A POWERFUL TOOL OF COMPOSING MELODY FOR DIFFERENT LANGUAGES

by

PENGCHAROEN Pakawadee

A Thesis Submitted to

The Hong Kong University of Science and Technology

in Partial Fulfillment of the Requirements for

the Degree of Master of Philosophy

in the Department of Electronic and Computer Engineering

August 2015, Hong Kong

Authorization

Pakawadee Pengcharoen

PENGCHAROEN Pakawadee

August 2015

A POWERFUL TOOL OF COMPOSING MELODY FOR DIFFERENT LANGUAGES

by

PENGCHAROEN Pakawadee

This is to certify that I have examined the above MPhil thesis
and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by
the thesis examination committee have been made.

[illegible]

Prof. Danny Hin-Kwok TSANG (ECE)
Thesis Supervisor

[illegible]

Prof. Raymond Chi-Wing WONG (CSE)
Thesis Co-supervisor

[illegible]

Prof. Wei ZHANG (ECE)
Chairman of Thesis Examination
Committee

[illegible]

Prof. Gary Shueng-Han CHAN (CSE)
Thesis Examination Committee Member

[illegible]

Prof. Bertram SHI (ECE)
Head of Department

Department of Electronic and Computer Engineering

August 2015

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to everyone who contributed to this research. First and foremost, I would like to express my special appreciation to my supervisors, Prof. Danny Hin-Kwok Tsang and Prof. Raymond Chi-Wing Wong for their guidance, persistent help and support in all the time of my research. I could not have imagined having better advisors and mentors for my MPhil study.

Besides this, I would like to thank Prof. Gary Shueng-Han Chan and Prof. Wei Zhang for being on my thesis examination committee.

Additionally, I would like to thank Prof. Sukree Sinthupinyo for giving me such a great opportunity, help, support and encouragement for this MPhil study.

My sincere thanks also goes to all my colleagues and friends: Jean-loup Lamothe, Naveen Pitipornvivat, Xiang Zhang, Yuxuan Jiang, Bo Sun, Susie Yuan, Cindy Chen and Coleman Yu for their friendship, kindness, support along the way and all the fun we have had during these two years. Also, I thank my friends at Chulalongkorn University for their help and advice. Also, thanks to all of my friends for assisting in my thesis experiment.

My deepest appreciation goes to my family: my parents, Prasit Pengcharoen and Samaporn Pengcharoen, for their endless love and the support that they have provided me through my entire life and in particular. Also, special thanks to my sister, Narissara Pengcharoen, for assisting and advising on my research. Lastly, my sincere thanks to my boyfriend, Thanakorn Chindanonda, for his endless love and unconditional support along the way. Without their care and encouragement, this thesis would not have been completed. I will forever be grateful to them.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

A POWERFUL TOOL OF COMPOSING MELODY FOR DIFFERENT LANGUAGES

by

PENGCHAROEN Pakawadee

Department of Electronic and Computer Engineering

The Hong Kong University of Science and Technology

Abstract

Song is a composition of music and lyrics that can express both feeling and story. Composing a song is not a trivial task for a novice because it needs a knowledge of music theory and experience in a musical instrument. Nowadays computers have been proven to be powerful tools for several fields including the music field. A music generator is a common tool for helping a novice to create a song. The thesis presents two crucial elements for a robust music generator, a melody composer and a statistical music accompaniment generator. The melody composer aims to compose a pleasing melody for Thai and Japanese lyrics based on a lyric-note correlation. The statistical music accompaniment generator is designed based on a probabilistic model to generate suitable accompaniment with different rhythmic patterns for a given melody. Generally, there is no single correct melody for lyrics and there is no single correct accompaniment for a particular melody. Therefore, two studies are conducted to evaluate the effectiveness of both systems in terms of subjective quality.

CHAPTER 1
INTRODUCTION

A song is a composition of music and lyrics that can express both feeling and story. In general, to compose a song, a songwriter needs to create a melody and develop music accompaniment respectively. Both knowledge of music theory and experience in a musical instrument are crucial. It is not a trivial task for a novice to compose a song.

Computers have been proven to be powerful tools for several fields, and this is also so in the music field. A music generator is a common tool for helping a novice to create a song. In general, a music generator combines music theoretical knowledge, such as rules of musical structure, with algorithms and is implemented in a user-friendly interface, which makes it easy, efficient and enjoyable, such as in an interactive and algorithmic composition system, a real-time interactive music system or musical tools for entertainment and education. There are several popular commercial products, and some examples are shown in the following section.

1.1 Music Composition Software

1.1.1 Band-in-a-Box

The Band-in-a-Box (BIAB) is very popular music software produced by PG Music Incorporated [1]. It provides a back-up-band or computer-generated accompaniment for a musician to play along with. The software allows users to adjust a tempo, adjust a key, choose instruments and select musical styles. Users can create their own musical styles in the program. There is a notation window for editing notes. Moreover, a composed song can be exported as either a MIDI or an audio file.

1.1.2 JAMMER Professional

JAMMER Professional is a product of SoundTrek. It is fast and easy software for creating musical accompaniment in a wide variety of styles by entering some chords on the lead sheet, choosing a musical style and then pressing the compose button [2].

1.1.3 Songsmith

Songsmith is a musical accompaniment application developed by Microsoft Research [3]. The product began from a research project called MySong [4]. Songsmith generates a musical accompaniment for a voice that a user records to the system. The system allows the user to adjust tempo, genre and overall mood, such as happy, sad or jazzy.

1.1.4 One Man Band

One Man Band is a virtual music-keyboard with real-time automatic accompaniment [5]. The styles of the accompaniment are utilized from the standard style files used in Yamaha, PSR, DGX and Tyros arranger keyboards.

1.1.5 Harmony Assistant

Harmony Assistant is software for music composition and editing [6]. The software includes accompaniment generation, instruments selection and drum pattern generation. With the editing function, the software allows users to edit a score performed through a mouse, computer keyboard or external MIDI keyboard.

We can see that, based on the above research, there are several commercial music composition software products. In general, this software is implemented by using algorithmic composition techniques. The following section is the overview of the algorithmic composition models.

1.2 Algorithmic Composition Models

Algorithmic composition is a process in which a song is automatically composed by algorithms. The different compositional algorithms can be categorized by the methodological approaches [7], [8].

1.2.1 Grammars

In formal language, formal grammar is a set of rules for symbols or strings. Music can be described by a grammar set in the same way as a language. In music composition, grammar is constructed by rules which are derived by hand from principles in music theory or examined in a corpus of pre-existing musical compositions. The symbols are generated by mapping musical objects such as rhythms, harmonies and notes. There have been several studies on music composition by using grammars, such as in [9] and [10]. However, this approach seems to be very difficult to effectively implement if the system is complex.

1.2.2 Knowledge-based Systems

Knowledge-based systems are based on the musical structure various composition rules from music theory. Based on this model, a new piece of music is generated based on a pre-made set of arguments as knowledge which may be dynamically changed or learned, such as in [11] and [12].

1.2.3 Markov Chains

Markov chains are stochastic processes of state transitions without memory. The next state depends on only the transition probability of the current state. There have been several studies that have applied Markov chains for music composition [13], [14], [15]. The states of the Markov chain are mapped to musical objects, such as notes or pitch values. The transition probabilities can be either derived by hand from principles in music theory or trained from a corpus of pre-existing musical compositions. The music is composed as a result of non-deterministic methods by weighting the possibilities of random events.

Hidden Markov Models (HMMs) are Markov processes with hidden states which are not visible. However, the output of the state can be observed, so HMMs involved in the transition probabilities and the output probabilities which make them can find a globally optimized sequence of states. HMMs are very popular to use in music composition, especially for harmonization [16], [17].

1.2.4 Artificial Neural Networks

Artificial Neural Networks (ANNs) are statistical models inspired by biological neural networks. ANNs consist of interconnected neurons sending messages to each other and the connection between neurons has a weight. Generally, there are several connected layers of neurons as recurrent networks. The computational process of ANNs is firstly using a training set for learning and setting a weight for each connection between neurons. Then the input neurons, which are activated by the input of the system, pass on to other neurons. Note that a neuron will be activated by being weighted and transformed by a function. The process is repeated until the output neuron is activated. Typically, the output is generated in a similar pattern to the training set. There have been several studies on music composition using ANNs [18], [19], [20]. Using ANNs in music composition, a corpus of pre-existing musical compositions is needed for the training phase. Also, a mapping between musical objects and the input and output of the network is necessary.

1.2.5 Evolutionary Algorithms

Evolutionary Algorithms (EAs) are inspired by biological evolutionary processes, such as selection, mutation, recombination and reproduction. Candidate solutions play the role of individuals in a population, and the quality of the solutions is determined by a fitness function. There is a common pattern for EAs, as set out in the following two steps. The first step is candidate generation. The candidate solutions are generated by either a specified method or random method, and each candidate is evaluated by the fitness function. The next step is selection. The existing population is selected to breed a new generation. The better solution measured by the fitness function tends to be selected with higher probability. To increase the variation, some operators are used, such as mutation or recombination. These two steps are iteratively applied so that the worse solution is cut out and the results tend to be better. The evolutionary methods in music composition are based on genetic algorithms. The operators, such as mutation and selection, make the different solutions evolve to be fitted with music. The difficult part of using EAs in music composition is how to define the fitness function. Different studies have proposed different fitness functions [21], [22], [23], and a common way of implementing the fitness function is a weighted sum of the features of the musical composition.

1.2.6 Hybrid Systems

Hybrid Systems are systems using more than one algorithm in order to improve the results. In music composition, there are many hybrid systems, for example, the hybridization of Markov chains with other models. Verbeurgt et al. [24] proposed a hybrid approach for music composition combining Markov chains and neural networks in which the pattern of melody is generated by the Markov chains and then refined by the neural networks. In another hybrid approach, Thornton [25] proposed combining a set of grammar rules with Markov models. The rules are defined from the existing composition, inferring a hierarchy of Markov models. And Manaris et al. [26] proposed music analysis and composition incorporating neural networks with a genetic algorithm. These hybrid systems have one concern, however, which is the complexity and resources to combine the algorithms.

Based on the above research of algorithmic composition models, we can see that there are many methods for music composition. Although this research area has been studied for many decades, it is still open for further study and the proposal of new models and algorithms. The related work of this thesis is discussed in the next chapter.

CHAPTER 2
RELATED WORK

The work of this thesis can be divided into two domains of research, which are a melody generator for music novices and automatic accompaniment generation.

2.1 Melody Generator for Novices

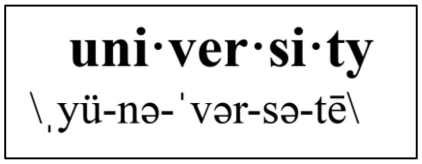
To help novices to compose music, a melody generator is developed with a user-friendly interface. The algorithm behind this system is related to the algorithmic composition described in the previous chapter. Most of the existing work on algorithmic composition considers the temporal correlation among all notes of the melody. However, there is an interesting existing work T-Music, which is a melody composer based on frequent pattern mining, proposed by Cheng et al. [27]. This melody composer considers not only the temporal correlation but also the lyric-note correlation.

2.1.1 T-Music Overview

The T-Music system is a user-friendly tool for a user to create a song, even if the user has no knowledge of music theory. Users can compose a melody simply by inputting lyrics and settings to the system. Then the system immediately generates a set of melodies corresponding to the lyrics. The system supports three different languages of lyrics: English, Cantonese and Mandarin.

2.1.2 Lyric-note Correlation

Lyrics are the words of a song, and each word contains at least a single vowel sound called a syllable, which is associated with a tone. Tone is a pitch in language to express emotion and to convey emphasis or contrast. Tones appear in many languages, such as English, Mandarin, Cantonese, Thai and Japanese. For example, in English there are three kinds of stresses or tones, which are non-stress, primary stress and secondary stress. In general, the stressed syllables are often louder than non-stressed syllables, the primary stress sounds with a higher pitch and the secondary stress sounds with a lower pitch. In the International Phonetic Alphabet (IPA), the primary stress is indicated by a high vertical line and the secondary stress is indicated by a low vertical line before the syllable. For example, Figure 2.1 shows five syllables with tones in the IPA style of the word 'university' from the Merriam-Webster dictionary. The first syllable, which is \ˌyü\, has a secondary stress, and the third syllable, which is \ˈvər\, has a primary stress.



uni·ver·si·ty
\ˌyü-nə-ˈvər-sə-tē\

Figure 2.1 Syllables and tones of word 'university'

To generate the set of melodies corresponding to the lyrics, the system considers the lyric-note correlation. According to the definition in [27], the lyric-note correlation corresponds to the correlation between the changing trend of a sequence of consecutive *notes* and the changing trend of a sequence of consecutive *words*. The changing trend of a sequence of notes corresponds to a series of *pitch differences* between every two adjacent notes since each note has its pitch. The changing trend of a sequence of words corresponds to a series of *tone differences* between every two adjacent syllables since each syllable has its tone.

2.1.3 System Architecture

The system has two phases, which are Probabilistic Automaton (PA) building and Melody Composition, as shown in Figure 2.2.

In Phase I, the system is given a song database containing songs with their own lyrics. The first process is that for each song in the database, its lyrics are processed. The lyrics are segmented into syllables and then each syllable is extracted to obtain its note and its tone is then stored in the sequence. Next, the obtained sequence is mined to find all frequent patterns, and finally, the PA is built based on the frequent patterns.

In Phase II, the system is given lyrics by a user. The process is that the given lyrics are processed to extract the tones of all syllables, obtaining a tone sequence. Then the constructed PA is executed by using the obtained tone sequence as an input to generate a new melody as an output.

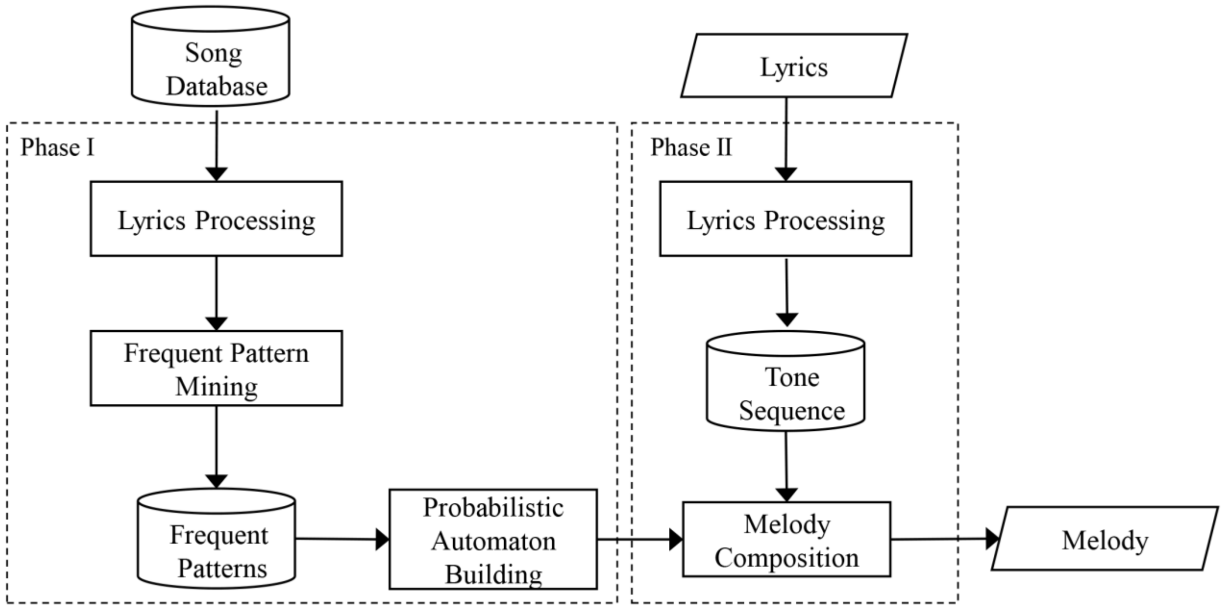


Figure 2.2 Architecture of T-Music

2.2 Automatic Accompaniment Generation

Accompaniment of melody is a common feature of music. Automatically generating the accompaniment by computer is a challenging and interesting music research topic. There have been several studies on this, and a number of approaches have been used for accompaniment generation.

Pardo and Birmingham [28] proposed their own algorithm for harmonic analysis, called Segment Labeling. There are two main tasks for the analysis. First, segmentation is a task to split the music into appropriate segments. Second, labeling is a task to give a suitable root pitch for each segment. The algorithms are based on template matching and graph-search techniques. However, for this proposed method, the labeling task performs well when given the correct segmentation.

Neural networks also have been used for accompaniment generation, especially for interactive systems or real-time applications. Cunha and Ramalho [29] introduced a system generating real time accompaniment. The system uses a hybrid model which combines neural networks with a rule-based sequence tracker. The neural networks are for gaining prior knowledge such as chord sequences and the recurrent chord sequences can be predicted by the rule-based sequence tracker.

Gang and Lehmann [30] also proposed a method for generating accompaniment for melody in live performance situations using neural networks. The model was evaluated by measuring the difference between the resulting chord and the chord from the author of the source book using distance functions.

Currently, machine learning is being adapted to accompaniment generation. Specifically, probabilistic models have been used extensively in several studies for accompaniment prediction, as demonstrated in the following examples.

Allan and William [16] created a harmonization system by using two HMMs to generate chorale harmonization. The training data is pieces by Johan Sebastian Bach. The first HMM generates an accompaniment by selecting a sequence of note intervals for a given melody. The second is for ornamentation. Although this method generates the harmony lines for a melody, it uses chords as an intermediate representation. However, the model is not applicable for the general setting because polyphonic music with a specific four voice structure has to be provided as input.

Paiement et al. [31] proposed a graphical model for accompaniment generation given a melody or root note progressions. This model is more general in that an input can be any melody, without regard to specific structure. This makes the model flexible and able to be integrated into other automatic composition systems. The results show that the tree structure can capture the chord progressions by considering global dependencies of the root notes. However, the model is restricted by the fixed length of the sequence.

Chuan and Chew [32] proposed a hybrid system combining music theory and statistical learning for automatically generating style-specified accompaniment. The system has several computational approaches, which are the machine learning technique for chord tone determination, the Neo-Riemannian transformation for building the chord progressions of consecutive checkpoints and Markov chains for final chord progression generation. Also, this system considers the phrase structure information by adding phrase position attributes, such as start, middle, end or bridge, into the melodic representation. This helps the system to determine the checkpoints, which are very important for the harmonization process because this system does not have a sequential processing time. The chords are assigned at the checkpoints first, and then the bars between the checkpoints will be determined by the chord progressions. The results of studies show that the accompaniment generated by this system is close to the original and the chord transitions support the phrase structure of the melody.

Simon et al. [4] proposed MySong, which is an interactive system automatically generating accompaniment for a vocal melody using a Hidden Markov Model (HMM). A user can interact with the system by singing into a microphone, and then corresponding accompaniment will be generated for that vocal melody. An interesting part of this research is that the system was evaluated in terms of subjective quality by two studies. The results show that the accompaniment generated by the system receives a rating score similar to that of accompaniment generated by musicians. Moreover, the results show that the system is easy to use for people with no knowledge of music theory. This system has been applied by using user-oriented machine learning [17] and has been adjusted to expose the model and algorithm parameters to end-users. This allows the user to have more interactivity with the system, leading to more creative expressiveness. However, the model is limited by the assumption of measure boundaries. The measure boundaries in the training phase are determined from the training data, and the measure boundaries of the vocal melody are determined by the timing of the drum beat along with which the user sings. These mean that every chord is considered to have the same length, which eliminates information in terms of duration. To illustrate, a chord with a single beat duration is different in its harmonic implication from the same chord with a full measure duration.

To incorporate the duration information, Groves [33] proposed an automatic accompaniment system using a Hidden Semi-Markov Model (HSMM). HSMM is an extension of the probabilistic model in that each state can have its own sub-sequence of observations. In the model, an extra parameter, which is duration, is inserted into the state sequence information so each state is considered for all its possible durations. Although this model can capture the duration information, there are many limitations. For example, more training data is needed, because of the many possible durations of chords. Also, there is an assumption that the key signature of the input melody is known. Lastly, from the results in [33], there are some dissonant chords which have only one single note length occurring in the melody. The reason is that the probabilities of the note observation over a chord are slightly overridden by that chord's sequence probabilities.

Lastly, another a type of finite state machine, a finite state transducer (FST), which is commonly used in natural language processing and speech recognition, has been applied in music-related tasks. Forsyth and Bello [34] introduced an accompaniment generation system for a melody using an FST. A quantization of a music signal into a finite set of symbols, which is a computed codebook or binary templates, was proposed. The system was evaluated by two metrics, accuracy and per-symbol distortion. However, there was no evaluation in terms of human perception of the quality of the generated accompaniment.

CHAPTER 3
CONTRIBUTIONS

Motivated by the existing works and the limitations of the existing systems, the main contributions of this thesis are as follows.

1. Improving the existing system T-Music to support different languages for melody composition. Currently, the system supports lyrics in English, Cantonese and Mandarin. To make the system a more powerful tool, the Thai and Japanese languages are integrated, which allows users to compose melodies with lyrics of these languages. The lyrics processing for the Thai and Japanese languages is presented. Also, the challenging parts of this extension are syllabification and automatic alignment processes for a phrase or sentence without any explicit word boundary.

2. A statistical music accompaniment generator is proposed to overcome the limitations of the existing works. The system is based on a probabilistic model, generating appropriate accompaniment with different rhythmic patterns for a given melody. A phrase structure is also considered in the model in order to generate the appropriate accompaniment. The system is also easy to plug into any music composition system because it does not require specific input format and a chord symbol mapping. Moreover, it does not require a lot of training data so it can generate personalized accompaniment to match the interests and tastes of the user by the user inputting a song he/she likes into the database.

3. The results of two studies to evaluate the effectiveness of the two systems are presented. The evaluation includes the effectiveness of subjective-acceptable generated melodies and subjective-acceptable generated accompaniments. The results are also analyzed and discussed.

The rest of thesis is organized as follows. Chapter 4 introduces a method of composing melody for Thai and Japanese. Chapter 5 proposes a statistical music accompaniment generator. Chapter 6 reports the experimental results of both systems, and chapter 7 discusses the results of two studies. Finally, chapter 8 concludes this thesis and proposes future work.

CHAPTER 4
COMPOSING MELODY FOR DIFFERENT LANGUAGES

As mentioned in Chapter 2, the T-Music system composes a melody based on given lyrics from a user. In Phase II of the system, the given lyrics are processed for tone extraction and automatic alignment. In tone extraction, the tones of all syllables are extracted from the lyrics to obtain a tone sequence for melody composition in the next step. However, the process of tone extraction depends on the language of the given lyrics because each language has distinctive tone patterns. For improving the T-Music system to compose melodies for different languages, the lyrics processing needs to be improved to support different languages. This leads to two problems, tone extraction and automatic alignment in different languages based on given lyrics. In this thesis, two languages, Thai and Japanese, are integrated into the system.

4.1 Problem Statement

4.1.1 Tone Extraction

Tone extraction is a problem of finding a tone sequence given lyrics in different languages. This problem can be formulated as follows.

Given lyrics *L*, a *syllable sequence* containing syllables of lyrics *L* can be defined as

$$S(L) = \{l_1, l_2, l_3, \dots, l_n\}, \tag{1}$$

where *n* is a number of syllables of *L* and l_i is the *i*th syllable of *L*, where $i \in [1, n]$. Moreover, a *phonetic sequence* containing syllables of phonetic symbols of *L* can be defined as

$$P(L) = \{p_1, p_2, p_3, \dots, p_m\}, \tag{2}$$

where *m* is a number of syllables of phonetic symbols of *L* and p_j is the *j*th syllable of phonetic symbols of *L*, where $j \in [1, m]$.

For each syllable of the phonetic sequence $P(L)$, there is a corresponding tone. The tones of all syllables of the phonetic sequence can be defined as a *tone sequence* as follows:

$$T(L) = \{t_1, t_2, t_3, \dots, t_m\}, \tag{3}$$

where t_j is a tone corresponding to p_j.

The following is an example of finding a tone sequence given English lyrics. Suppose that the given lyrics are L = "When I see you again". In the tone extraction process, a syllable sequence of the lyrics is obtained as *S(L)* = {When, I, see, you, again}, and a phonetic sequence containing phonemes of L is obtained as $P(L)$ = {wen, ˈaɪ, ˈsi, ju, ə, ˈgen}. Note that in English the tone of a phoneme is obtained from its kind of stress. This means that each phoneme of these lyrics has a stress pattern of non-stress, primary stress, primary stress, non-stress, non-stress and primary stress respectively. Suppose that each of the three types of stresses can be represented by an integer value as follows: 0 represents a non-stress, 1 represents a primary stress and 2 represents a secondary stress. Finally, a tone of the sequence of these lyrics can be obtained as $T(L)$ = {0, 1, 1, 0, 0, 1}.

4.1.2 Automatic Alignment

Automatic alignment is a problem of finding pairs of syllables and corresponding phonemes. This problem can be formulated as follows.

Given a syllable sequence of lyrics L as $S(L) = \{l_1, l_2, l_3, \dots, l_n\}$ and a corresponding phonetic sequence $P(L) = \{p_1, p_2, p_3, \dots, p_m\}$, a sequence of alignments of the lyrics L can be denoted as

$$A(L) = \{(l_1, r_1), (l_2, r_2), \dots, (l_n, r_n)\}, \tag{4}$$

where the list of phonemes associated with each l_i are denoted as $r_i = (p_{i_1}, \dots, p_{i_k})$ such that $(r_1, \dots, r_n) = (p_1, \dots, p_m)$.

From the previous example, *S(L)* = {When, I, see, you, again} and $P(L)$ = {wen, ˈaɪ, ˈsi, ju, ə, ˈgen}, a sequence of alignments can be obtained as *A(L)* = {(When, (wen)), (I, (ˈaɪ)), (see, (ˈsi)), (you, (ju)), (again, (ə, ˈgen))}.

4.2 Lyrics Processing in the Existing System

The existing system already supports three languages, English, Cantonese and Mandarin. The processes of tone extraction and automatic alignment are as follows.

4.2.1 Tone Extraction

For English, a tone sequence is obtained based on the stress of each syllable of given lyrics. First, for each word, finding the corresponding phonetic symbols is done by searching through a dictionary containing tuples of a word with corresponding phonetic symbols and corresponding parts of speech. This is a trivial task because words in English sentences are separated by a space. Next, for each phonetic symbol, partitioning into syllables can be done by analyzing syllable boundaries, such as the positions of the starting consonant, vowel or ending consonant, based on spelling and pronunciation. Next, for each word, a stress pattern can be obtained by analyzing the stress of each syllable based on the corresponding phonetic symbols. If there is no information for the corresponding phonetic symbols, the stress pattern will be obtained by considering the number of syllables as different numbers of syllables obtain different stress patterns. Finally, the tone sequence is obtained based on the stress patterns of all words.

The processes of tone extraction for Cantonese and Mandarin are different from the process for English because of the different linguistic structures. In Cantonese and Mandarin, a tone sequence is obtained based on the tone of each syllable. In the structure of these language, one character has exactly one syllable with corresponding pronunciation and tone. One character can have many possible pronunciations depending on the word. For example, the character '见' can be pronounced as 'gin3' when the character is in the word '见外' or 'jin6' when the character is in the word '见龙在田'. Note that the number is the corresponding tone of the pronunciation. Cantonese has been defined to have six tones and Mandarin has been defined to have five tones. The corresponding pronunciation and tone of each character can be obtained as follows. First, for each character, the most frequent pronunciation and tone appearing in a given dictionary containing tuples of the character with the corresponding pronunciation, corresponding tone and list of words containing that character are found. Next, for each character, the next concatenated characters, at most four characters, are combined as a new word and then looked up in the word list of the first character.

If the new word exists in the word list, the corresponding pronunciation and tone of the first character will be modified to the corresponding ones of that word list. Finally, the tone sequence is obtained based on the tones of all characters.

4.2.2 Automatic Alignment

In English, Cantonese and Mandarin the automatic alignment methods are trivial. In English, each syllable can be mapped to the corresponding phoneme(s) based on mapping between English characters and phonetic symbols. In Cantonese and Mandarin, each Chinese character is monosyllabic, which means that the alignment is a one-to-one mapping between a syllable and a phoneme.

4.3 Lyrics Processing for Thai

4.3.1 Thai Orthography

Thai script is derived from the Khmer script, which is modeled after the Brahmic script of ancient India. Thai has a complex orthography (the methodology of writing a language). For example, there is a silent letter to preserve original spellings and several letters represent the same sound.

Thai text consists of a string of characters without explicit word boundaries. It is written in a sequence from left to right. There is no inter-word spacing but sometimes there is inter-phrase spacing for ease of reading. For example, the sentence “ฉันรักคุณ” composes of three words: “ฉัน”, “รัก” and “คุณ”.

Group	Characters
Consonants	ก, ข, ฃ, ค, ฅ, ฆ, ง, จ, ฉ, ช, ซ, ฌ, ญ, ฎ, ฏ, ฐ, ฑ, ฒ, ณ, ด, ต, ถ, ท, ธ, น, บ, ป, ผ, ฝ, พ, ฟ, ภ, ม, ย, ร, ฤ, ล, ฦ, ว, ศ, ษ, ส, ห, ฬ, อ, ฮ
Leading vowels	เ, แ, ไ, ใ, โ
Trailing vowels	ะ า ◌ำ
Upper vowels	◌ิ ◌ี ◌ึ ◌ื
Lower vowels	◌ุ ◌ู
Tonal marks	◌่ ◌้ ◌๊ ◌๋
Other special marks	◌็ ◌์ ◌ั ๆ ฯ

Table 4.1 Seven groups of Thai characters

Thai characters are classified into seven groups based on their positions and functions in a word, as shown in Table 4.1. These characters can be compounded based on the rules of the Thai writing system to be a syllable. Moreover, Thai has five tones: mid, low, falling, high and rising. Tone is indicated by an interaction of the initial consonant, the vowel, the final consonant (if present), and sometimes a tone mark. A particular tone mark may denote different tones depending on the initial consonant. In phonological representation, each syllable is made up of three parts:

(1) A consonant or combined consonants
(2) A vowel or a vowel with a final consonant
(3) A tone mark

For example, a consonant, ‘ก’ (k), is compounded with a vowel, ‘ะ’ (a), obtaining ‘กะ’ (kà).

4.3.2 Overview of Lyrics Processing for Thai

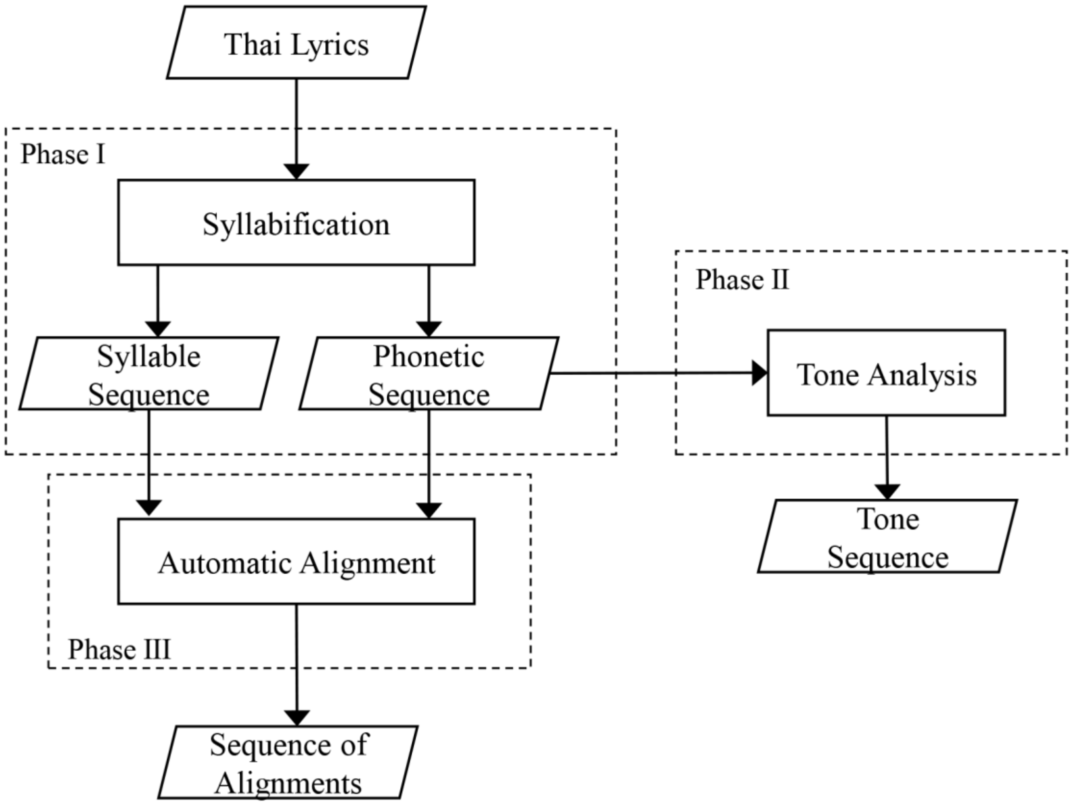


Figure 4.1 Overview of lyrics processing for Thai

Figure 4.1 shows the lyrics processing for Thai. The process is composed of three phases. The tone extraction process is in phase I and phase II, which will be described first. The last phase is the automatic alignment, which will be described in Section 4.3.5. The first phase is a syllabification of given Thai lyrics, obtaining a syllable sequence and a phonetic sequence. Next, all phonemes in the obtained phonetic sequence are analyzed to find their corresponding tones, obtaining a tone sequence in phase II.

Both the syntactic and semantic structures of the Thai language are complex. As mentioned, there are no explicit word boundaries in the Thai language, and this complicates the syllabification in phase I. Another major problem is how to handle unknown words or ambiguous words that can be differently pronounced. For example, the word "กรณี" can be pronounced as "ka-ra-nee" or "korn-nee". However, Thai dictionary defines the correct pronunciation which is "ka-ra-nee". Thus, using the dictionary can handle the issue of the ambiguous words.

There are many previous studies related to Thai syllabification. Pooworawan [35] introduced a dictionary-based approach. The method uses forward longest matching search and tries to match the string with a dictionary. Kongsupanich [36] also proposed a dictionary-based approach in which the main idea is to remove each character from a sentence backward and then try to match remaining characters with a dictionary. Generally, the dictionary-based approach is simple. However, there are limitations when the word is unknown or ambiguous. This also means that the results depend on the coverage of the dictionary. Lorchirachoonkul [37] introduced an algorithm segmenting a string into tokens and then using a precedence matrix to build syllables, and Jucksriporn et al. [38] proposed Thai syllabification using minimum character clustering and a trigram statistical model. These approaches were proposed to overcome the limitations from unknown and ambiguous words. Moreover, there have also been some hybrid approaches proposed in order to handle these limitations. For example, Khruahong [39] proposed a hybrid approach combining a dictionary-based with a pattern-based strategy.

To overcome the mentioned limitations, this thesis proposes a new hybrid approach for Thai syllabification. The approach combines a dictionary-based approach with a rule-based approach.

4.3.3 Hybrid Approach for Thai Syllabification

This section introduces a new hybrid approach combining a dictionary-based with a rule-based approach for Thai syllabification. The model has two main phases, as shown in Figure 4.2. The first phase is syllabification based on a dictionary. Given lyrics and a dictionary, the lyrics are segmented into words by using the Iterative Segmentation algorithm proposed in [40], as shown in Algorithm 1.

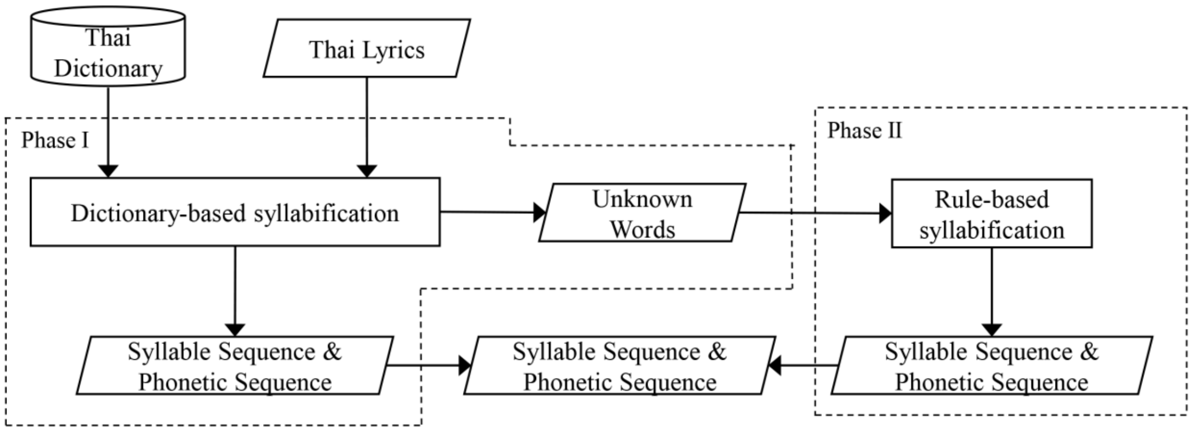

Thai Dictionary
Thai Lyrics
Phase I
Phase II
Dictionary-based syllabification
Unknown Words
Rule-based syllabification
Syllable Sequence & Phonetic Sequence
Syllable Sequence & Phonetic Sequence
Syllable Sequence & Phonetic Sequence

Figure 4.2 Thai syllabification model

Algorithm 1 Iterative Segmentation

Input: Lyrics L with length n

Output: Stack of word boundary P

```
1    Initialize:  $b_1 \leftarrow 0$, $b_2 \leftarrow 0$, Stack of possible word boundary $P$,
2                 $currentBoundary \leftarrow (b_1, b_2)$
3    while $b_2 \leq n$ do
4        $currentBoundary \leftarrow (b_1, b_2)$
5        if $P$ is empty then
6            $b_2 \leftarrow$ the right index of maximumMatching($L$)
7            $currentBoundary \leftarrow (b_1, b_2)$
8            $P$.push($currentBoundary$)
9            $b_1 \leftarrow b_2$
10           $b_2 \leftarrow b_2 + 1$
11       else if matchDict($currentBoundary$) then
12           $b_2 \leftarrow b_2 + 1$
13       else
```

```
        **If** hasMatchedWord(*currentBoundary*) **then**
            $b_2$ ← right index of matched word of current boundary
            *currentBoundary* ← ($b_1$, $b_2$)
            *P*.push(*currentBoundary*)
            $b_1$ ← $b_2$ + 1
        **else**
            **while** *P* is not empty
                *p* ← *P*.pop()
                **if** a new matched word is found in *p* **then**
                    *P*.push(*newWord*)
                    $b_1$ ← the left index of *p*
                    $b_2$ ← the right index of *p*
                    *currentBoundary* ← ($b_1$, $b_2$)
                    **break**
                **end if**
            **end while**
        **end if**
    **end if**
**end while**
*P*.push(*currentBoundary*) //adding last boundary
**return** *P*
```

The main idea of Algorithm 1 is that it tracks every possible word-break position, so it is more complex than the maximal matching algorithm. The first possible word is obtained by using the maximal matching algorithm. Then, it finds the next possible word by considering the following characters, called a current boundary. The current boundary still keeps extending by the following character in the case that the word of the current boundary matches the dictionary. If the end of the string is reached, the process is done. If a character which makes the word of the current boundary mismatch the dictionary is reached, the algorithm will find the longest word which exists in the dictionary from the current boundary and marks this word as a possible word. If the word is not found from the current boundary, the algorithm will find another longest word that exists in the dictionary from all possible word boundaries in a recursive manner. If the word is found, the process will start over from that point. Note that a word mismatching the dictionary means that there is no word in the dictionary starting with that mismatched word. For example, there is no word in the dictionary starting with ‘inbl’. To illustrate the algorithm, an example of iterative segmentation of English lyrics is presented. With the lyrics “themeninblack”, the process of iterative segmentation is shown in Table A.1 (Please see Appendix A.1).

After finishing the process of Algorithm 1, a syllable sequence containing all segmented words of the lyrics is obtained. Then, each word can be searched to find its corresponding pronunciations from the dictionary, obtaining a phonetic sequence. However, there are some possible words denoted as ‘Unknown Words’ which do not have their pronunciations in the dictionary. In this case, the unknown word will be handled by the process in phase II.

Phase II is a syllabification given unknown words using the rule-based approach. There are two sub-methods of this process. The first is to segment a word into syllables using nondeterministic finite automata (NFA) based on segmenting rules and then merge some syllables together using merging rules.

The rules for segmenting syllables are listed based on the Thai writing system, an existing work [38] and observations of word patterns and their pronunciations from the Royal Institute Dictionary. The segmenting rules are listed as follows.

1. A single consonant is allowed because it can be pronounced like it is implicitly appended by ‘ะ’ (à) or ‘อ’ (ɔɔ), such as the way ‘ก’ can be pronounced as ‘กะ’ (kà) or ‘กอ’ (kɔɔ).
2. Upper and lower vowels always follow one consonant.
3. A tone mark may follow a consonant or an upper vowel or a lower vowel.
4. A trailing vowel always follows one consonant, except ‘ะ’ (a), which can follow a trailing vowel ‘า’ (aa) in order to produce a compound vowel ‘เ-าะ’ (ɔ̀).
5. A leading vowel always precedes at least one consonant.
6. The special mark ‘◌็’ (maitaikhu) and ‘◌์’ (karan) always follow one consonant.

7. The special mark ‘็’ (maitaikhu) is always followed by one consonant, except only the one case ‘ก็’ (kɔ̂).
8. For the special mark ‘็’ (maitaikhu), if there is no leading vowel and it is followed by a consonant ‘อ’ (ɔɔ), it needs one consonant after the consonant ‘อ’ (ɔɔ).
9. A syllable consisting of a leading vowel and an upper vowel needs at least one consonant after the upper vowel.
10. The special mark ‘ั’ (maihan-akat) with a tone mark needs one consonant after the tone mark.
11. A tone mark with no preceding vowel needs at least one trailing vowel or consonant after the tone mark.
12. The special mark ‘ๆ’ is always a singleton.

Figure 4.3 shows the non-deterministic finite automata (NFA) diagram of the segmenting rules. In the diagram, C is a consonant, U is an upper vowel, L is a lower vowel, R is a trailing vowel, F is a leading vowel and T is a tone mark.

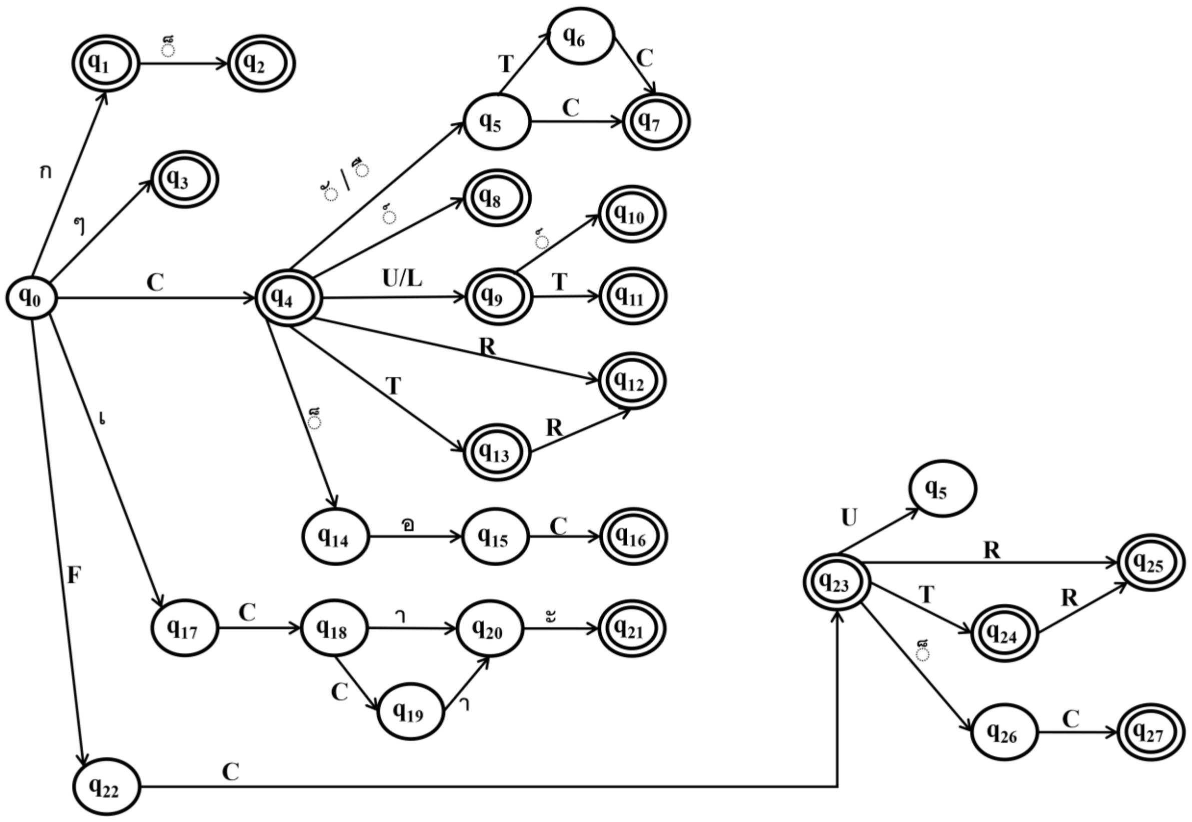


Figure 4.3 Thai segmenting NFA

After the input strings, which are the unknown words, are segmented by the NFA, obtaining possible syllables, the concatenated syllables may be merged together according to the merging rules as follows.

1. A current syllable is the character ‘ร’ and the next syllable is also the character ‘ร’ (for producing Rorhan).
2. A current syllable is ‘รร’ and the previous syllable is a single consonant.
3. A current syllable is a leading vowel and the next syllable is a single consonant.
4. A current syllable is a trailing or upper or lower vowel and the next syllable is a single consonant.
5. The last character of a current syllable is a single consonant denoted by c_1 and the first character of the next syllable is a single consonant denoted by c_2. Also, c_1 concatenated

with c_2, obtaining c_1c_2, needs to follow the rules of two consonants in the Thai language (Please see Appendix A.2).

6. A current syllable is a single consonant and the next syllable is also a single consonant.
7. If a current syllable is a single consonant, then it is merged with the previous syllable in order to produce a final consonant.
8. A current syllable contains both ‘ ื’ and ‘อ’ and the previous syllable contains ‘เ’ without ‘ ื’(for producing the compound vowel ‘เ ือ’[ɨa]).
9. If a current syllable is equal to ‘วย’, ‘อย’ or ‘ฤก’ then it is merged with the previous syllable.

After finishing the merging process in phase II, all syllables of unknown words are obtained. Then each syllable can be converted directly to phonetic symbols without a tone mark based on the Thai character – phonetic symbols mapping. (Please see Appendix A.3.) Finally, a final syllable sequence and a final phonetic sequence are obtained by combining the results from the dictionary-based approach with the rule-based approach.

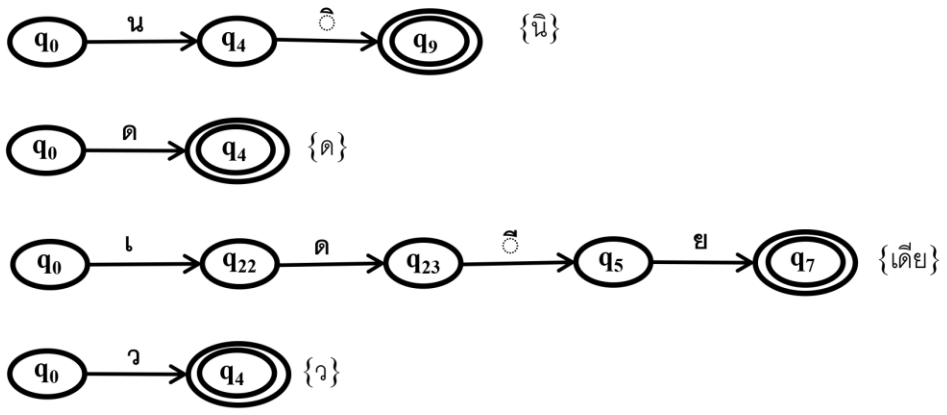


Figure 4.4 NFA that accepts Thai words 'นิดเดียว'

To illustrate the process in phase II, suppose that the given lyrics are ‘ภาษาไทยง่ายนิดเดียว’ (“The Thai language is very easy”) and after the process in phase I, the lyrics are segmented as {ภา, ษา,

ไทย, ง่าย, นิดเดียว} and the obtained phonetic sequence is {pʰaa, săa, tʰay, ŋâay, X} with an unknown word ‘นิดเดียว’. Note that ‘X’ indicates the unknown word’s position. Next, the unknown word ‘นิดเดียว’ is passed to phase II and will be accepted by the NFA as shown in Figure 4.4, obtaining possible syllables as {นิ, ด, เดีย, ว}. Then, the syllables are merged as follows. The syllables {นิ} and {ด} are merged together according to the seventh merging rule. Also, the syllables {เดีย} are merged with {ว} according to the seventh merging rule as well. Next, the segmented syllables of unknown words ‘นิดเดียว’ are {นิด, เดียว}. Then each syllable can be converted directly to phonetic symbols without tone marks as {nit, diaw}. Finally, a syllable sequence is obtained as {ภา, ษา, ไทย, ง่าย, นิด, เดียว} and a phonetic sequence is obtained as {pʰaa, săa, tʰay, ŋâay, nit, diaw}. Then a tone sequence will be obtained by the tone analysis process in phase II, as shown in Figure 4.1, which will be described in the following section.

4.3.4 Thai Tone Analysis

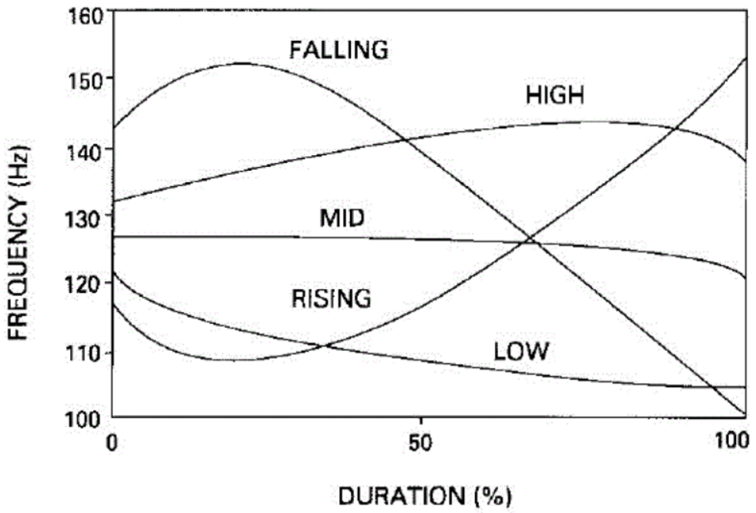


Figure 4.5 Five tones of the Thai language

As mentioned, Thai has five tones, mid, low, falling, high and rising, as shown in Figure 4.5. For the dictionary-based approach, the tone of each syllable can be easily extracted from the tone mark in the phonetic symbols provided by the dictionary. Table 4.2 shows the tone marks of five tones.

Tone	**Tone mark in Thai character**	**Tone mark in phonetic symbols**
Mid	No tone mark	No tone mark
Low	◌่	ˋ
Falling	◌้	ˆ
High	◌๊	ˊ
Rising	◌๋	ˇ

Table 4.2 Thai tone marks

However, for the rule-based approach, all syllables of unknown words need to be obtained first and then the tones analyzed according to the Thai tone rules. (Please see in Appendix A.4.)

Let an integer represent a tone as follows: 0 represents the mid tone, 1 represents the low tone, 2 represents the falling tone, 3 represents the high tone and 4 represents the rising tone.

From the previous example, the phonetic sequence is obtained as {p^{h}aa, sǎa, t^{h}ay, ŋâay, nit, diaw}. The first four syllables are obtained from the dictionary-based approach so their tones can be directly extracted from the phonetic symbols using the mapping shown in Table 4.2. The sequence of the first four tones is {0, 4, 0, 2}. However, the remaining two syllables are obtained from the rule-based approach, so their tones can be obtained by considering the Thai tone rules shown in Appendix A.4 given the Thai segmented syllables {นิด, เดียว}. For the syllable 'นิด', its initial consonant is 'น', which is in the low class, there is no tone mark and the syllable is closed with a dead final consonant (ด) and has a short vowel (◌ิ), so its tone is a high tone. For the syllable 'เดียว', its initial consonant is 'ด', which is in the mid class, there is no tone marks and the syllable is open and has a long vowel (เ◌ียว), so its tone is a mid tone. Thus, the sequence of the two remaining tones is {3, 0}. Finally, the tone sequence of the lyrics ‘ภาษาไทยง่ายนิดเดียว’ is {0, 4, 0, 2, 3, 0} and the phonetic sequence can be modified by adding tone marks as {p^{h}aa, săa, t^{h}ay, ŋâay, nít, diaw}.

4.3.5 Automatic Alignment

Automatic alignment for Thai is a method to align each word in an entry with its corresponding phonetic symbols. The alignment method is based on the mapping between Thai characters and phonetic symbols. For the previous example, the obtained syllable sequence is {ภา, ษา, ไทย, ง่าย, นิด, เดียว} and the obtained phonetic sequence is {p^{h}aa, săa, t^{h}ay, ŋâay, nít, diaw}. A sequence of alignments is obtained as {(ภา, p^{h}aa), (ษา, săa), (ไทย, t^{h}ay), (ง่าย, ŋâay), (นิด, nít), (เดียว, diaw)}.

4.4 Lyrics Processing for Japanese

4.4.1 Japanese Orthography

In the Japanese writing system, a character refers to a grapheme which denotes all or part of a word. There are three types of characters: kanji, hiragana and katakana. Kanji are ideographic characters adopted from the Chinese writing system, while Hiragana and katakana are both kana systems, which are Japanese syllabaries. Hiragana are used to write grammatical inflections and native words, for which there are no kanji, and Katakana are used for transcription of foreign language words into Japanese. Figure 4.6 shows an example of Japanese orthography for the sentence '私はバンコクに行きました' meaning 'I went to Bangkok'. The word '私' (watashi) is kanji, meaning 'I'. The words 'は' (wa) and 'に' (ni), which are binding particles, are hiragana. And the word 'バンコク' (Bankoku) is katakana, presenting the name of a foreign city, 'Bangkok'. The verb '行きました' (ikimashita) meaning 'went' contains a mixture between a kanji character indicating a verb and hiragana characters representing the verb conjugation for the past tense.

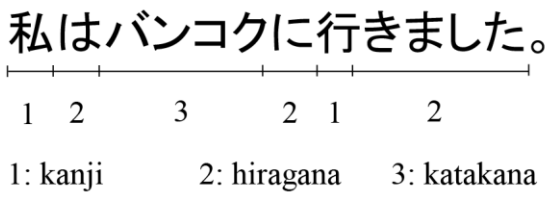


Figure 4.6 An example of Japanese orthography

Japanese written text can be transliterated into kana characters (in this thesis hiragana are used) without loss of information about its meaning, as shown in Figure 4.7. However, there is no white space delimiting words in the Japanese writing system, and only sentence boundaries are delimited. Therefore, the text must be segmented prior to converting each group of graphemes to their meaning or pronunciation. Figure 4.7 also shows the correct word segmentation of the sentence.

私	は	バンコク	に	行きました。
わたし	は	ばんこく	に	いきました。

Figure 4.7 Transliteration of Japanese words into hiragana

4.4.2 Overview of Lyrics Processing for Japanese

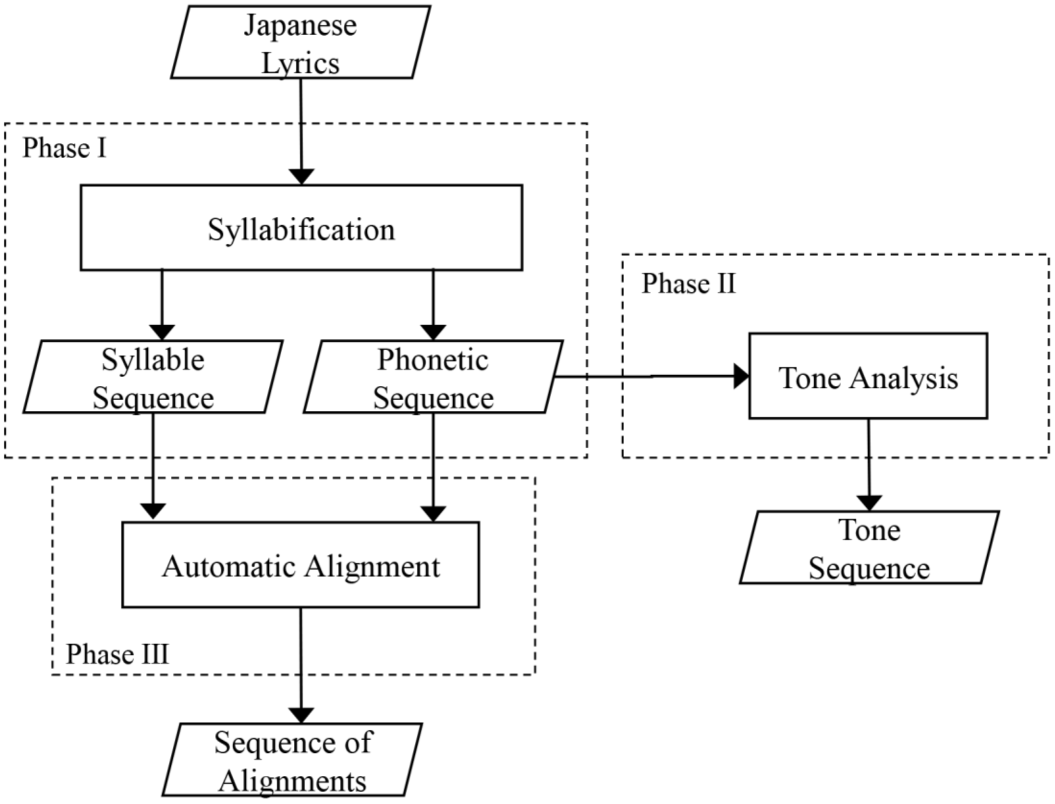


Figure 4.8 Overview of lyrics processing for Japanese

Figure 4.8 shows lyrics processing for Japanese lyrics. The process is quite similar to the lyrics processing for the Thai language. The process is composed of three phases. The tone extraction process is in phase I and phase II, which will be described first. The last phase is the automatic alignment, which is described in Section 4.4.5.

The first phase is a syllabification of Japanese lyrics, obtaining a syllable sequence and a phonetic sequence. Next, all syllables in the obtained syllable sequence are analyzed to find the corresponding tones, obtaining a tone sequence in phase II. The syllabification is described in the next section.

4.4.3 Japanese Syllabification

To syllabify Japanese text, the text must be segmented into words prior to converting them to the corresponding pronunciations. Given Japanese lyrics and a dictionary containing word frequencies, the lyrics are segmented into words by using dynamic programming proposed in [40]. The main idea is to find the set of word boundaries considered most likely based on the frequency of the segmented as shown in Algorithm 2. Note that *word* (*i*,*j*) is a substring of the lyrics *L* from index *i* to *j*. The scoreFunction is a function for finding the score of a specific word based on the word frequency in the dictionary, and a lower score is better in this algorithm. Also, the word matching the dictionary means that there is at least one word in the dictionary starting with that word.

After finishing the process of Algorithm 2, a syllable sequence containing all segmented words of the lyrics is obtained. Then, each word can be searched to find its corresponding pronunciations from the dictionary, obtaining a phonetic sequence containing all syllables of pronunciation with tone information. To illustrate, given the lyrics '私はバンコクに行きました', the obtained syllable sequence from Algorithm 2 is {私, は, バンコク, に, 行き, ま, した}. Then all segmented words are searched for their corresponding pronunciations with tone information from the dictionary. The corresponding pronunciations of each word are {{wa, ta, shi}, {wa}, {ban$^{\downarrow}$, ko, ku}, {ni}, {i, ki}, {ma}, {shi, ta}}. Finally, a phonetic sequence is obtained as {wa, ta, shi, wa, ban$^{\downarrow}$, ko, ku, ni, i, ki, ma, shi, ta}. (Note that the symbol '$^{\downarrow}$' denotes a high-toned syllable.) Then a tone sequence will be obtained by the tone analysis process in phase II of Figure 4.8, which will be described in the following section.

Algorithm 2 Japanese Word Segmentation

```
Input:   Lyrics $L$ with length $n$
Output:  Array of index of word segmentation $prev$
1  Initialize:  endBefore array with size $n+1$ and each element of array is set to be the
2  max integer value, prev array with size $n+1$ and each element of array is set to be –1.
3  $endBefore[0] \leftarrow 0$
4  for each index $i$ of character of $L$ do
5      for each index $j$ of character of $L$ do
6          if $j \geq i$ and $word(i,j)$ matches the dictionary then
7              $score \leftarrow endBefore[i] + \text{scoreFunction}(word(i,j))$
8              if $score < endBefore[j]$ do
9                  $endBefore[j] \leftarrow score$
10                 $prev[j] \leftarrow i$
11             end if
12         end if
13     end for
14 end for
15 return $prev$
```

4.4.4 Japanese Tone Analysis

In Japanese, there are three tones, a high tone, a low tone and a non-tone, depending on the accent. Words with the same pronunciation but different tones produce words with different meanings. The tone of Japanese lyrics can be extracted directly from the corresponding pronunciations obtained from the dictionary.

Let an integer represent a tone as follows: 0 represents the low tone and1 represents the high tone. For the previous example, given the lyrics '私はバンコクに行きました', the phonetic sequence is obtained as {wa, ta, shi, wa, ban🠗, ko, ku, ni, i, ki, ma, shi, ta}. Thus, a tone sequence is obtained as {0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0}.

4.4.5 Automatic Alignment

Automatic alignment for Japanese is a method to align each kanji character in an entry with its corresponding hiragana. The alignment algorithm based on pattern matching and linguistic constraints is proposed as Algorithm 3. The linguistic constraints are listed as follows.

1. The kana character must align directly with the corresponding phoneme of that kana character.
2. Intra-syllabic segments cannot exist for kana strings. The segmentation is based on the syllable rather than the character level. Some single kana characters must be combined with the preceding kana character. For example, for the ka-n combination, a segment between ka and n is disallowed.

The main idea of Algorithm 3 is as follows. First, generating all possible cases of phoneme segmentation from $P(L)$ assigned to a single character, the number of possible cases is C_{n-1}^{m-1}, where *n* is the length of the syllable sequence and *m* is the length of the phonetic sequence. Next, for each case, calculating the score of each single character by comparing the assigned phoneme to every assigned phoneme of the same character from the words in the dictionary, there are two scores, *matchScore* and *containScore.* Both scores are initialized to some values so that the *matchScore* is greater than the *containScore.* For the score calculation of each character, if both the phoneme and character of one case and the dictionary are exactly the same as each other, the score is increased by the *matchScore.* However, if the word in the dictionary contains the character of the case and the phoneme of the dictionary contains the phoneme of the phoneme of the case, the score is increased by the *containScore.* Next, the score of each single character is summed to be a total score for each case, and then the total score is adjusted according to the linguistic constraints. Finally, the case which has the greatest score is selected to be the proper segmentation. Please see Appendix A.5 for the illustration of Algorithm 3.

Algorithm 3 Automatic Alignment for Japanese

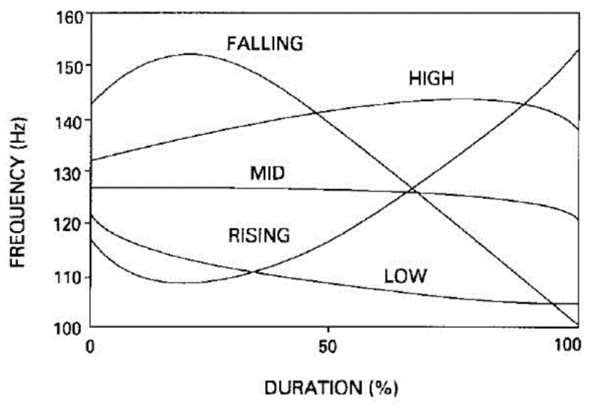
```
Input:    A syllable sequence $S(L) = \{l_1, l_2, l_3, \dots, l_n\}$ and
          a corresponding phonetic sequence $P(L) = \{p_1, p_2, p_3, \dots, p_m\}$
Output:   A sequence of alignments $A(L) = \{(l_1, r_1), (l_2, r_2), \dots, (l_n, r_n)\}$
1   Initialize:   $C \leftarrow \emptyset$,
2                 matchScore, containScore where matchScore > containScore,
3                 Generating all possible cases of phoneme segmentation $C$ from $P(L)$.
4                 score array containing scores for every possible cases $C$ initialized to 0.
5   for each case $c_k$ in $C$ do
6       for each syllable $l_i$ do
7           $r_i$ is a corresponding phoneme of $l_i$
8           for each tuple $t_j$ in the dictionary do
9               $w_j$ is a word of $t_j$
10              $y_j$ is a phonetic sequence of $t_j$
11              if $w_j$ equals to $l_i$ and $y_j$ equals to $r_i$
12                  score[$c_k$] $\leftarrow$ score[$c_k$] + matchCost
13              else if $w_j$ contains $l_i$ and $y_j$ contains $r_i$
14                  score[$c_k$] $\leftarrow$ score[$c_k$] + containCost
15              end if
16          end for
17      end for
18      Adjust the score[$c_k$] according to the linguistic constraints.
19  end for
20  return $c_k$ which has the greatest score.
```

4.5 Tonal Transformation

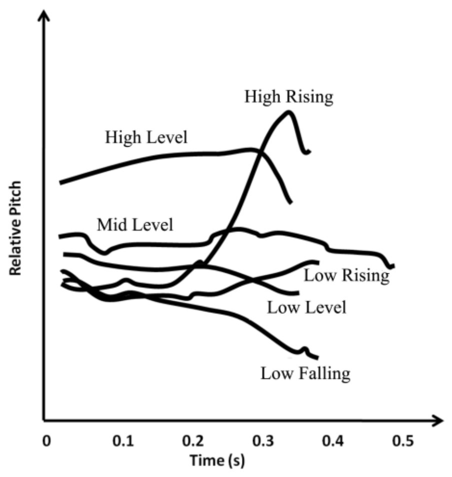
To compose a melody for Thai and Japanese lyrics with the T-Music system, this thesis uses a probabilistic automaton constructed from Cantonese songs. The reason is that all tones in Thai and Japanese can be mapped to Cantonese tones. Thus, all tones in the obtained tone sequence from the lyrics processing need to be transformed into tones of Cantonese. The tonal transformation of the two languages is described as follows.

4.5.1 Tonal Transformation for Thai

Recall that Thai has five tones, which are mid, low, falling, high and rising tones. In Cantonese, there are six tones, which are high level, high rising, mid-level, low rising, low level and low falling tones. Figure 4.9 shows the fundamental frequency contours for the Thai and Cantonese tones.



(a)



(b)

Figure 4.9 Fundamental frequency contours representing (a) Thai and (b) Cantonese tones

Based on the fundamental frequency representing the tones of both languages, each Thai tone is transformed to the most similar Cantonese tone, as shown in Table 4.3.

Thai tone	**Cantonese tone**
mid	mid-level
low	low falling
falling	low level
high	high level
rising	high rising

Table 4.3 Transformation from Thai tones to Cantonese tones

4.5.2 Tonal Transformation for Japanese

Cantonese tones can be divided into two groups, high and low tones. The high-tone group is comprised of high level, high rising and mid-level tones, and the low-tone group is comprised of low rising, low level and low falling tones. Recall that Japanese has two tones, which are high and low tones. Therefore, the two tones of Japanese can be easily transformed to these two groups of Cantonese tones. The high tone of Japanese is transformed to the Cantonese tone by randomly choosing one tone from the high-tone group. Similarly, for the low tone, the transformation is done by randomly choosing one tone from the low-tone group.

CHAPTER 5
STATISTICAL MUSIC ACCOMPANIMENT GENERATOR

After creating a melody for a song, music accompaniment is developed to provide the harmonic background and the rhythmic structure for the song. Figure 5.1 shows the structure of a song composed of a melody line and an accompaniment line. The accompaniment is usually provided by one or more instruments, which are used to play chords, such as a piano, organ, acoustic guitar or electronic keyboard.

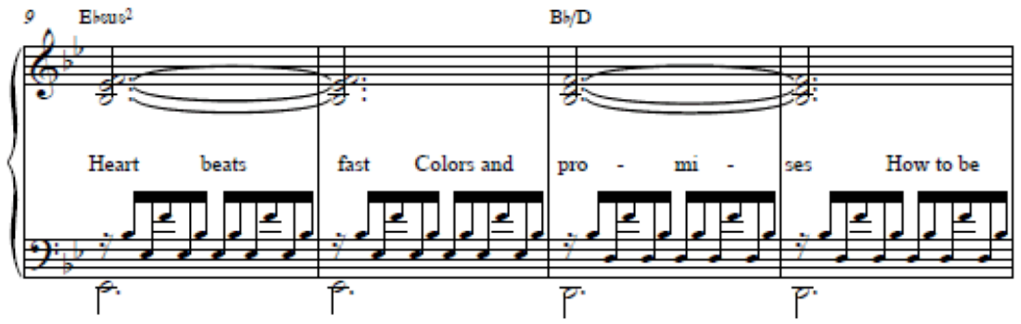


Figure 5.1 Structure of a song

Accompaniment composition is highly dependent on the coordination between the accompaniment part and the melody part of a song, which is known as harmony in music theory. To compose well-coordinated accompaniment, suitable chords which match the melody need to accompany the melody.

To ensure that readers of varying musical backgrounds can follow this thesis, some terms of music theory which are necessary to understand throughout this thesis are described in the following section.

5.1 Music Terminology

A *key* identifies a tonic note which is the starting note of the key and chord. It can be inferred from the key signature, which can be drawn as major or minor. Besides this, in every key, there are seven fundamental notes, which are known as a scale in music. Basically, there are totally seven chords which can be chosen to generate the accompaniment based on a key.

Figure 5.2 shows the C major scale, which consists of the notes C, D, E, F, G, A and B. Based on the C major scale, there are seven triad chords which are indicated by roman numerals in Figure 5.3.

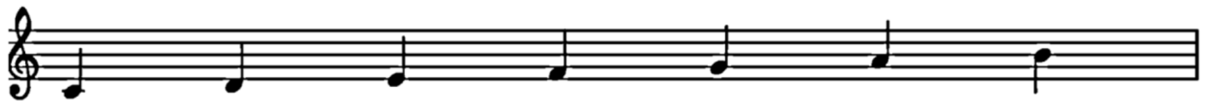


Figure 5.2 C major scale

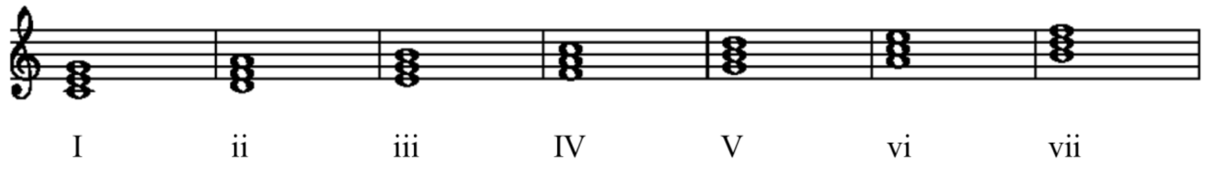


Figure 5.3 Seven triad chords based on C major key

A *phrase* is a group of consecutive melodic notes, which is similar to a sentence in a passage. In one song, there are several phrases. A phrase is broken down into bars depending on the beat of the song specified by the composer. The number of beats in each bar (or measure) is specified by a time signature, such as 3/4. Inside a bar, different notes can be found.

A *chord* is a harmonic unit with at least three different notes [41]. The most basic chord is a triad, which consists of three distinct notes called the first (root), third and fifth note. Chords are commonly classed by their root note. A chord will change to another chord from time to time throughout the song, depending on the note in the melody. This change is known as *chord progression* in music theory. There is no certain pattern of chord progression. However, different chord progressions can lead to different styles and moods of a song. Generally, it is not hard for humans to notice that certain chords are always preferable and sound more natural when they comes after a specific chord. Based on the preferences of chords, there are some correlations between the different chords. Therefore, chord progressions can be predicted for a new melody by studying the correlations [41, 42].

5.2 Problem Statement

Accompaniment generation is a problem of finding corresponding chords for a given melody. The problem can be formulated as follows.

Given a melody M, a *melody sequence* divided into segments using a bar as a unit can be defined as

$$M = \{m_1, m_2, m_3, \dots, m_b\}, \tag{5}$$

where b is a number of bars of M.

For each melody in a bar m_i, where $i \in [1, b]$, there is a corresponding chord. A *chord sequence* of the melody M can be defined as

$$C(M) = \{c_1, c_2, c_3, \dots, c_b\}, \tag{6}$$

where c_i is a chord corresponding to m_i.

5.3 System Overview

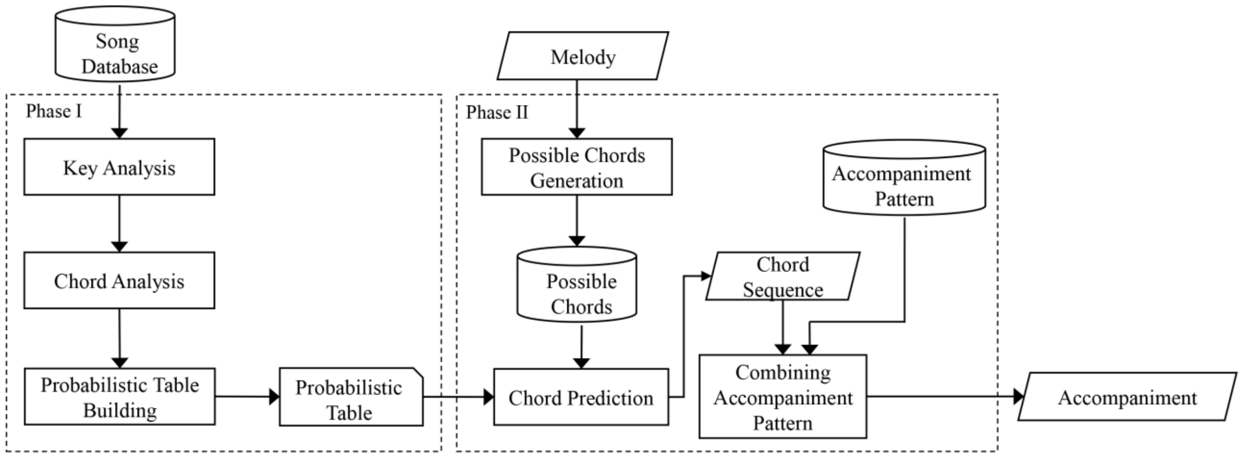


Figure 5.4 Architecture of statistical music accompaniment generator

The statistical music accompaniment generator is a system based on a probabilistic model that automatically generates appropriate accompaniment with different rhythmic patterns for a given melody. The architecture of the system is presented in Figure 5.4. The system is comprised of two phases. The first phase is probabilistic table generation, and the second phase is accompaniment generation.

5.4 Probabilistic Table Generation

This section introduces the first phase of the system. The main idea is that each song in the database is analyzed to find the most frequent pattern of successive chords to generate the probabilistic table.

First, some definitions are introduced. A *note* is represented by a *note position* measured by the note's distance from a tonic note with a semitone as the unit, regardless of which octave the note belongs to. For example, in Figure 5.5, suppose that the key is C major. The tonic note is the note C, which is the zeroth position, and the note F# is the sixth position.

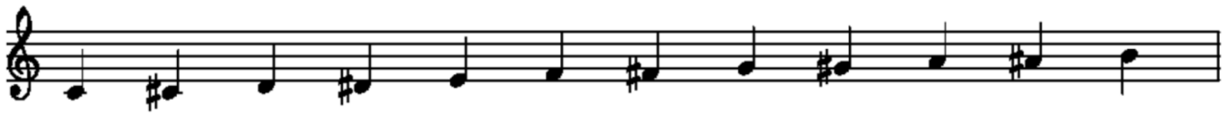


Figure 5.5 All notes in an octave

The frequency of each note position existing in a song is denoted as $f_0, f_1, f_2, \dots, f_{11}$, called *note-frequency tabulation*. The note frequency of a song with multiple tracks can be denoted as a frequency array called a *note-array*. Each element is denoted as $\{f_0, f_1, f_2, \dots, f_{11}\}$.

KEY_TABLE is defined as a 2D-array of the note-frequency tabulation used for representing some known scales and chords. If a note exists in the scale of a key, the frequency of the note is assigned to be 1; otherwise 0 is assigned. The *KEY_TABLE* of 12 keys is presented in Table 5.1. Note that '+' denotes a major key and '-' denotes a minor key.

Key	Element
C+/A-	{1,0,1,0,1,1,0,1,0,1,0,1},
C#+/A#-	{1,1,0,1,0,1,1,0,1,0,1,0},
D+/B-	{0,1,1,0,1,0,1,1,0,1,0,1},
D#+/C-	{1,0,1,1,0,1,0,1,1,0,1,0},
E+/C#-	{0,1,0,1,1,0,1,0,1,1,0,1},
F+/D-	{1,0,1,0,1,1,0,1,0,1,1,0},
F#+/D#-	{0,1,0,1,0,1,1,0,1,0,1,1},
G+/E-	{1,0,1,0,1,0,1,1,0,1,0,1},
G#+/F-	{1,1,0,1,0,1,0,1,1,0,1,0},
A+/F#-	{0,1,1,0,1,0,1,0,1,1,0,1},
A#+/G-	{1,0,1,1,0,1,0,1,0,1,1,0},
B+/G#-	{0,1,0,1,1,0,1,0,1,0,1,1}

Table 5.1 KEY_TABLE

A *CHORD_TABLE* is also defined as a 2D-array for seven triad chords. According to music theory, the root note and the third note of a triad are more important than the fifth note. So, the two important notes are weighted with a higher weight by assigning the root note and the third note as ‘0.4’, the fifth note as ‘0.2’, and else as ‘0’. The *CHORD_TABLE* is presented in Table 5.2.

Chord in roman numeral	Element
I	{0.4,0,0,0,0.4,0,0,0.2,0,0,0,0},
ii	{0,0,0.4,0,0,0.4,0,0,0,0.2,0,0},
iii	{0,0,0,0,0.4,0,0,0.4,0,0,0,0.2},
IV	{0.2,0,0,0,0,0.4,0,0,0,0.4,0,0},
V	{0,0,0.2,0,0,0,0,0.4,0,0,0,0.4},
vi	{0.4,0,0,0,0.2,0,0,0,0,0.4,0,0},
vii	{0,0,0.4,0,0,0.2,0,0,0,0,0,0.4}

Table 5.2 CHORD_TABLE

5.4.1 Key Analysis

Key analysis is a process to determine the key of a song. The process is as follows. Let F be a note-array of a song in the database. The mean $\bar{x}$ and the standard deviation σ of the frequency of notes, which exists in F, are calculated. The threshold is set to be $\bar{x} - \sigma$. Let F' be a modified note-array. For each element in F, if it is greater than the threshold, that element in F' is assigned to be '1', else it is assigned to be '0'. Then comparing the similarity between F' and each key k in the *KEY_TABLE* by calculating the Euclidean distance d_k as

$$d_k = \sqrt{\sum_{i=0}^{11} (F'[i] - KEY\_TABLE[k][i])^2}\,, \qquad (7)$$

where $k \in [0,11]$ and $i \in [0,11]$, then the key of the kth position in the *KEY_TABLE* is assigned to the song, whose d_k is minimum.

5.4.2 Chord Analysis

Chord analysis is a process to find suitable chord progressions for a given melody. This task is challenging because chords may change once a bar, twice a bar, or even once every two bars, depending on the notes in the melody. In this system, the chords are analyzed every two beats or three beats depending on the time signature of the song. If the time signature is 4/4, 2/4, 2/2, 4/2 or 12/8, the chords are analyzed every two beats; otherwise the chords are analyzed every bar. To analyze the chord progressions, first, the whole song is divided into bars. Then analyzing the beat, if the beat is divisible by two, the bar is subdivided into two equal segments; otherwise the bar is kept as a segment. Let s denote a segment and S denote a song in terms of a series of segments. The note appearing in a segment is represented as the note-frequency tabulation mentioned before, $s = \{f_0, f_1, f_2, \dots, f_{11}\}$. A song with n segments is represented as a 2D-array as $S = \{\{f_0, f_1, f_2, \dots, f_{11}\}_1, \{f_0, f_1, f_2, \dots, f_{11}\}_2, \dots, \{f_0, f_1, f_2, \dots, f_{11}\}_n\}$. For each s, the mean $\bar{x}$ and the standard deviation σ of the frequency of notes which exist in s are calculated. The threshold is set to be $\bar{x} - 2\sigma$. Let s' be a modified note-array. If each element in s is greater than the threshold, that element in s' is assigned to be '1', and else is assigned to be '0'. Also, the root note, the third note and the fifth note are weighted. The root note and the third note are multiplied by 0.4 and the fifth note is multiplied by 0.2. Then for each s', the similarity between the vector s' and each chord k in the *CHORD_TABLE* is compared by calculating the Euclidean distance d_k as

$$d_k = \sqrt{\sum_{i=0}^{11}(s'[i] - CHORD\_TABLE[k][i])^2} \tag{8}$$

where $k \in [0, 6]$ and $i \in [0,11]$. Then the chord of the kth position in the *CHORD_TABLE*, defined as c_k, is assigned to the segment whose d_k is minimum. If the analyzed chord is different from the chord of the successive segment, it means that there is a chord progression; otherwise there is no chord progression.

Moreover, there is an issue of tonality. To determine the tonality, the chord of the last bar is examined. If the song ends at chord vi of the major key, the song is regarded as the relative minor of the major. Then, the built chord from the major key is mapped to the corresponding one of the minor key. For example, chord I in C major will map to chord III in A minor and chord vi in C major will map to chord I in A minor.

5.4.3 Probabilistic Table Building

A progression from chord *A* to chord *B* is denoted as $A \rightarrow B$, where *A* and *B* are two different chords. For each song in the database, a chord sequence can be obtained from the chord analysis as $\{c_1, c_2, c_3, \dots, c_n\}$. Let *k* be the position of segment in the *S*, and c_k denotes the chord adopted for s_k. The chord progression which takes place between s_{k-1} and s_k is denoted as $c_{k-1} \rightarrow c_k$. Let '*' be any of the seven chords. For each song, there are *p* occurrences of $c_{k-1} \rightarrow c_k$ so that an occurrence is denoted as $Pc_{k-1} \rightarrow c_k$ and there are q occurrences of $* \rightarrow *$. Therefore the normalized value of $c_{k-1} \rightarrow c_k$ is defined as $\frac{p}{q}$. Similarly, the song has the normalized value of $c_{k-1} \rightarrow *$. Next, the occurrences of the normalized values of different progressions are arranged in a table *t*, as shown in Table 5.3. Also, the normalized value of $c_{k-1} \rightarrow *$ is stored in an array *a*. Algorithm 4 shows the calculation of the normalized value of each song.

chord	I	ii	iii	IV	V	vi	vii
I	*0,*	$\frac{p_{I\to ii}}{q}$,	$\frac{p_{I\to iii}}{q}$,	…	…	…	…
ii	…	*0,*	…	…	…	…	…
iii	…	…	*0,*	…	…	…	…
IV	…	…	…	*0,*	…	…	…
V	…	…	…	…	*0,*	…	…
vi	…	…	…	…	…	*0,*	…
vii	…	…	…	…	…	…	*0*

Table 5.3 Structure of table *t*

Algorithm 4 Calculating normalized value of each song

Input: A song in format of *S*

Output: Table *t*, Array *a*

```
1  Initialize:  $t \leftarrow \emptyset$,
2               $a \leftarrow \emptyset$
3  for each segment $s_k$ of $S$ do
4      $prev \leftarrow c_{k-1}$
5      $curr \leftarrow c_k$
6      if $prev \neq curr$ then
7          $q \leftarrow q + 1$
8          $a_{prev} \leftarrow a_{prev} + 1$
9          $t_{prev,curr} \leftarrow t_{prev,curr} + 1$
10     end if
11 end for
12 for each $i = 0$ to 6 and $j = 0$ to 6 do
13     $t_{i,j} \leftarrow t_{i,j}/q$
14     $a_i \leftarrow a_i/q$
15 end for
16 return $t$, $a$
```

Suppose there are *m* songs in the database. Two chords are represented as *u* and *v*. The normalized probability of $u \rightarrow v$ is denoted as $P(u \rightarrow v)$. It is equal to the sum of the normalized values of $u \rightarrow v$ of all songs divided by the sum of the normalized values of $u \rightarrow *$. The element in the *u*th row and *v*th column in table t_i is denoted as $t_i(u, v)$, and the element in the *u*th position in array a_i is denoted as $a_i(u)$. The normalized probability $P(u \rightarrow v)$ is obtained as the following equation:

$$P(u \rightarrow v) = \frac{\sum_{i=0}^{m} t_i(u, v)}{\sum_{i=0}^{m} a_i(u)}. \qquad (9)$$

Finally, the normalized probability calculated from all songs in the database is arranged the same as the table *t* denoted as a probabilistic table *T*. Two probabilistic tables are generated, one built from major songs and another from minor songs. The major probabilistic table is named *T*+, and the minor probabilistic table is named *T*-.

However, there is an issue that the probabilistic table *T* does not allow two same successive chords to exist since the probability is 0. However, some songs will use the same chord continuously in order to create a calm and peaceful atmosphere. Therefore, the probabilities of the same chord progressions are modified using the similarity of (α, β, γ) and generated as discussed in Section 5.5.1. Note that this system restricts the same chord progression in the verse and for the chords I and V. The similarity and the weighted similarity are denoted as s_α and w_α respectively. The similarity is the Euclidean distance of the note in the melody and the actual chord. The weighted similarity can be obtained as the following equation:

$$w_\alpha = \frac{s_\alpha}{s_\alpha + s_\beta + s_\gamma}. \qquad (10)$$

The unknown probability of the same chord progression is denoted by *X* and can be obtained by examining the proportion of the similarity of the three chords. The same chord progression is assumed to occur in α. The probability of the same chord progression can be obtained as the following equation:

$$\frac{X}{w_\alpha} = \frac{P(\beta \rightarrow \alpha) + P(\gamma \rightarrow \alpha)}{w_\beta + w_\gamma}. \qquad (11)$$

5.5 Accompaniment Generation

Accompaniment Generation is the second phase of the system, which automatically generates the accompaniment according to a given melody. The main idea is that the given main melody is divided into segments and then the corresponding chord of each segment is predicted. Finally, each chord is combined with different accompaniment patterns to compose a pleasing song.

5.5.1 Possible Chords Generation

Given a main melody *M*, the melody is divided into segments using a bar as a unit. Then possible chords are found using the chord analyzing method described in Section 5.4.2. After considering the Euclidian distances, the three most similar chords are adopted. Let Q denote a tuple of chords generated from a bar ordered by the similarity in descending order. Elements in the tuple are denoted as $Q = (\alpha, \beta, \gamma)$, where α is the most similar chord, β is the second most similar chord and γ is the third most similar chord.

5.5.2 Chord Prediction

According to music theory, phrases are restricted to end in several patterns of chord progressions, which is known as cadence in music. Therefore, the chord prediction cycle is performed from the last bar to the first bar in every phrase in this system. If there is no phrase information given, it is assumed that a phrase ends at every four bars. If the given phrase is less than four bars, called a sub-phrase, it is combined with the next phrase to form a complete phrase.

Let *P* denote a phrase. Given a melody M with *n* phrases and *m* bars in a phrase, a phrase is represented in a form of the chord tuples as $P = \{(\alpha_1, \beta_1, \gamma_1), (\alpha_2, \beta_2, \gamma_2), \ldots, (\alpha_m, \beta_m, \gamma_m)\}$. For each phrase, the bar *m* is processed first by considering its chord tuple and cadence. Then, the bar *m*-1, *m*-2 continues to be processed until it reaches the first bar. The idea is illustrated in Figure 5.6. The possible ending chords restricted by cadence are stored in a set *L*, and set *K* stores the set of chords before the specific ending chord. The method *f(L)* is to find the first occurrence of the possible ending chord that exists in set *L* in order based on the tuple (α, β, γ). Suppose that *L* is {I, V, vii} and a given tuple is (ii, I, V). Chord I will be adopted. Next, the method *f(K)* is to find the chord before the specific ending chord using a similar method to *f(L)* by considering set *K*. Method *g(T)* finds the chord by looking up the probabilistic table *T*+ or *T*-. Let *k* denote the index of the bar from the end so c_{k+1}will be the chord adopted in the next bar. For tuple $(\alpha_k, \beta_k, \gamma_k)$, it looks up the probability of the element in the tuple progressing to the next bar as shown below:

$$P(\alpha_k \rightarrow c_{k+1}) = T[\alpha_k][c_{k+1}] \tag{12}$$

$$P(\beta_k \rightarrow c_{k+1}) = T[\beta_k][c_{k+1}] \tag{13}$$

$$P(\gamma_k \rightarrow c_{k+1}) = T[\gamma_k][c_{k+1}]. \tag{14}$$

Then, based on the normalized probability, a random number between 0 and 1 is generated and the corresponding chord adopted. The progress continues until it reaches the first bar of the phrase. Finally, a chord sequence *C*(*M*) containing the corresponding chords of all bars of the melody is obtained.

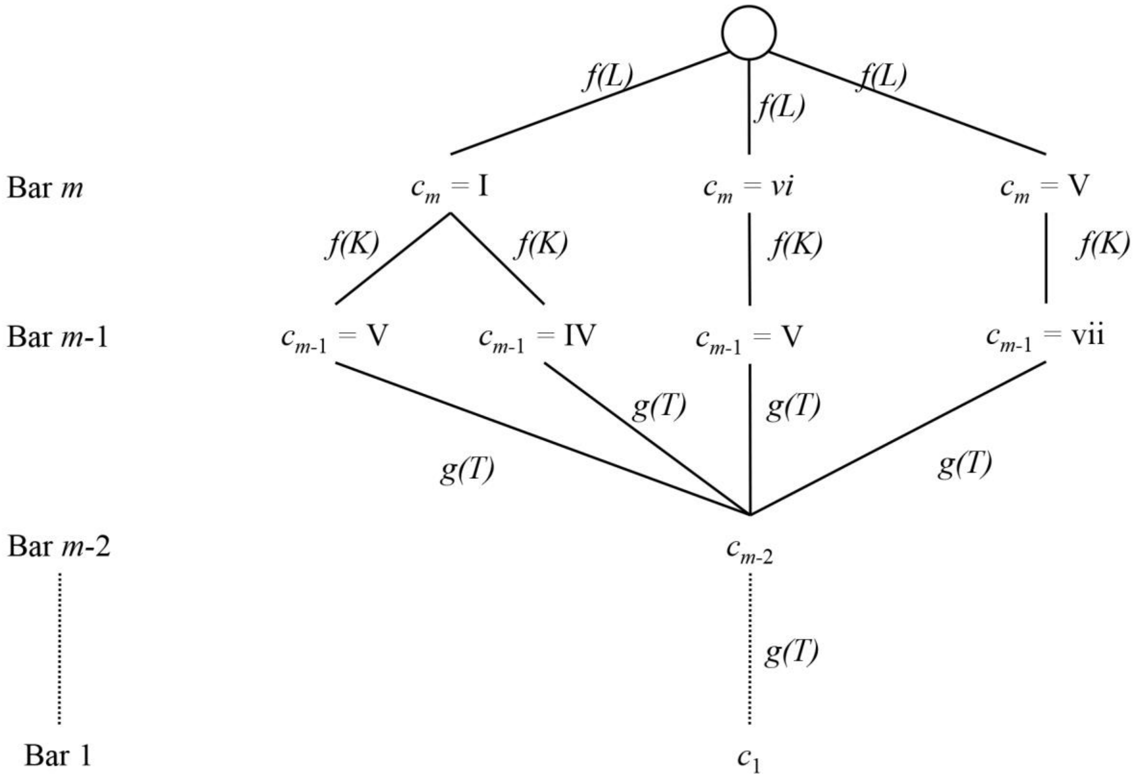


Figure 5.6 Flow chart of chord prediction

5.5.3 Combining Accompaniment Patterns

Throughout a song, there are different rhythmic accompaniment patterns in different parts to create different senses of depth. Figure 5.7 illustrates two different accompaniment patterns.

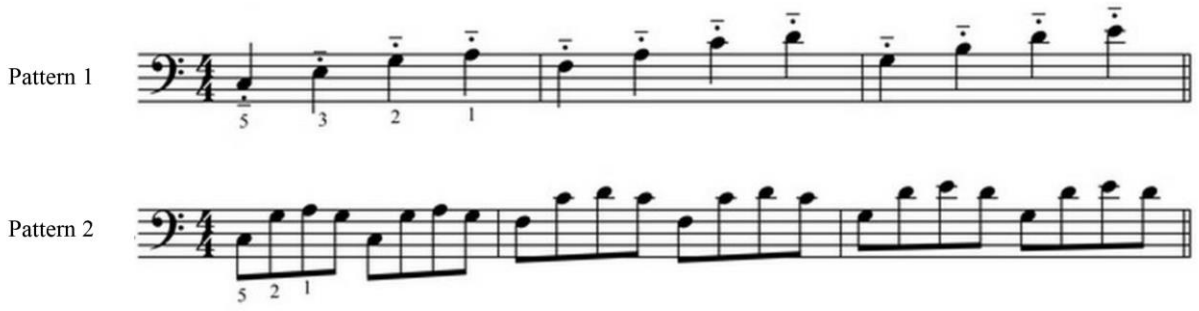


Figure 5.7 Examples of two different accompaniment patterns

To combine the accompaniment patterns with the obtained chord sequence *C*, the first process is that the chord sequence is separated into different parts, which are the chorus and verse, depending on the duration of the melody. The chorus is defined as the part of the melody starting from ½ to ¾ of the duration and the rest is verse. Then, given accompaniment patterns for different time signatures, for each chord c_i in *C*, a pattern for the chorus and another pattern for the verse are randomly chosen. Since the chorus is the climax of a song, the accompaniment should be richer and more complicated. Therefore, the pattern chosen for the chorus will contain more notes than the one for the verse, as shown in Figure 5.8 and Figure 5.9. Finally, for the ending of the song, one more bar containing a chord pattern based on the chord of the last bar of the last phrase is added in order to give a proper ending to the song.

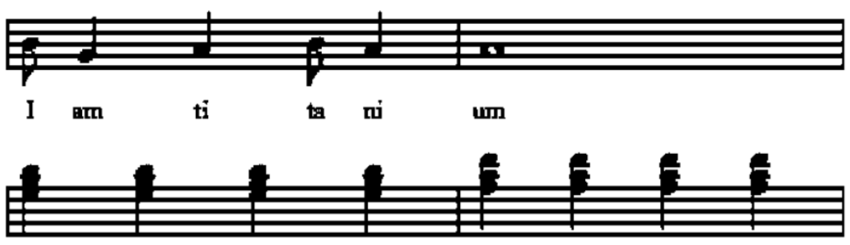


Figure 5.8 An example of an accompaniment pattern for a chorus

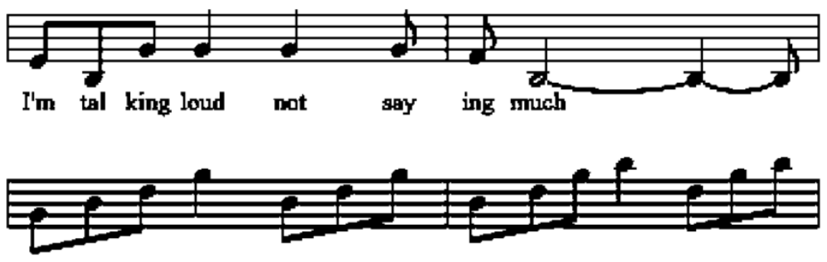


Figure 5.9 An example of an accompaniment pattern for a verse

CHAPTER 6
EXPERIMENTS AND EVALUATIONS

Two experiments were conducted to evaluate the effectiveness of the two systems, one for composing melody for different languages and the other a statistical music accompaniment generator.

6.1 Composing Melody for Different Languages

The first experiment was to evaluate the melody composer for five languages: English, Cantonese, Mandarin, Thai and Japanese. To achieve the evaluation, a survey was designed and conducted, aiming to generate objective feedback on aesthetic quality.

6.1.1 Melody Preparation

To obtain an accurate measure of a subjective rating is very difficult. To address this issue, a small number of melodies were generated because this allowed a greater number of different evaluations for each melody. For each language, ten melodies were prepared based on five lyrics. Five melodies were based on the original melodies and another five songs were generated by the T-Music system and given the same lyrics as their original versions. All melodies ranged from thirty seconds to one minute in length. Each melody generated by the T-Music system was set to have the same key signature, time signature, duration and lyrics as its original version. For the Cantonese, Thai and Japanese lyrics, the generated melodies were based on the Cantonese song database containing 241 pop songs but the generated melody was based on the 130 Mandarin pop songs in the database for Mandarin lyrics. Also, each melody was fixed to use a piano sound in order to present a fair comparison between the generated version and the original version.

6.1.2 Melody Questionnaire

The questionnaire was designed to assess the quality of each melody evaluated by the participant. The questionnaire consists of five statements, as shown in Table 6.1. All statements are in a closed format and require the participant to respond to a Likert scale [43]. The optional responses for the participant are strongly agree, agree, neutral, disagree and strongly disagree. The advantage of using the Likert scale is that it facilitates an instinctive response from the participants.

	Strongly disagree	**Disagree**	**Neutral**	**Agree**	**Strongly agree**
I am familiar with these lyrics.					
I am familiar with this melody.					
The melody is pleasing.					
The melody fits well with the lyrics.					
This style of melody is the style that I expect to accompany with these lyrics.					

Table 6.1 Melody questionnaire

6.1.3 Melody Evaluation

For each language, eight participants who know the language were asked to evaluate ten melodies which were a mixture of lyrics with the original human-composed melodies and lyrics with the generated melodies. For each melody, the participant made their evaluation by watching a video playing the melody along with its lyrics. Then after finish watching the video, the participant was asked to fill in the questionnaire shown in Table 6.1.

6.1.4 Experimental Results

To analyze the results, each scale of all Likert items was assigned to an integer value, as shown in Table 6.2. Then, each statement of each lyric's language was calculated to find the average responses for each system as follows. For each participant, the average response to each system from the scores of all songs generated by the system were calculated. Then for each system, the average scores from all participants were averaged to obtain the final average score. The results of all five statements are shown in Table 6.3 through Table 6.7. Note that 'GM' stands for 'Generated Melody' and the 'OM' stands for 'Original Melody'.

Likert scale	Integer value
Strongly agree	5
Agree	4
Neutral	3
Disagree	2
Strongly disagree	1

Table 6.2 Assigned integer values of Likert scales

Table 6.3 shows the average responses to the statement about familiarity with the lyrics for each language. The lyrics were rated as more familiar when they were paired with the original melodies. However, statistical test result shows that they were not significantly different at the $p < 0.01$ level.

	English	Cantonese	Mandarin	Thai	Japanese
GM	2.03	2.03	1.83	2.35	1.73
OM	2.50	2.75	2.33	2.98	1.90

Table 6.3 Average responses to the statement "I am familiar with these lyrics."

Table 6.4 shows the average responses to the statement about familiarity with the melody for each language. On average, for all languages, the melodies of the original version were rated as more familiar than the melodies of the generated version. All generated melodies received average scores of less than 2.2 for all languages. The trend of the differences in the ratings between the generated melodies and original melodies was similar to the trend of the differences for the melodies' familiarity. Thai songs were the most familiar, Cantonese songs were the second most familiar and Japanese songs were the least familiar.

	English	Cantonese	Mandarin	Thai	Japanese
GM	1.98	1.63	1.53	2.18	2.00
OM	2.63	3.20	2.65	3.40	2.38

Table 6.4 Average responses to the statement "I am familiar with this melody."

Generally, it is difficult for people to accept new melodies if they are very familiar with the old ones. Therefore, Table 6.5, Table 6.6 and Table 6.7 show only the responses for which the alternate melodies and lyrics were not familiar to the participants.

Table 6.5 shows the average responses to the statement about pleasantness of melody. The scores of the generated melodies show that the generated melody for Japanese lyrics was the most pleasing. English, Cantonese and Thai received almost the same average ratings. The average rating of the generated melody for Japanese lyrics was not significantly different to that of the original version at the $p < 0.05$ level.

	English	**Cantonese**	**Mandarin**	**Thai**	**Japanese**
GM	2.55	2.60	2.34	2.58	**2.90**
OM	3.68	3.56	3.91	4.07	3.30

Table 6.5 Average responses to the statement "The melody is pleasing."

The average responses to the statement as to whether the melody fits well with the lyrics are shown in Table 6.6. The results show that the generated melody of Japanese lyrics was rated on average as fitting better with the lyrics than the original melody.

	English	**Cantonese**	**Mandarin**	**Thai**	**Japanese**
GM	2.56	2.33	2.49	2.52	**3.11**
OM	3.94	3.91	4.02	4.40	2.83

Table 6.6 Average responses to the statement "The melody fits well with the lyrics."

Table 6.7 shows the responses to the statement "This style of melody is the style that I expect to accompany with these lyrics." The results show that the original melodies were rated on average higher for all languages. Also, the average rating of the generated melody for Japanese lyrics was not significantly different to that of the original version at the $p < 0.05$ level.

	English	**Cantonese**	**Mandarin**	**Thai**	**Japanese**
GM	2.40	2.39	2.34	2.75	2.87
OM	3.51	3.73	3.76	4.12	3.00

Table 6.7 Average responses to the statement "This style of melody is the style that I expect to accompany with these lyrics."

To assess how creative acts are performed by a computer, Colton et al. introduced two models of computational creativity theory, the FACE and the IDEA models [44]. Using the FACE model in a qualitative way, the values from aesthetic functions are used to compare creative acts. A concept/expression pair generated by the system is denoted as (c_i^g, e_i^g). Let S be a system to be assessed which produces $(c_1^g, e_1^g), \ldots, (c_n^g, e_n^g)$ concept/expression pairs. An aesthetic measure of evaluation is denoted as $\overline{a^g}$. Also, let *t* be a minimum acceptable aesthetic threshold value. Five measures are described as follows:

$$average(S) = \frac{1}{n}\, \sigma_{i=1}^{n} \overline{a^g}(c_i^g, e_i^g) \tag{15}$$

$$best\_ever(S) = max_{i=1}^{n}(\overline{a^g}(c_i^g, e_i^g)) \tag{16}$$

$$worst\_ever(S) = min_{i=1}^{n}(\overline{a^g}(c_i^g, e_i^g)) \tag{17}$$

$$precision(S) = \frac{1}{n}\left|\left\{\left(c_i^g, e_i^g\right) : 1 < i < n \wedge \overline{a^g}(c_i^g, e_i^g) > t\right\}\right| \tag{18}$$

$$reliability(S) = best\_ever(S) - worst\_ever(S). \tag{19}$$

The precision is a valuable measure obtained by dividing the number of generated works by the number of works which meet a minimum acceptable aesthetic level. The reliability is the ability of the software to deliver consistent results, measured by calculating a range between the best output and the worst output.

Table 6.8 reports the results of the assessment of the generated melodies (GM) and the original melodies (OM) for all five languages for the statement “The melody is pleasing.” To calculate the precision, the minimum acceptable aesthetic threshold *t* was set to be the worst score obtained from the original melodies. The results show that 85% of the generated melodies for Japanese lyrics and also more than two-thirds of the generated melodies for English lyrics were rated to be better than the worst original songs. For the reliability, smaller scores are better. The reliability of the generated melody for Japanese lyrics is identical to the reliability of the original melody.

	English		**Cantonese**		**Mandarin**		**Thai**		**Japanese**	
	GM	**OM**	**GM**	**OM**	**GM**	**OM**	**GM**	**OM**	**GM**	**OM**
average	2.55	3.68	2.60	3.56	2.34	3.91	2.58	4.07	2.90	3.30
best_ever	3.50	4.50	3.38	4.50	3.25	4.50	3.25	4.63	3.88	4.25
worst_ever	1.88	2.50	1.88	2.63	1.63	3.25	2.13	3.25	2.13	2.50
precision	**0.69**	1.00	0.59	1.00	0.40	1.00	0.41	1.00	**0.85**	1.00
reliability	1.63	2.00	1.50	1.88	1.63	1.25	1.13	1.38	**1.75**	1.75

Table 6.8 Assessment of creativity acts calculated on average responses to the statement “The melody is pleasing.”

The average responses to the statement “The melody fits well with the lyrics.” were assessed in terms of the creativity acts shown in Table 6.9. Similarly to the previous assessment, the worst score of the original melody was used as the minimum acceptable aesthetic threshold. The results show that most of the generated melodies for Japanese lyrics and also almost 60% of the generated melodies for English lyrics meet the threshold. A song generated from Japanese lyrics also more reliably matches the lyrics than the original song.

	English		**Cantonese**		**Mandarin**		**Thai**		**Japanese**	
	GM	**OM**	**GM**	**OM**	**GM**	**OM**	**GM**	**OM**	**GM**	**OM**
average	2.56	3.94	2.33	3.91	2.49	4.02	2.52	4.40	3.11	2.83
best_ever	3.63	4.50	3.13	4.63	3.88	4.75	3.13	4.88	3.75	3.63
worst_ever	2.00	2.88	1.75	3.25	1.38	3.00	2.13	3.63	2.50	1.75
precision	**0.59**	1.00	0.31	1.00	0.51	1.00	0.31	1.00	**0.95**	1.00
reliability	1.63	1.63	1.38	1.38	2.50	1.75	1.00	1.25	**1.25**	1.88

Table 6.9 Assessment of creativity acts calculated on average responses to the statement “The melody fits well with the lyrics.”

To summarize, the results show that in overview the results for the generated melodies were worse than for the original melodies. However, the generated melody of Japanese lyrics was rated on average as fitting better with the lyrics than the original melody. Also, almost 60% of the generated melodies for English lyrics, about 50% for Mandarin and less than 50% for Cantonese and Thai were rated as fitting better than the worst original melody. The expectation of the style of the generated melody of Japanese lyrics was not significantly different from the original melody. 85% of the generated melodies for Japanese lyrics and also more than two-thirds of the generated melodies for English lyrics were rated as more pleasing than the worst original melodies. The discussion of these results are shown in Section 7.1.

6.2 Statistical Music Accompaniment Generator

The second experiment was conducted to assess the ability of automatic accompaniment generation, which produces suitable chord sequences for melodies. In fact, there is no single correct accompaniment for a particular melody because the chords can be assigned differently depending on musicians or genres. Also, different accompaniment patterns can be produced for a single melody. Therefore, the goal of conducting the experiment was not to focus on the evaluation of producing chords correctly but to evaluate in terms of subjective quality.

6.2.1 Accompaniment Preparation

Given a melody, three types of accompaniment were generated to accompany the same melody for conducting the experiment. The first type was generated by the proposed statistical music accompaniment generator (SMAG). The second type was generated from 'Band-in-a-Box' (BIAB), which is commercial software for generating chord sequences. The last type was generated by a musician (MUS).

All three types of accompaniment were generated for each of eighteen melodies, ranging from fifteen to twenty-five seconds in length. All melodies with their generated accompaniment were exported to MIDI files without any filtering. Also, each song was fixed to use piano sounds in order to present a fair comparison across the three types of accompaniment. In this experiment, the SMAG generated the probabilistic tables based on 53 English pop songs.

6.2.2 Accompaniment Evaluation

Seventeen participants were asked to listen to melodies for the evaluation. Each participant was asked to listen 18 melodies in a randomized order, and each melody was presented as a pair of accompaniments from two of three types. Note that within a paring, the accompaniment parts were different depending on the selected type but the melody was identical. The order of accompaniments in each pair was randomly selected, but each participant saw each possible ordered pairing of the three types three times.

Each participant was asked to rate each accompaniment on its subjective quality. He/she was required to listen to both accompaniments before assigning scores. The rating scale was a scale from one to ten for which each participant assigned the scores by considering the relative subjective quality of each pair of accompaniments. Therefore, the preferred accompaniment would receive a higher score. Also, tied scores were allowed.

6.2.3 Experimental Results

Figure 6.1 shows the mean scores assigned to the three types of accompaniment along with the standard error of each mean. The mean scores assigned to the SMAG, MUS and BIAB were 5.1, 5.4 and 7.2 respectively. The ANOVA was performed to isolate the effects of the rating using 95% confidence intervals. There are two effects, which are the effect of the rater and the effect of the condition. For the effect of the rater, the ANOVA result shows that there are some statistically significant differences among the raters. This result is expected because a large group of raters certainly have divergent means on a subjective scale. For the effect of the condition, the ANOVA result shows that there is at least one of the means that is significantly different. Post-hoc tests were performed, and the tests revealed that there are significant differences between the SMAG and the MUS conditions and between the BIAB and the MUS conditions, but not between the SMAG and BIAB conditions.

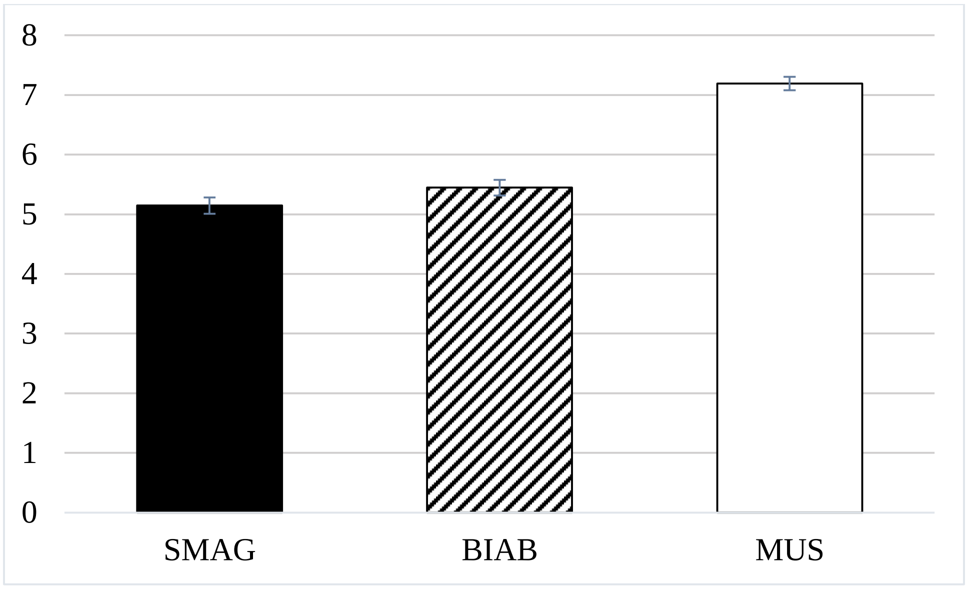


Figure 6.1 Mean scores of three types of accompaniment

The win/loss records of each condition are also shown in Table 6.10. When a SMAG accompaniment was paired against a BIAB accompaniment, the BIAB accompaniment was preferred 46 times, the SMAG accompaniment was preferred 39 times, and 17 times they had tied scores. These results show that the chords generated by the SMAG can compete with the chords generated by the BIAB.

Comparison	**Wins for each condition**
SMAG vs. MUS	SMAG 18, MUS 78, Ties 6
SMAG vs. BIAB	SMAG 39, BIAB 46, Ties 17
MUS vs. BIAB	MUS 74, BIAB 18, Ties 10

Table 6.10 Win/loss records of each condition

To conclude, based on the mean scores from the three types of accompaniment and the statistical hypothesis test, the SMAG can produce the accompaniment in the range of the generated accompaniment by the BIAB. However, it cannot reach the range of human-composed accompaniment, which is similar to the BIAB. The discussion of these results is shown in Section 7.2.

CHAPTER 7
DISCUSSION

7.1 Composing Melody for Different Languages

Based on the results from the survey, the generated melodies for Japanese lyrics obtained the highest scores in terms of the average and the precision scores. To analyze these results, the average ratings for each generated melody of all languages were calculated and analyzed. The results and also the participants' opinion show that there are several factors affecting the rating scores as follows.

The first factor is pitch. Each note relates to a syllable of lyrics and has its corresponding pitch relating to a tone of the syllable. The pitch difference within a phrase, even in the melody rated with the lowest score, is acceptable based on the participants' opinions. Although the pitch difference within a phrase may be pleasing, the overall pitch of the melody affects the rating scores. From the results, the melodies with the higher pitches tend to be rated higher. This implies that the taste of the raters affects the rating. In this case, the raters tend to prefer higher pitches than lower pitches.

The second factor is the duration sequence containing durations of notes. The duration sequence is obtained by randomly choosing from top-N sequences, given pitch difference and tones. However, there is an issue with note duration within a phrase. Without considering the pronunciation of words, it leads to the issue that the melody does not fit well with the lyrics. To illustrate, the example of Thai lyrics is discussed as follows. There is a case that a syllable within a word is assigned too long a note duration, and also in the case of a syllable with a short vowel makes a strange melody. Suppose that the Thai lyrics 'เธอต้องทำงาน', containing three words, 'เธอ' (you), 'ต้อง' (need to), and 'ทำงาน' (work), can be obtained as a phonetic sequence as {t^həə, tɔ̂ŋ, t^ham, ŋaan} and a sequence of alignments as {(เธอ, (t^həə')), (ต้อง, (tɔ̂ŋ)), (ทำงาน, (t^ham, ŋaan))}. Commonly, the word 'ทำงาน' is always pronounced as two syllables 't^ham' and 'ŋaan' continuously and also the first syllable, ‘t^{h}am’, has a short vowel. However, after generating a melody, the note duration of the syllable ‘t^{h}am’ is too long, which makes a strange melody for these lyrics.

The third factor is note value. A melody containing phrases which have more notes, with short values, will be rated higher than a melody containing phrases which have fewer notes, with long values. For example, most phrases in song1 have many short value notes, such as semiquavers, while the phrases in song2 have few notes and they have long values, such as a semibreve with a tie sign. Song1 tends to be rated higher for the following reasons. Firstly, when the notes have short values, it can also reduce the effect of the second factor. Second, since the generated melody is a MIDI file based on the sound of a single instrument, the melody with shorter note values sounds more pleasing.

The last factor is the meaning of lyrics. Melodies which express feeling that harmonizes with the meaning of the lyrics tend to be rated higher. The following is an example of a melody which does not harmonize with the lyrics. The lyrics are about everything being arranged by God and the singer being brave to love, no matter what. However, the generated melody expresses sadness. In fact, it would be much better to express a positive feeling than sadness to harmonize with these lyrics. Therefore, it is very important that the generated melody be harmonized with the meaning of the lyrics.

Besides the above, biases may occur as follows. Firstly, breaking a whole melody into sub-melodies for the evaluation may cause bias because some sub-melodies may have several repeated patterns which are not pleasing compared with other sub-melodies. Listening to the whole piece is better to see the overall structure of the melody but it is too long for the evaluation. Secondly, some biases can be caused by the participants themselves. For example, different participants have different tastes in music. Also, even if the participant knows the lyrics’ language, he/she may not listen to music frequently or have no musical experience.

7.2 Statistical Music Accompaniment Generator

Analysis of the experimental results indicates that the songs generated by the MUS were rated as being significantly more pleasing. However, the SMAG can produce the accompaniment in the range of the generated accompaniment by the BIAB. This means that the SMAG has the same good performance level as the commercial software within the scope of the experiment. The SMAG was analyzed and compared with the MUS in order to point out the causes why the SMAG could not reach the level of the MUS. There are two observed causes, which are accompaniment patterns and ornaments, which are described in the following.

The first observation is accompaniment patterns. Different accompaniment patterns lead to different styles of songs. Each style has its own impact on the emotions. In the SMAG, different groups of accompaniment patterns are specified for different time signatures, and the SMAG currently chooses the accompaniment pattern by randomly choosing for the verse and chorus of the song. Also, the accompaniment pattern chosen for the chorus contains more notes than for the verse. By the MUS, the accompaniment patterns are chosen by considering the overall given melody first in order to analyze the style of the melody. Although all different accompaniment patterns can make a song good, the melody and the structure of the song need to be considered before choosing the suitable accompaniment patterns because some accompaniment patterns may be biased against melody and other patterns may be biased against rhythm. Moreover, every part of a song inextricably links with all the other parts. This implies that previous accompaniments also have some effect on the current accompaniment. Therefore, to choose the suitable accompaniment patterns which harmonize with the given melody, it is necessary to consider the overall song.

Another observation is ornaments. Ornaments are fast notes performed around a central note in order to produce musical flourishes. They are added by the composer and are not necessary to carry the overall line of the accompaniment. By the MUS, arpeggiation, which is one type of ornament, has been included for the accompaniment composition but not in the SMAG. Therefore, listeners may be pleased more by the MUS more than the SMAG.

CHAPTER 8
CONCLUSION AND FUTURE WORK

This thesis presents a powerful tool for composing melody for different languages and a statistical music accompaniment generator. The melody composer aims to compose pleasing melodies for lyrics in different languages based on a lyric-note correlation considering a tone sequence of the lyrics. Lyrics processing for Thai and Japanese are presented. A hybrid approach combining a dictionary-based with a rule-based approach for Thai syllabification is proposed in order to obtain a tone sequence corresponding to the given lyrics for melody generation. Similarly, for Japanese lyrics, dynamic programming is introduced for word segmentation in order to obtain a tone sequence. A method of automatic alignment for Japanese is also proposed. Moreover, a statistical music accompaniment generator is presented. The system aims to generate appropriate accompaniment with different rhythmic patterns for a given melody based on a probabilistic model.

In this thesis, two studies were conducted to evaluate the effectiveness of the melody composer for different languages and the statistical music accompaniment generator in terms of subjective quality. The first study was conducted to evaluate the melody composer for English, Cantonese, Mandarin, Thai and Japanese lyrics. The results show that the generated melody for Japanese lyrics obtained the highest score and was more subjectively acceptable compared with the original melody.

The limitations of this system are as follows. Obtaining Japanese phonetic and tone sequences is limited by the dictionary coverage. Also, the system can generate a melody in a single style. As mentioned in the discussion chapter, there is still a lot of potential to improve the system, for example, by tuning the pitch of the melody according to the gender of the user such as a higher pitch for female, improving the method of getting the duration sequence by considering the type of vowel of a syllable, including the rest note in to the melody sequence, considering the emotion, feeling and the style of the melody which harmonizes with the lyrics by classifying the existing songs in the database into different categories.

The second study was conducted to evaluate the statistical music accompaniment generator by comparing the generated accompaniment from the proposed system with the generated accompaniment from existing commercial software (BIAB) and a musician’s composition. The results show that the proposed generator can produce accompaniment in the range of the BIAB.

The limitations of this system are as follows. The input melody has no key change. The system supports only one style of the accompaniment and can generate only single instrument. There is also a lot of potential to improve the statistical music accompaniment generator to reach the range of the musician’s composition, as discussed in Chapter 7. For example, besides, grouping the accompaniment patterns according to different time signatures, they can also be grouped into different genres. Also, the songs in the database can also be classified into different genres, such as pop, country, jazz and rock, in order to generate probabilistic tables for each genre and produce accompaniment according to different styles. A cross-checking can also be included for checking the generated accompaniment after accompaniment generation. Moreover, the proposed model can be modified to support many musical instruments. Since the instruments act semi-independently in a song according to a style of a song, they can be assigned different music note information such as volume, duration and the duration of the melody that they should play.

REFERENCES

[1] “Band-in-a-box.” http://www.pgmusic.com/. Accessed: 2015-05-18.

[2] “Jammer professional.” http://www.soundtrek.com/. Accessed: 2015-05-18.

[3] “Songsmith.” http://research.microsoft.com/en-us/um/redmond/projects/songsmith/. Accessed: 2015-05-23.

[4] I. Simon, D. Morris, and S. Basu, “Mysong: automatic accompaniment generation for vocal melodies,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2008, pp. 725–734.

[5] “One man band.” http://www.1manband.nl/omb.htm. Accessed: 2015-05-23.

[6] “Harmony assistant.” http://www.myriad-online.com/en/products/harmony.htm. Accessed: 2015-05-23.

[7] G. Nierhaus, *Algorithmic Composition: Paradigms of Automated Music Generation*. Springer Science & Business Media, 2009.

[8] J. D. Fernández and F. Vico, “Ai methods in algorithmic composition: A comprehensive survey,” *Journal of Artificial Intelligence Research*, pp. 513–582, 2013.

[9] G. M. Rader, “A method for composing simple traditional music by computer,” *Communications of the ACM*, vol. 17, no. 11, pp. 631–638, 1974.

[10] C. Roads and P. Wieneke, “Grammars as representations for music,” *Computer Music Journal*, pp. 48–55, 1979.

[11] S. Gill, “A technique for the composition of music in a computer,” *The Computer Journal*, vol. 6, no. 2, pp. 129–133, 1963.

[12] M. T. Thomas, “Vivace: A rule based {AI} system for composition,” 1985.

[13] K. Jones, “Compositional applications of stochastic processes,” *Computer Music Journal*, pp. 45–61, 1981.

[14] P. Langston, “Six techniques for algorithmic music composition,” in *15th International Computer Music Conference*, Citeseer, 1989.

[15] F. Pachet, P. Roy, G. Barbieri, and S. C. Paris, “Finite-length Markov processes with constraints,” *transition*, vol. 6, no. 1/3, 2001.

[16] M. Allan and C. K. Williams, “Harmonising chorales by probabilistic inference,” *Advances in Neural Information Processing Systems*, vol. 17, pp. 25–32, 2005.

[17] D. Morris, I. Simon, and S. Basu, “Exposing parameters of a trained dynamic model for interactive music creation.,” in *AAAI*, pp. 784–791, 2008.

[18] P. M. Todd, “A connectionist approach to algorithmic composition,” *Computer Music Journal*, pp. 27–43, 1989.

[19] A. Melo, “A connectionist model of tension in chord progressions,” *Master’s thesis, University of Edinburgh*, 1998.

[20] A. E. Coca, R. A. Romero, and L. Zhao, “Generation of composed musical structures through recurrent neural networks based on chaotic inspiration,” in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, IEEE, 2011, pp. 3220–3226.

[21] K. Ricanek, A. Homaifar, and G. Lebby, “Genetic algorithm composes music,” 1993.

[22] M. Johnson, D. R. Tauritz, and R. Wilkerson, “Evolutionary computation applied to melody generation,” in *Proc. of the ANNIE*, 2004.

[23] E. Özcan and T. Erçal, “A genetic algorithm for generating improvised music,” in *Artificial Evolution*, Springer, 2008, pp. 266–277.

[24] K. Verbeurgt, M. Fayer, and M. Dinolfo, “A hybrid neural-Markov approach for learning to compose music by example,” in *Advances in Artificial Intelligence*, Springer, 2004, pp. 480–484.

[25] C. Thornton, *Hierarchical Markov modeling for generative music*. Ann Arbor, MI: MPublishing, University of Michigan Library, 2009.

[26] B. Manaris, P. Roos, P. Machado, D. Krehbiel, L. Pellicoro, and J. Romero, “A corpus-based hybrid approach to music analysis and composition,” in *Proceedings of the National*

Conference on Artificial Intelligence, vol. 22, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007, p. 839.

[27] C. Long, R.-W. Wong, and R. K. W. Sze, “T-music: A melody composer based on frequent pattern mining,” in *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, IEEE, 2013, pp. 1332–1335.

[28] B. Pardo and W. P. Birmingham, “Algorithms for chordal analysis,” *Computer Music Journal*, vol. 26, no. 2, pp. 27–49, 2002.

[29] U. S. CUNHA and G. RAMALHO, “An intelligent hybrid model for chord prediction,” *Organised Sound*, vol. 4, pp. 115–119, 6 1999.

[30] A. Sheh and D. P. Ellis, “Chord segmentation and recognition using em-trained hidden Markov models,” *ISMIR 2003*, pp. 185–191, 2003.

[31] J.-F. Paiement, D. Eck, and S. Bengio, “Probabilistic melodic harmonization,” in *Advances in Artificial Intelligence*, Springer, 2006, pp. 218–229.

[32] C.-H. Chuan and E. Chew, “A hybrid system for automatic generation of style-specific accompaniment,” in *4th Intl Joint Workshop on Computational Creativity*, 2007.

[33] R. Groves, “Automatic harmonization using a hidden semi-Markov model,” in *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2013.

[34] J. P. Forsyth and J. P. Bello, “Generating musical accompaniment using finite state transducers,”

[35] Y. Poowarawan, “Dictionary-based Thai syllable separation,” in *Proceedings of the Ninth Electronics Engineering Conference*, 1986.

[36] S.-n. Kongsupanich, *The Transformation of Thai Morphemes to Phonetic Symbols for Thai Speech Synthesis System*. PhD thesis, Master Thesis. Faculty of Engineering, King Mongkutâ€™s Institute of Technology Ladkrabang, Bangkok, 1997.

[37] V. Lorchirachoonkul and C. Khuwinphunt, “Thai soundex algorithm and Thai-syllable separation algorithm,” tech. rep., Research Report. School of Applied Statistics, National Institute of Development Administration, Bangkok, 1982.

[38] C. Jucksriporn and O. Sornil, “A minimum cluster-based trigram statistical model for Thai syllabification,” in *Computational Linguistics and Intelligent Text Processing*, Springer, 2011, pp. 493–505.

[39] S. Khruahong, S. Nitsuwat, and P. Limmaneepraserth, “Thai syllable segmentation for text-to-speech synthesis by using suited-syllable-structure mapping,” in *International Symposium on Communications and Information Technology (ISCIT)*, 2003.

[40] “Text boundary analysis in java.” http://icu-project.org/docs/papers/-text_boundary_analysis_in_java/. Accessed: 2015-05-18.

[41] B. D. Giorgi, M. Zanoni, A. Sarti, and S. Tubaro, “Automatic chord recognition based on the probabilistic modeling of diatonic modal harmony,” in *Multidimensional Systems (nDS), 2013. Proceedings of the 8th International Workshop on*, VDE, 2013, pp. 1–6.

[42] C. L. Krumhansl, J. J. Bharucha, and E. J. Kessler, “Perceived harmonic structure of chords in three related musical keys.,” *Journal of Experimental Psychology: Human Perception and Performance*, vol. 8, no. 1, p. 24, 1982.

[43] R. Likert, “A technique for the measurement of attitudes.,” *Archives of psychology*, 1932.

[44] S. Colton, A. Pease, and J. Charnley, “Computational creativity theory: The face and idea descriptive models,” in *Proceedings of the Second International Conference on Computational Creativity*, 2011, pp. 90–95.

APPENDIX A

COMPOSING MELODY FOR DIFFERENT LANGUAGES

A.1 Iterative Segmentation Algorithm

Process	**Explanation**
t h e m e| n i n b l a c k	The first word is 'theme', obtained by the maximal matching algorithm.
t h e m e| n| i n b l a c k	Finding the next possible word, 'n' matches the dictionary.
t h e m e| n i| n b l a c k	'ni' matches the dictionary.
t h e m e| n i n| b l a c k	'nin' matches the dictionary.
t h e m e| n i n b| l a c k	'ninb' mismatches the dictionary so find the longest word which exists in the dictionary from 'ninb'. However, the word is not found so move to find the word in 'theme' instead.
t h e m| e n i n b l a c k	The word 'them' is found and start over the process.
t h e m| e| n i n b l a c k	'e' matches the dictionary.
t h e m| e n| i n b l a c k	'en' matches the dictionary.
t h e m| e n i| n b l a c k	'eni' matches the dictionary.
t h e m| e n i n| b l a c k	'enin' mismatches the dictionary so find the longest word which exists in the dictionary from 'enin'. However, the word is not found so move to find the word in 'them' instead.
t h e| m e n i n b l a c k	The word 'the' is found and start over the process.
t h e| m| e n i n b l a c k	'm' matches the dictionary.
t h e| m e| n i n b l a c k	'me' matches the dictionary.
t h e| m e n| i n b l a c k	'men' matches the dictionary.
t h e| m e n i| n b l a c k	'meni' matches the dictionary.
t h e| m e n i n| b l a c k	'menin' matches the dictionary.

t h e| m e n i n b| l a c k	‘meninb’ mismatches the dictionary so find the longest word which exists in the dictionary from ‘meninb’.
t h e| m e n| i n b l a c k	The word ‘men’ is found and start over the process from this point.
t h e| m e n| i| n b l a c k	‘i’ matches the dictionary.
t h e| m e n| i n| b l a c k	‘in’ matches the dictionary.
t h e| m e n| i n b| l a c k	‘inb’ matches the dictionary.
t h e| m e n| i n b l| a c k	‘inbl’ mismatches the dictionary so find the longest word which exists in the dictionary from ‘inbl’.
t h e| m e n| i n| b l a c k	The word ‘in’ is found and start over the process from this point.
t h e| m e n| i n| b| l a c k	‘b’ matches the dictionary.
t h e| m e n| i n| b l| a c k	‘bl’ matches the dictionary.
t h e| m e n| i n| b l a| c k	‘bla’ matches the dictionary.
t h e| m e n| i n| b l a c| k	‘blac’ matches the dictionary.
t h e| m e n| i n| b l a c k|	‘black’ matches the dictionary and it reaches the end of the string so the process is done, obtaining {the, men, in, black}.

Table A.1 Example of iterative segmentation algorithm

A.2 Two Consonants in Thai Language

Thai allows concatenation of two consonants. Let c_1 be the first consonant and c_2 be the second consonant. The two consonants can be denoted as c_1c_2 according to the rules shown in Table A.2.

$\boldsymbol{c_1}$	$\boldsymbol{c_2}$
ก, ข, ค, จ, ซ, ต, ป, พ, ฟ, ท, ศ, ส, ห	ร
ก, ข, ค, ป, พ, ห	ล
ก, ข, ค, ห	ว
ห	ง, ญ, น, ม, ย, ร, ล, ว
อ	ย

Table A.2 Rules of Thai two consonants

A.3 Thai Characters – Phonetic Symbols Mapping

Thai characters are mapped to specific phonetic symbols according to types of speech sound of the characters in syllables, as shown in Table A.3. There are three types of speech sound, which are the initial consonant, vowel and final consonant.

Initial Consonant	Phonetic Symbol
ก	k
ข	kʰ
ฃ	kʰ
ค	kʰ
ฅ	kʰ
ฆ	kʰ
ง	ŋ
จ	j
ฉ	cʰ
ช	cʰ
ซ	s
ฌ	cʰ
ญ	y
ฎ	d
ฏ	t
ฐ	tʰ
ฑ	tʰ
ฒ	tʰ
ณ	n
ด	d
ต	t
ถ	tʰ
ท	tʰ
ธ	tʰ

Vowel	Phonetic Symbol
ะ	a
า	aa
◌ิ	i
◌ี	ii
◌ึ	ʉ
◌ื	ʉʉ
◌ือ	ʉʉ
◌ุ	u
◌ู	uu
เะ	e
เ◌็	e
เ	ee
แะ	ɛ
แ◌็	ɛ
แ	ɛɛ
โะ	o
โ	oo
เาะ	ɔ
ออ	ɔɔ
เะอ	ə
เอ	əə
เ◌ิ	əə
เ◌ียะ	ia
เ◌ีย	ia

Final Consonant	Phonetic Symbol
ก	k
ข	k
ฃ	k
ค	k
ฅ	k
ฆ	k
ง	ŋ
จ	t
ฉ	t
ช	t
ซ	t
ฌ	t
ญ	n
ฎ	t
ฏ	t
ฐ	t
ฑ	t
ฒ	t
ณ	n
ด	t
ต	t
ถ	t
ท	t
ธ	t

Initial Consonant	Phonetic Symbol
น	n
บ	b
ป	p
ผ	p^h
ฝ	f
พ	p^h
ฟ	f
ภ	p^h
ม	m
ย	y
ร	r
ล	l
ฤ	r
ฦ	l
ว	w
ศ	s
ษ	s
ส	s
ห	h
ฬ	l
อ	u
ฮ	h

Vowel	Phonetic Symbol
เ◌ือะ	ʉa
เ◌ือ	ʉa
◌ัวะ	ʉa
◌ั	ua
รร	a
◌ำ	am
ไ	ay
ใ	ay
เา	aw

Final Consonant	Phonetic Symbol
น	n
บ	b
ป	p
ผ	p
ฝ	p
พ	p
ฟ	p
ภ	p
ม	m
ย	y
ร	n
ล	n
ฤ	n
ฦ	n
ว	w
ศ	t
ษ	t
ส	t
ห	n
ฬ	n
อ	ɔɔ

Table A.3 Thai characters - phonetic symbols mapping

A.4 Thai Tone Rules

Thai tone rules are indicated by considering the consonants, vowels, and tone marks in a syllable. Thai has five tones, which are mid (M), low (L), falling (F), high (H) and rising (R). Table A.4 presents the Thai tone rules.

Initial consonant class	**No tone mark**					**Tone mark**			
	the syllable is open and has a **long** vowel	the syllable is closed with a **sonorant** (**live**) consonant ending	the syllable is open and has a **short** vowel	the syllable is closed with a **stop** (**dead**) final consonant and has a		◌่	◌้	◌๊	◌๋
				short vowel	**long vowel**				
	Live syllable		**Dead syllable**						
Low: ค ฅ ฆ ง ช ซ ฌ ญ ฑ ฒ ณ ท ธ น พ ฟ ภ ม ย ร ล ว ฬ ฮ	M		H		F	F	H		
Mid: ก จ ฎ ฏ ด ต บ ป อ	M		L			L	F	H	R
High: ข ฃ ฉ ฐ ถ ผ ฝ ส ห ศ ษ	R		L			L	F		

Table A.4 Thai tone rules

A.5 Automatic Alignment for Japanese

To illustrate Algorithm 3 Automatic Alignment for Japanese, the following is an example of the process. Given a syllable sequence $S(L) = \{粟, 立, つ\}$ and a corresponding phonetic sequence $P(L) = \{\mathrm{a, wa, da, tsu}\}$. The dictionary is also given, as shown in Table A.5. Also, the *matchScore* is initialized to be 10 and the *containScore* is initialized to be 1.

Word	**Phonemes**
粟飯	a wa me shi
粟	a wa
粟粒	zo ku ryu u
糯粟	mo chi a wa
雛罌粟	hi na ge shi
粳粟	u ru a wa
苛立たしい	i ra da ta shi i
いら立つ	i ra da tsu
つ	tsu

Table A.5 Japanese Dictionary

The processes of obtaining the segmented phonemes with the greatest scores are shown as follows.

1. Based on the given $S(L)$ with length 3 and $P(L)$ with length 4, the number of possible cases of phoneme segmentation is $C_2^3 = 3$. The three possible cases assigning phonemes to every single character are as follows.

   Case 1: [a] [wa] [da, tsu]

   Case 2: [a] [wa, da] [tsu]

   Case 3: [a, wa] [da] [tsu]

2. For each case, a score is calculated for each single character. Only case 3 is illustrated for the score calculation.

   Case 1: [a] + [wa] + [da, tsu] = 4 + 0 + 1 = 5

   Case 2: [a] + [wa, da] + [tsu] = 4 + 0 + 11 = 15

   Case 3: [a, wa] [da] [tsu]

   [a, wa] is corresponding to the first character ‘粟’.

   First, the score is initialized to be 0. Then, the assigned phoneme is compared to every assigned phoneme of the same character from the words in the dictionary.

Word	**Phonemes**	**Score**
粟飯	**a wa** me shi	1
粟	**a wa**	11
粟粒	zo ku ryu u	11
糯**粟**	mo chi **a wa**	12
雛罌**粟**	hi na ge shi	12
粳**粟**	u ru **a wa**	13
苛立たしい	i ra da ta shi i	13
いら立つ	i ra da tsu	13
つ	tsu	13

Thus, the score for [a, wa] is 13.

Next, the score of [da] corresponding to the second character ‘立’ is calculated.

Word	**Phonemes**	**Score**
粟飯	a wa me shi	0
粟	a wa	0
粟粒	zo ku ryu u	0
糯粟	mo chi a wa	0
雛罌粟	hi na ge shi	0
粳粟	u ru a wa	0
苛**立**たしい	i ra **da** ta shi i	1
いら**立**つ	i ra **da** tsu	2
つ	tsu	2

Thus, the score for [da] is 2.

Next, the score of [tsu] corresponding to the second character 'つ' is calculated.

Word	**Phonemes**	**Score**
粟飯	a wa me shi	0
粟	a wa	0
粟粒	zo ku ryu u	0
糯粟	mo chi a wa	0
雛罌粟	hi na ge shi	0
粳粟	u ru a wa	0
苛立たしい	i ra da ta shi i	0
いら立**つ**	i ra da **tsu**	1
つ	**tsu**	11

Thus, the score for [tsu] is 11.

Finally the total score of the case 3: [a, wa] [da] [tsu] is 13 + 2 +11 = 26.

3. For each case, the score is adjusted according to the linguistic constraints as follows.

Case	**粟**	**立**	**つ**	**Score**	**Adjusted score**
1	[a]	[wa]	[da, tsu]	5	5
2	[a]	[wa, da]	[tsu]	15	**30**
3	[a, wa]	[da]	[tsu]	26	**52**

The scores of case 2 and case 3 are increased because they match the constraint that the kana character must align directly with the corresponding phoneme of that kana character, which is 'つ' matching [tsu].

4. Case 3 obtains the greatest score. Thus, the syllable sequence $S(L) = \{粟, 立, つ\}$ and the corresponding phonetic sequence $P(L) = \{\mathrm{a}, \mathrm{wa}, \mathrm{da}, \mathrm{tsu}\}$ are obtained as a sequence of alignments as $(L) = \{(粟, (\mathrm{a}, \mathrm{wa})),\ (立, (\mathrm{da})),\ (つ, (\mathrm{tsu}))\}$.