a real number $f_j(x, y)$, which is the value of the jth feature on y. So a feature extractor maps each pair (x, y) to a vector of feature values $f(x, y) = (f_1(x, y), ..., f_m(x, y))$.
Our Reranking model associates a parse with a score $v_w(y)$ which is a linear function of the feature values $f(x, y)$. That is, each feature f_j is associated with a weight w_j, and the feature values and weights define the score $v_w(y)$ of each dependency graph y as follows:

$$v_w(y) = w \cdot f(x, y) = \sum_{j=1}^{m} w_j f_j(x, y).$$

Given a sequence x, the Reranking model output $F(x)$ on sequence x is the highest scoring dependency graph in the n-best dependency graphs $Y(x)$ for x, i.e.,

$$F(x) = \arg \max_{y \in GEN(x)} w \cdot f(x, y).$$

The feature weight vector w is estimated from the annotated jazz corpus by using perceptron algorithm. Because we used annotated training data we know the correct dependency graph $y_*(x)$ for each sequence x in the training data. The correct dependency graph $y_*(x)$ is not always a member of the n-best parser's output $Y(x)$. In addition, there are some cases the n-best parser could not find any parses which reduce the coverage score in evaluation phase.

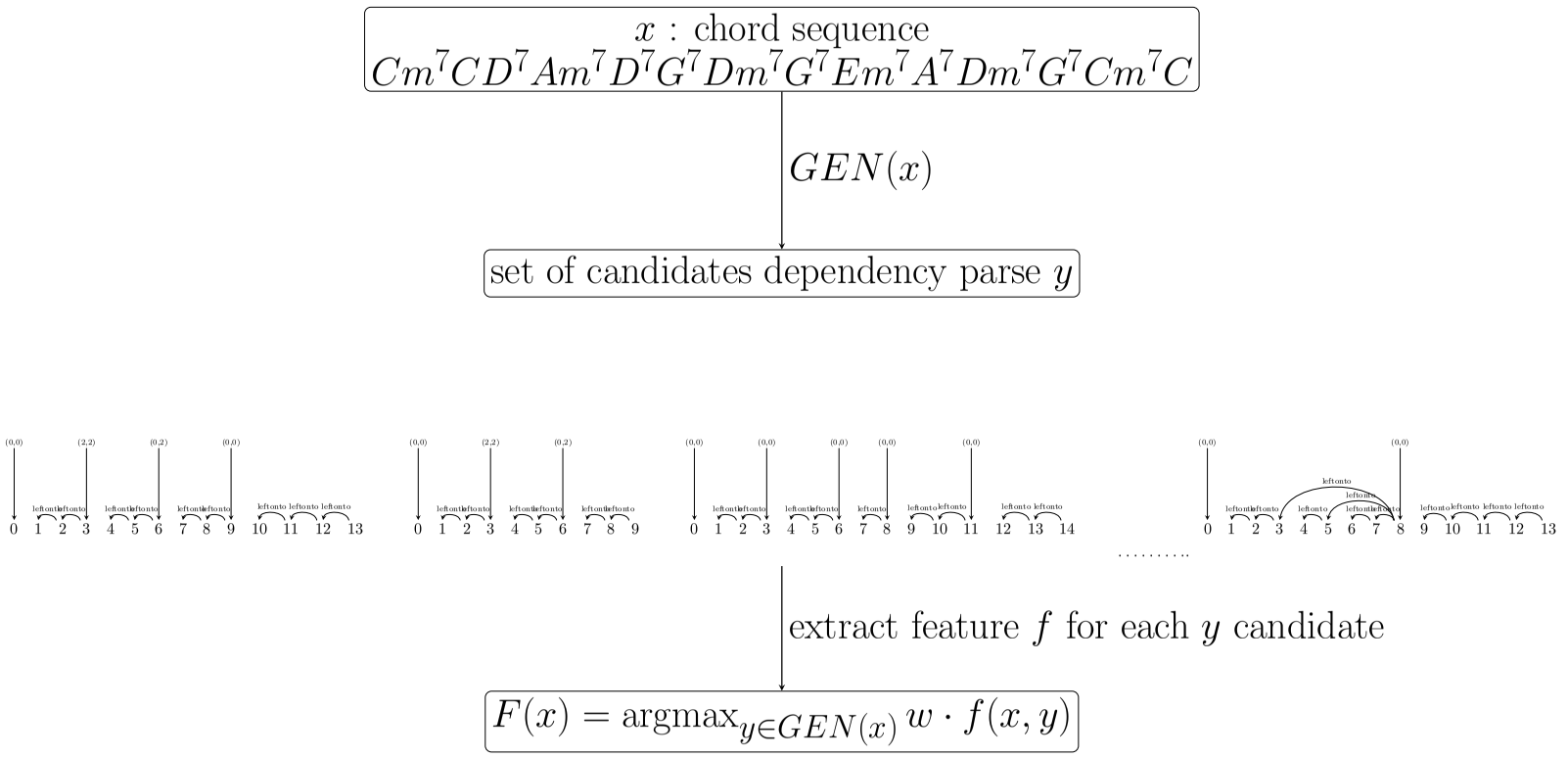


Figure 3.1: Reranking framework.

3.3 Features

As described in the previous section, for any pair (x, y), $f(x, y) \in \mathbb{R}^d$ is a feature vector representing that pair. Each component $f_k(x, y)$ for $k = 1...d$ in this vector is referred to as a feature. The features allows us to represent different properties of the input x, in conjunction with the dependency graph y. Each feature has an associated parameter, w_k , whose value is estimated using a set of training examples. The training set consists of a sequence of examples $(x^{(i)}, y^{(i)})$ for $i = 1...n$, where each $x^{(i)} \in X$, and each $y^{(i)} \in Y$.
Features belong to feature template, which are abstract template from which specific features are instantiated. The rest of this section outlines the feature template used in the experiment. These feature template used here were developed using the n-best parses provided by Mark Granroth-Wilding.

3.3.1 Features template

In this thesis, we defined feature templates for n-gram feature. These features include unigram model, bigram statistical model, and trigram statistical model. We consider all possible unigrams, bigrams and trigrams seen in the training data. To do this, we use feature templates to generate sets of features.
Given chord sequence 13 from corpus C^7 C^7 F^7 F C^7 Gm^7 Abm^7 Gm^7 C^7. We define c_i to be the ith coordination in the chord sequence, s_i to be the name of a lexical category schema for the ith coordination, p_i to be the ith position of chord in the sequence. Coordination dependency graph for this chord sequence illustrated in figure 3.2 and position dependency graph of each chord is illustrated in figure 3.3.

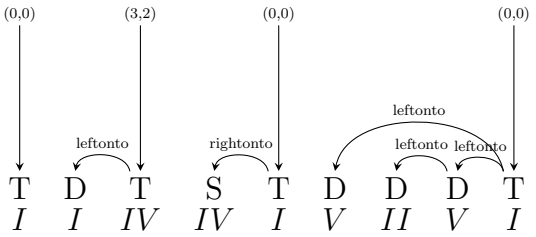


Figure 3.2: Coordination dependency graph.

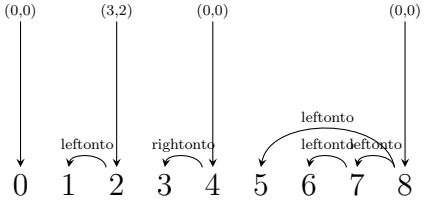


Figure 3.3: Coordination dependency graph is converted to position number.

Using Jazz chord sequence corpus, there are 1760 features that meet the feature template. Based on these definitions, we define our feature templates for n-gram feature followed

by the number of features in that template and a brief description of the instances of the template as follow:

- **Unigram features (381):** Identity of c_h . Identity of c_m . Identity of s_h . Identity of s_m. Identity of p_h. Identity of p_m.
- **Bigram features (1103):** Identity of the 4-tuple $\langle c_h, c_m, s_h, s_m \rangle$. Identity of subsets of this 4-tuple, e.g, identity of $\langle c_h, c_m \rangle$. Identity of $\langle p_h, p_m \rangle$.
- **Trigram features (276):** Identity of the 6-tuple $\langle c_h, c, c_m, s_h, s, s_m \rangle$. Identity of $\langle p_h, p, p_m \rangle$.

Given sequence 13 from corpus with dependency graphs above, we apply feature template for extracting features from annotated gold standard. The total features achieve 174 features consist of 64 features from unigram, 94 features from bigram, and 16 features from trigram. The example for feature extraction shown in table 3.1, 3.3, and 3.2.

Table 3.1: Unigram feature extraction for sequence 13.

Unigram features example (total 64 features)
UNIGRAM:1; UNIGRAM:2; UNIGRAM:3; UNIGRAM:4; UNIGRAM:5 UNIGRAM:6; UNIGRAM:7; UNIGRAM:8; UNIGRAM:9; UNIGRAM:10 UNIGRAM:11; UNIGRAM:12; UNIGRAM:13; UNIGRAM:14; UNIGRAM:15 UNIGRAM:16; UNIGRAM:17; UNIGRAM:18; UNIGRAM:19; UNIGRAM:20 UNIGRAM:21; UNIGRAM:22; UNIGRAM:23; UNIGRAM:24; UNIGRAM:25 UNIGRAM:26; UNIGRAM:27; UNIGRAM:28; UNIGRAM:29; UNIGRAM:30 UNIGRAM:31; UNIGRAM:32; UNIGRAM:33; UNIGRAM:34; UNIGRAM:35 UNIGRAM:36; UNIGRAM:37; UNIGRAM:38; UNIGRAM:39; UNIGRAM:40 UNIGRAM:II; UNIGRAM:VI+; UNIGRAM:V; UNIGRAM:I; UNIGRAM:VII UNIGRAM:III; UNIGRAM:VI; UNIGRAM:II-; UNIGRAM:V-; UNIGRAM:I- UNIGRAM:bVII–; UNIGRAM:IV-; UNIGRAM:bIII–; UNIGRAM:bVI– UNIGRAM:bII–; UNIGRAM:bV—; UNIGRAM:bI—; UNIGRAM:bIV— UNIGRAM:bbVII—; UNIGRAM:bbIII—-; UNIGRAM:bbVI—- UNIGRAM:bbII—-; UNIGRAM:D; UNIGRAM:T

Table 3.2: Trigram feature extraction for sequence 13.

Trigram features example (total 16 features)
TRIGRAM:7:6:5; TRIGRAM:17:16:15; TRIGRAM:26:25:24 TRIGRAM:29:28:27; TRIGRAM:40:39:38; TRIGRAM:I:V:II TRIGRAM:I-:V-:II-; TRIGRAM:bVI–:bIII–:bVII– TRIGRAM:bV—:bII–:bVI–; TRIGRAM:bbII—-:bbVI—-:bbIII—- TRIGRAM:T:D:D; TRIGRAM:I:T:V:D:II:D; TRIGRAM:I-:T:V-:D:II-:D TRIGRAM:bVI–:T:bIII–:D:bVII–:D; TRIGRAM:bV—:T:bII–:D:bVI–:D TRIGRAM:bbII—-:T:bbVI—-:D:bbIII—-:D

Table 3.3: Bigram feature extraction for sequence 13.

Bigram features example (total 94 features)
BIGRAM:1:6; BIGRAM:2:3; BIGRAM:3:6; BIGRAM:4:5
BIGRAM:5:6; BIGRAM:6:7; BIGRAM:7:ROOT; BIGRAM:8:9
BIGRAM:9:10; BIGRAM:10:11; BIGRAM:11:16
BIGRAM:12:13; BIGRAM:13:16; BIGRAM:14:15
BIGRAM:15:16; BIGRAM:16:17; BIGRAM:17:ROOT
BIGRAM:18:25; BIGRAM:19:20; BIGRAM:20:25
BIGRAM:21:22; BIGRAM:22:23; BIGRAM:23:24
BIGRAM:24:25; BIGRAM:25:26; BIGRAM:26:ROOT
BIGRAM:27:28; BIGRAM:28:29; BIGRAM:29:ROOT
BIGRAM:30:31; BIGRAM:31:32; BIGRAM:32:33
BIGRAM:33:34; BIGRAM:34:39; BIGRAM:35:36
BIGRAM:36:39; BIGRAM:37:38; BIGRAM:38:39
BIGRAM:39:40; BIGRAM:40:ROOT; BIGRAM:II:V
BIGRAM:VI+:II; BIGRAM:V:I; BIGRAM:I:ROOT
BIGRAM:VII:III; BIGRAM:III:VI; BIGRAM:VI:II-
BIGRAM:II-:V-; BIGRAM:V-:I-; BIGRAM:I-:ROOT
BIGRAM:bVII–:bIII–; BIGRAM:IV-:bVII–
BIGRAM:I-:IV-; BIGRAM:bIII–:bVI–
BIGRAM:bVI–:ROOT; BIGRAM:bVI–:bII–
BIGRAM:bII–:bV—; BIGRAM:bV—:ROOT
BIGRAM:bV—:bI—; BIGRAM:bI—:bIV—
BIGRAM:bIV—:bbVII—; BIGRAM:bbVII—:bbIII—-
BIGRAM:bbIII—-:bbVI—-; BIGRAM:bbVI—-:bbII—-
BIGRAM:bbII—-:ROOT; BIGRAM:D:D; BIGRAM:D:T
BIGRAM:T:ROOT; BIGRAM:II:D:V:D; BIGRAM:VI+:D:II:D
BIGRAM:V:D:I:T; BIGRAM:I:T:ROOT; BIGRAM:VII:D:III:D
BIGRAM:III:D:VI:D; BIGRAM:VI:D:II-:D
BIGRAM:II-:D:V-:D; BIGRAM:V-:D:I-:T
BIGRAM:I-:T:ROOT; BIGRAM:bVII–:D:bIII–:D
BIGRAM:IV-:D:bVII–:D; BIGRAM:V-:D:I-:D
BIGRAM:I-:D:IV-:D; BIGRAM:bIII–:D:bVI–:T
BIGRAM:bVI–:T:ROOT; BIGRAM:bVI–:D:bII–:D
BIGRAM:bII–:D:bV—:T; BIGRAM:bV—:T:ROOT

Table 3.4: Perceptron algorithm for finding optimal parameters of Reranking model.

Perceptron algorithm	
Inputs:	Training set (x_i, y_i) for $i = 1...n$
Initialization:	$w = 0$
Define:	$F(x) = \text{argmax}_{y \in GEN(x)}\, w \cdot f(x, y)$
Algorithm:	For $t = 1...T, i = 1...n$ $z_i = F(x_i)$ if $(z_i \neq y_i)$ $w = w + f(x_i, y_i) - f(x_i, z_i)$
Output:	Parameters w

3.4 Finding the Optimal Parameters

In this section, we talk about method for parameter estimation. This is the method for finding w parameters which is called perceptron algorithm. We will see that it is a very simple algorithm, but empirically it turns out to work really quite well on Reranking model.

3.4.1 Perceptron algorithm

Given the training data $D = (x_1, ..., x_{n'})$ is a sequence of chord progression. We used 10-fold cross validation technique to compute the n-best dependency graphs $Y(x)$ for each chord progression x in D.
Consider perceptron algorithm below, we have some training set of examples xi, yi for $i = 1...n'$, and function GEN and function f which are described early section. Function GEN enumerates candidates for an input, and function f defines a feature vector mapping.

The task is going to to learn our parameters w by using training examples as information. Firstly, we initialize parameters w equals the vector of zeros. Then, we define function F which takes input x and pick the highest scoring member of GEN as the final output of this model.
After T number of iterations, the algorithm will finish and return the final parameters w. In this implementation, we choose T equals 10. For each training example the perceptron algorithm does two things. Firstly, it calculates the current output z_i from the model which is the highest scoring structure under current parameters. Then, if z_i is equal to y_i, that means we have correctly recovered the correct structure on this particular example. And in that case, I do nothing to the parameters. Otherwise, if z_i is not equal to y_i, then the structure we have produced is somehow different from the target structure y_i. In that case, we make an update to the parameters w by adding the feature vector for x_i, y_i and subtracting the feature vector for x_i, z_i.
Intuitively, any features which are seen in the true structure have their parameter value

increased, whereas any features seen in the incorrectly proposed structure have their parameter values decreased. So this is kind of a reinforcing step which pushes up the parameter values for things seen in the truth and pushes down the parameter values for features seen in z_i which was the incorrectly produced structure.

Chapter 4

Evaluation

In this chapter, we will evaluate our Reranking model for analyzing music. First, we will cover some information about Jazz corpus which is the dataset for evaluating the performance. Next, we will show evaluation indexes include precision, recall, and f-score for Jazz chord sequence analysis. Finally, the current results of Reranking model will be shown and analyzed.

4.1 Corpus

In this thesis, we use the Jazz corpus for training and testing our Reranking model. There are 76 annotated sequences and about 3000 chords which are indicated in the Table 4.1 and 4.2.
Table 4.2 shows data structure of each chord. *Root* is an integer pitch class of transcribed chord root $(0, ..., 11)$, relative to the main key. *Type* is the main chord type (tetrad) symbol, chosen from a small vocabulary (e.g. '', 'm', 'M7'). *Addition* is any further transcribed tones added to the chord, but not included in the type symbol (e.g. '♭13'). *Bass* is the pitch class of a bass note, if one is given in the chord symbol (e.g. the B in 'C^{M7}/B'). *Duration* is duration of the chord in beats. Jazz corpus can be downloaded by links below.
Links download

- Download the corpus in a human-readable format: `http://jazzparser.granroth-wilding.co.uk/JazzCorpus?action=AttachFile&do=get&target=chord_corpus.txt`
- Download the corpus in a CSV format: `http://jazzparser.granroth-wilding.co.uk/JazzCorpus?action=AttachFile&do=get&target=corpus.tar.gz`

4.1.1 Annotating a Unique Gold-Standard Interpretation

Each chord in the corpus is annotated by a category from the lexicon identified by mnemonic label defined in Table 2.2. With a small set of combinatory rules and strongly

Table 4.1: First 10 rows in song corpus (76 sequences).

ID	Key	Bar length	First chord
2	F minor	4	0
4	C major	4	70
7	A minor	4	114
8	Eb major	4	165
9	F minor	4	191
10	C major	4	228
12	C major	3	295
13	G major	2	361
14	C major	4	370
16	Ab major	4	416

Table 4.2: First 10 rows in chord corpus (3085 rows).

Root	Type	Additions	Bass	Duration	Lexical schema	Coord middle	Coord end
0	m			4	T	False	False
5	m7			4	`TC_IVb`	False	False
7	7	b9		4	T	False	False
0	m			4	T	False	False
0	7	b9		4	T	False	False
5	m7			4	T	False	False
10	7	b9		4	T	False	False
3	M7			4	T	False	False
8	M7			4	`TC_IVb`	False	False
2	%7			4	`D_Rep`	False	False

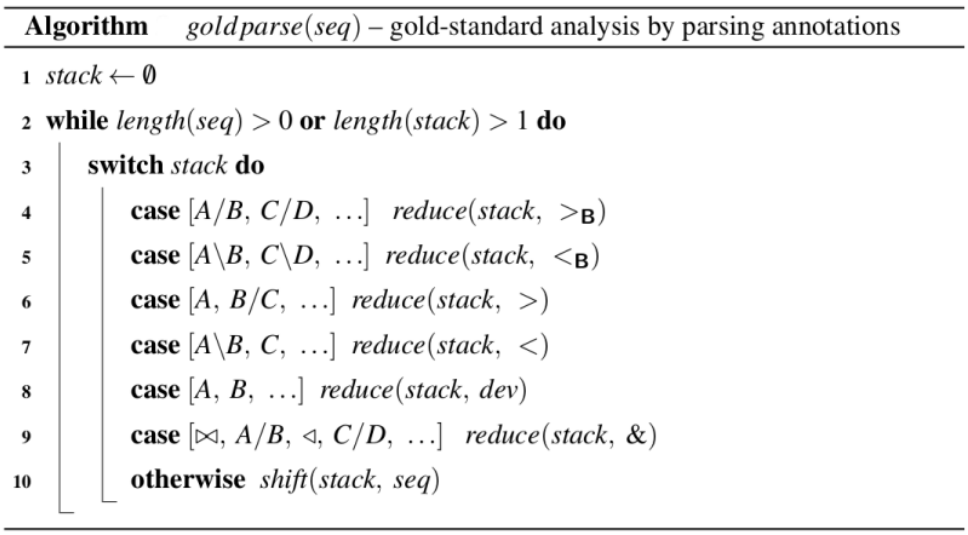


Figure 4.1: Algorithm for annotating a unique gold standard [18].

lexicalized grammar formalism of CCG, we have enough information to construct a unique logical form for each sequence.
Algorithm for annotating a unique gold standard is showed in Figure 4.1. This algorithm specifies a procedure which is sufficient to annotate a choice of lexical category for each chord and the locations of coordination in order to achieve a full annotation of a gold-standard tonal space path for each sequence.

4.2 Evaluation

In this thesis, we use Tonal space edit distance (TSED) metric to compare the output of a parse result to the annotated gold-standard harmonic structure. This metric is the closeness of the interpretation as a path through the tonal space.

4.2.1 Tonal space edit distance

The tonal space distance metric is computed as the edit distance between two path. That is the smallest number of insertions, deletions or substitutions of points on one path required to transform it into the other. Edit distance (also known as Levenshtein distance) can be computed using a dynamic programming algorithm described by Wagner & Fischer (1974) with a scoring function gives a penalty of 1 for deletion, insertion or substitution.

4.2.2 Accuracy evaluation

Once the alignment algorithm has found the optimal alignment between the two paths, we can compute the accuracy of the alignment using this scoring system.

The precision P (proportion of points in the output matched in the gold standard) is defined as the number of matching category choices Cat_m plus the number of matching points of coordination $Coord_m$, divided by the total number of chords N and annotated points of coordination in the re-annotation $Coord_r$.

$$P = \frac{Cat_m + Coord_m}{N + Coord_r}$$

The recall (proportion of points on the gold-standard path matched in the output) is defined as the same count of matching annotations, divided by the total number of chords N and annotated points of coordination in the original annotation $Coord_o$.

$$R = \frac{Cat_m + Coord_m}{N + Coord_o}$$

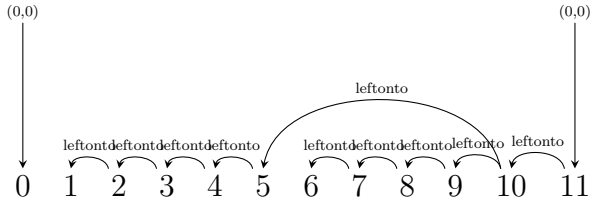
f-score F is the harmonic mean of R and P

$$F = \frac{2RP}{R + P}$$

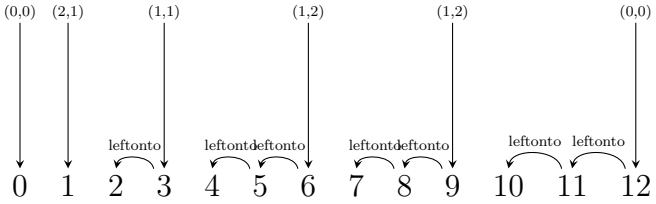
Consider two examples (4.2, 4.3), we see that when using PCCG parser, the total parses is 59. Meanwhile, when using St+PCCG parser, we only get 30 total parses. That confirms the competency of supertagging model compares with normal PCCG parser which generate more parse results that can reduce the performance of the algorithm. Among the parse results generated by PCCG, although using perfect Reranking (i.e., manually pick up best candidate without using Reranking model), the distance between the best candidate and gold standard only 3.5 compare with 3.0 generated from St+PCCG.
When applying Reranking model, they gained quite high results. Before Reranking, the distance between parse result generated by PCCG and St+PCCG parsers both get 4.5. However, after Reranking, we get the distance 3.5 for PCCG parser and 4.0 for St+PCCG parser. From two examples, we confirm the improvement in performance when using Reranking model for baseline parsers.

Gold standard (Total parses: 59)

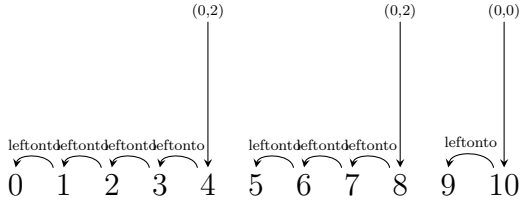
Sequence-69: “Cm GbM7 Gb7 B7 F7 Bb7 Eb Ab7 F7 Bb7 Eb Ab7 G7 Cm”



Before Reranking (Distance to gold standard: 4.5)



After Reranking PCCG (Distance to gold standard: 3.5. Found at index: 40)



Perfect Reranking (Distance to gold standard: 3.5, Found at index: 38)

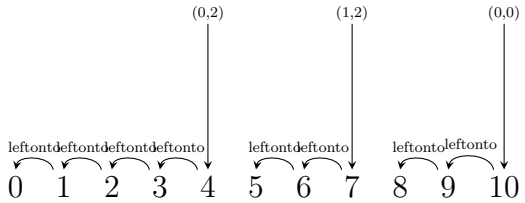
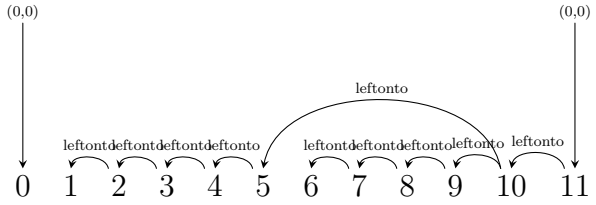


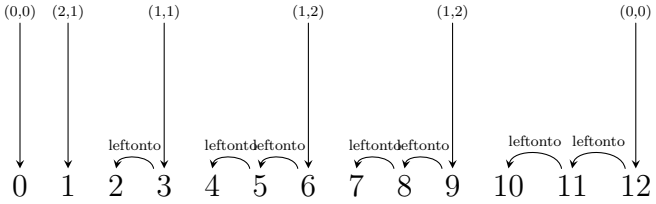
Figure 4.2: Demo Reranking with PCCG parser for sequence 69.

Gold standard (Total parses: 30)

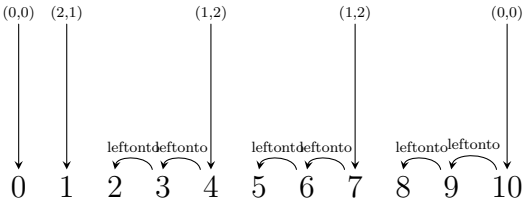
Sequence-69: “Cm GbM7 Gb7 B7 F7 Bb7 Eb Ab7 F7 Bb7 Eb Ab7 G7 Cm”



Before Reranking (Distance to gold standard: 4.5)



After Reranking St+PCCG (Distance to gold standard: 4.0, Found at index: 1)



Perfect Reranking (Distance to gold standard: 3.0, Found at index: 3)

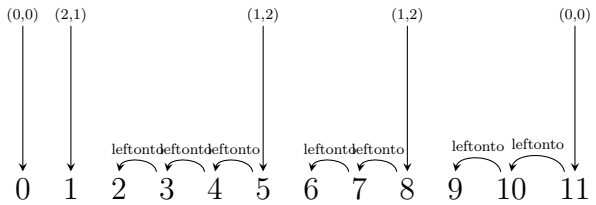


Figure 4.3: Demo Reranking with St+PCCG parser for sequence 69.

Table 4.3: Evaluation of each model's prediction using 10-fold cross-validation on the jazz corpus.

Model	P(%)	R(%)	F(%)
PCCG (Baseline)	92.29	88.78	90.50
Supertagging+PCCG (Baseline)	90.18	92.79	91.46
Perfect Reranking 5000-best PCCG	**97.18**	**96.96**	**97.07 (+6.57)**
Perfect Reranking 5000-best Supertagging+PCCG	**93.06**	**95.20**	**94.12 (+2.66)**
Reranking 10best PCCG (n-gram features)	**93.09**	**92.34**	**92.71*(+2.2)**
Reranking 1000best PCCG (n-gram features)	90.69	90.10	90.39 (-0.11)
Reranking 5000best PCCG (n-gram features)	89.07	89.54	89.30 (-1.2)
Reranking 10best Supertagging+PCCG (n-gram features)	90.53	92.30	91.40 (-0.06)
Reranking 1000best Supertagging+PCCG (n-gram features)	90.42	91.51	90.96 (-0.5)
Reranking 5000best Supertagging+PCCG (n-gram features)	**90.38**	**92.18**	**91.75*(+0.29)**

4.3 Current results

The following results are based on Reranking model with parameters w which is found by perceptron algorithm. The settings for the experiment as follow.

Implementation settings:

- Iteration $T = 10$
- Feature vector: n-gram
- 10-fold cross validation with 90% training set and 10% testing set

4.4 Results analysis

Table 4.3 shows that Reranking for 10best PCCG parser with n-gram features obtains the highest performance. The current f-score for this parser is 92.71%. Unfortunately, the performance drops down dramatically when increasing n-best parse results. This is because with more candidates and simple feature templates, the perceptron algorithm is easy to make mistake when picking the correct result.
According to Figure 4.4 and 4.5, we see that the probability of picking the right parse result from PCCG parser higher than Supertagging+PCCG parser. In PCCG parser, the parse results are generated by this parser have small distance distribution to gold-standard. Then, with TSED metric for calculate precision, recall, and f-score, the performance for PCCG parser is higher than Supertagging+PCCG parser. We can see that histogram of 5000-best PCCG distance between best parse result and gold standard with

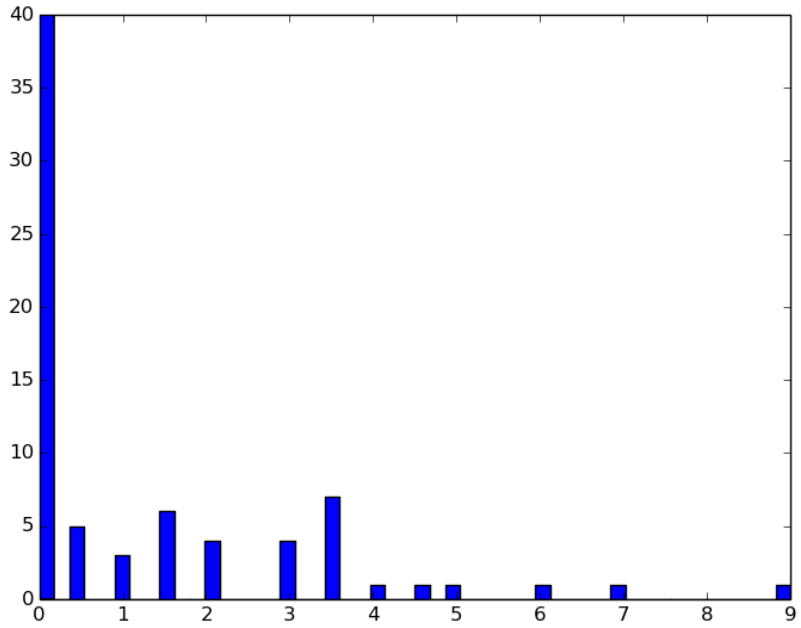


Figure 4.4: Histogram of 5000-best PCCG distance between best parse result and gold standard with $min = 0.0, max = 9.0, mean = 1.26$

maximum distance only 9 units. Meanwhile, the maximum distance between best parse result and gold standard of Supertagging+PCCG is 19 units. In general, we will have higher performance when applying Reranking model for PCCG parser.

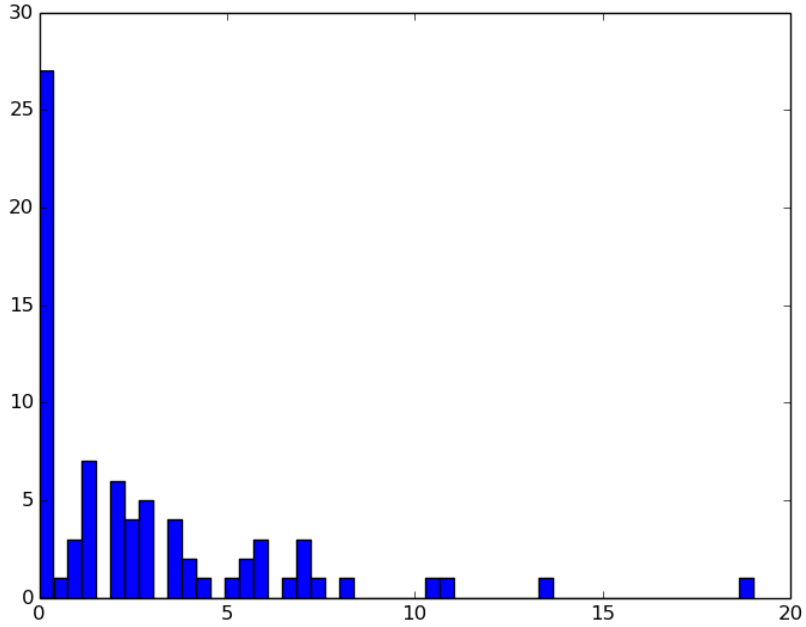


Figure 4.5: Histogram of 5000-best Suppertagging+PCCG distance between best parse result and gold standard with $min = 0.0, max = 19.0, mean = 2.74$.

Chapter 5

Conclusions

This thesis presented reranking model which is a discriminative approach. The feature template support for this model are quite simple. In current research, we only use n-gram model extracted from dependency graph of gold standard Jazz corpus. The advantage of Reranking model is easy to understand and implement with high speed in training. The drawback also exists in defining more useful feature template for the enhancement of the system.
We have used Reranking model to improve CCG parser for Jazz Chord Sequences. Our contribution is that with simple feature setting and perceptorn algorithm, we have improved performance of the system by 2.2% in PCCG parse results. This current result is not really high and good enough. However, Reranking model approach is really promising with the highest performance can be reach by manually picking the right result from list of parsing instances.
In current study, we see that the distribution of parsing results which may match the gold standard not enough to reach the higher performance. In PCCG parse results, the chance of finding match structure analysis higher than St+PCCG parse results which cause higher accuracy in final results.

5.1 Future work

In the future, we may try other methods which appropriate for chord sequence characteristics such as Conditional Random Fields. This is another log-linear model which is also define a set of feature vector in d dimension. Dynamic programming algorithm can be used for estimating the parameter for this model.
The current features are extracted from dependency graph only provide $leftonto, rightonto$ relation information between chords. In the next study, we try to extract feature from parse tree instead of dependency graph.
Consider figure 5.1, it is added more information about combinatory rules such as application rules (appf), composition rules (compf) and coordination rules (coord). With more information, we can extract more useful feature for Reranking model and might achieve more higher accuracy for the system.

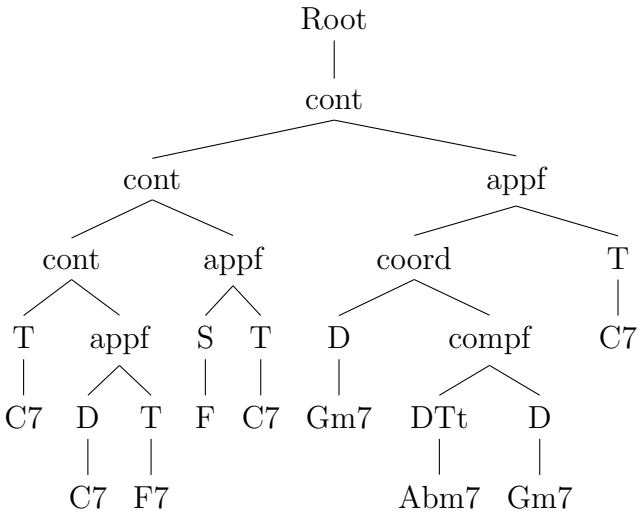


Figure 5.1: Parse tree for sequence C7 C7 F7 F C7 Gm7 Abm7 Gm7 C7.

From this parse tree, we can define feature template such as tree topology feature which describe the shape of parse tree (preference for right-branching, parallelism in conjunctions), local context feature which are ancestor paths of nodes in the tree, CCG which is combination of unary rule applications, and heads of conjunctive structures.

Bibliography

[1] Mario Baroni, Simon Maguire, and William Drabkin. The concept of musical grammar. *Music Analysis*, 1983.

[2] Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL '05)*, 2005.

[3] Elaine Chew. Towards a Mathematical Model of Tonality. *PhD thesis, MIT*, 2000.

[4] Andrew Choi. Jazz Harmonic Analysis as Optimal Tonality Segmentation, 2011.

[5] Yen-Lu Chow and Richard Schwartz. The n-best algorithm: An efficient and exact procedure for findingthe n most likely sentence hypotheses. *In Proceedings of the IEEE International Conference on Acoustic, Speech, and Signal, Processing*, 1990.

[6] Stephen Clark and James R. Curran. Wide-coverage efficient statistical parsing with CCG and loglinear models. *Computational Linguistics*, 2007.

[7] Michael Collins. Three Generative, Lexicalised Models for Statistical Parsing. *In The Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, San Francisco. Morgan Kaufmann.*, 1997.

[8] Michael Collins and Terry Koo. Discriminative Reranking for Natural Language Parsing. Technical report, Technical report, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 2005.

[9] Deryck Cooke. *The Language of Music*. 1959.

[10] Grosvenor Cooper and Leonard B. Meyer. *The Rhythmic Structure of Music*. 1960.

[11] JR Curran D Ng, M Honnibal. Reranking a wide-coverage CCG parser. *Proceedings of ALTA*, 2010.

[12] David Cope. *Computer Models of Musical Creativity*. MIT Press, 2005.

[13] David Temperley. *The Cognition of Basic Musical Structures*. MIT Press, 2001.

[14] Eugene Narmour. *The Analysis and Cognition of Basic Melodic Structures*. Univ of Chicago Pr, 1991.

[15] Leonhard Euler. *Tentamen novae theoriae musicae ex certissismis harmoniae principiis dilucide expositae*. Saint Petersberg Academy, 1739.

[16] Fred Lerdahl. *Tonal Pitch Space*. Oxford University Press, 2004.

[17] Fred Lerdahl and Ray Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, 1983.

[18] Mark Granroth-wilding. *Harmonic analysis of music using combinatory categorial grammar*. PhD thesis, University of Edinburgh, 2013.

[19] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. Implementing A Generative Theory of Tonal Music. *Journal of New Music Research*, 2006.

[20] Julia Hockenmaier and Mark Steedman. Generative Models for Statistical Parsing with Combinatory Categorial Grammar. *In Proceedings of the 40th Meeting of the Association for Computational Linguistics*, 2002.

[21] James H. Jeans. *Science and Music*. 1937.

[22] Johnson-Laird, Phil N.Kang, Olivia E.Leong, and Yuan Chang. On musical dissonance. *Music Perception*, 2012.

[23] P. N. Johnson-laird. Jazz improvisation: A theory at the computational level. *In Howell, P., West, R., and Cross, I. J., editors, Representing Musical Structure, Academic Press Ltd., San Diego, CA.*, 1991.

[24] Carol L. Krumhansl. *Cognitive Foundations of Musical Pitch*. Oxford University Press, Oxford, 1990.

[25] Hugh Christopher Longuet-Higgins. *Letter to a musical friend*. The Music Review, 1962.

[26] Hugh Christopher Longuet-Higgins and Mark Steedman. *On interpreting Bach*. Machine Intelligence, 1971.

[27] Mark Granroth-Wilding and Mark Steedman. A Robust Parser-Interpreter for Jazz Chord Sequences. 2014.

[28] Leonard B. Meyer. *Emotion and Meaning in Music*. University of Chicago Press, Chicago, IL, 1956.

[29] François Pachet. Computer analysis of jazz chord sequences: Is Solar a blues? *Readings in Music and Artificial Intelligence, Harwood Academic Publishers*, 2000.

[30] Mark Steedman and Jason Baldridge. Combinatory Categorial Grammar. 2011.

[31] Mark J. Steedman. A Generative Grammar for Jazz Chord Sequences. *Music Perception*, 1984.

[32] Sharon Thompson-Schill, Peter Hagoort, Peter Ford Dominey, Henkjan Honing, Stefan Koelsch, D. Robert Ladd, Fred Lerdahl, Stephen C. Levinson, and Mark Steedman. Multiple Levels of Structure in Language and Music. 2013.

[33] Barbara Tillmann and Emmanuel Bigand. The Relative Importance of Local and Global Structures in Music Perception. *The Journal of Aesthetics and Art Criticism*, 2004.

[34] Satoshi Tojo, Keiji Hirata, and Masatoshi Hamanaka. Computational Reconstruction of Cognitive Music Theory, 2013.

[35] Dmitri Tymoczko. *A Geometry of Music: Harmony and Counterpoint in the Extended Common Practice*. Oxford University Press, Oxford, 2011.

[36] David Wright. Mathematics and Music. *Springer Proceedings in Mathematics*, 2009.

[37] Yue Zhang and Stephen Clark. Shift-Reduce CCG Parsing. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011.