

Doctoral Dissertation

Japanese Incremental Text-to-speech Synthesis based on Accent Phrase Unit

Tomoya Yanagita

Program of Information Science and Engineering
Graduate School of Science and Technology
Nara Institute of Science and Technology

Supervisor: Professor Satoshi Nakamura
Augmented Human Communication Laboratory (Division of Information Science)

Submitted on April 13, 2023

A Doctoral Dissertation
submitted to Graduate School of Science and Technology,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of Engineering

Tomoya Yanagita

Thesis Committee:

Supervisor Satoshi Nakamura
(Professor, Division of Information Science)
 Taro Watanabe
(Professor, Division of Information Science)
 Sakriani Sakti
(Associate Professor, Japan Advanced Institute of Science and Technology)
 Shinnosuke Takamichi
(Lecturer, University of Tokyo)

Japanese Incremental Text-to-speech Synthesis based on Accent Phrase Unit*

Tomoya Yanagita

Abstract

Incremental Text-to-Speech (iTTS) synthesizes a speech incrementally from a synthesis chunk smaller than sentence units. The iTTS is a key component for simultaneous speech-to-speech translation and spoken dialogue systems. The challenge of iTTS is to maintain high speech quality with low latency by optimizing synthesis chunks. Existing iTTS systems use a word unit as a synthesis chunk. The Japanese language has a mora-timed rhythm and a tonal aspect accent, where an accent phrase is a critical unit for representing accents and meaning. This dissertation proposes a high-quality Japanese iTTS system with low latency by using accent phrase units. We first propose HMM-based Japanese iTTS and investigate the speech quality and latency. Experimental results show that (1): an accent phrase unit and features are necessary to improve speech quality for Japanese iTTS, (2): using the following one accent phrase unit effectively improves speech quality. Second, we investigate neural iTTS for Japanese. The neural iTTS systems proposed for English use a prefix-to-prefix neural iTTS framework with 1-2 word units a look-ahead. Since the Japanese language is based on accent phrase units, using a prefix-to-prefix neural iTTS with a look-ahead approach increases latency. We propose an alternative approach to the neural iTTS that does not use look-ahead. We propose a method that uses an accent phrase unit and exploits information embedded in the previous synthesizing step. We experimentally investigated latency and speech quality. The experimental results show that the proposed Japanese iTTS is able to synthesize

*Doctoral Dissertation, Graduate School of Science and Technology, Nara Institute of Science and Technology, April 13, 2023.

better speech quality, with a similar latency range, than that of the conventional baseline prefix-to-prefix neural iTTS with word units. The proposed approach can be applied to the other tonal-aspect accented languages.

Keywords:

Incremental TTS system, HMM, End-to-end, Japanese language, Accent phrase

Contents

1. Introduction	1
1.1 Research Background	1
1.2 Research Scopes and Future Directions in This Dissertation	6
1.3 Contribution of this Dissertation	7
2. Speech and Text-to-speech Technology	8
2.1 Introduction	8
2.2 Process of Producing Speech and Acoustic Features	8
2.3 HMM-based Text-to-speech	9
2.3.1 Text Processing	10
2.3.2 Definition of HMM	13
2.3.3 Acoustic Model of HMM	14
2.3.4 Context Clustering	15
2.3.5 Generation Algorithm for Acoustic Features	15
2.3.6 Speech Generation by a Digital Filter	16
2.4 End-to-end Based Text-to-speech	17
2.4.1 Text Processing for End-to-end Text-to-speech	17
2.4.2 Definition of Neural Network Framework	18
2.4.3 Attention-based Encoder-decoder for Acoustic Model	22
2.4.4 Forward Attention in Acoustic Model for Text-to-speech	24
2.4.5 Neural Vocoder for Generating a Speech	25
2.5 Summary	27
3. HMM-based Japanese Incremental TTS	29
3.1 Introduction	29
3.2 Related Work of HMM-based Incremental TTS	29
3.3 Proposed Method	30
3.3.1 Linguistic Features	31
3.3.2 Synthesis Chunk	32
3.4 Experiment	33
3.4.1 Dataset and Experimental Conditions	33
3.4.2 Evaluation without Next Linguistic Feature	34

3.4.3	Evaluation with Next Linguistic Feature	40
3.5	Summary	44
4.	Neural End-to-end Incremental TTS	46
4.1	Introduction	46
4.2	Related Works	46
4.3	End-to-end Japanese ITTS Inputs	47
4.4	Methods	48
4.4.1	Sentence-based TTS	48
4.4.2	Word-based iTTS	50
4.4.3	Proposed Accent-phrase-based iTTS	50
4.5	Experiment	53
4.5.1	Dataset and Models	53
4.5.2	Evaluation Indexes	55
4.5.3	Objective Evaluation of Methods	56
4.5.4	Objective Evaluation of Relationship between Speech Quality and Latency	57
4.5.5	Subjective Evaluation of Naturalness	58
4.6	Summary	60
5.	Conclusion and Future Direction	62
5.1	Conclusion	62
5.2	Future Direction	63
References		66
Acknowledgements		73

List of Figures

1	Machine Speech-to-Speech Translation systems.	1
2	Machine incremental Speech-to-Speech Translation system.	2
3	Accent types for the Japanese language. Note that a circle represents mora units, an orange circle represents a high pitch, a blue circle represents a low pitch, and a dashed circle represents the next mora units.	4
4	An example of the Japanese accent sandhi. Note that a circle represents mora units, an orange circle represents a high pitch, a blue circle represents a low pitch, and a dashed circle represents the next mora units.	5
5	Relationships between research scope and this dissertation.	7
6	A producing process of human speech.	8
7	An overview of the HMM-based TTS system.	10
8	An example of Japanese linguistic features for TTS.	11
9	Flowchart of extracting accent phrase information for Japanese TTS inputs	12
10	An illustration of a left-to-right HMM model.	13
11	An illustration of a discrete-time filter on the basis of a source-filter model.	17
12	Visualizing a non-causal CNN.	20
13	Visualizing a causal CNN.	20
14	An architecture of Tacotron2 [1].	24
15	An architecture of Parallel WaveGAN [2]	25
16	A differential process scopes in TTS and iTTS systems.	30
17	Possible ways of connecting several chunks while synthesizing a text. ChunkCurr shows a current synthesis chunk as an accent phrase.	33
18	Comparison of generated F0 sequences based on different linguistic and temporal localities.	39

19	MOS scores of iTTS systems with various chunk connections and its latency.	41
	Note that The total latency for “ChunkCurr+wPast+wNext” would be 1.85. “ChunkCurr+wPast+wAllPast” uses a current accent phrase from previous accent phrases, so the latency for “ChunkCurr+wPast+wAllPast” would range from 1.72 to 1.98 seconds. Latency for ChunkCurr is an actual value (1.72 seconds). Latency for “ChunkCurr+wPast+wNext” and “ChunkCurr+wPast+wAllPast” are simulated values by the latency of “ChunkCurr.”	
20	Relationships between MOS scores with available next features, various chunks, and its latency.	44
	Note that latency for “ChunkCurr” and a TTS are actual values in Table 3. Latency for “ChunkCurr+wPast+wNext” is simulated values by “ChunkCurr” in Table 3.	48
21	Japanese accent features in an accent phrase	48
22	Synthesis chunks and their elements for a sentence-based TTS, a prefix-to-prefix iTTS, and the proposed accent-phrase-based iTTS	49
23	Proposed approaches to the Japanese iTTS system	52
24	Relationship between latency and its frequency. The top figure is (1-2): baseline prefix-to-prefix iTTS with one morphoneme+look-1 (black: average latency of 0.464 seconds) and (3-3): iTTS with one accent phrase as a synthesis chunk (red: average latency of 0.655 seconds). The middle figure is (1-3): baseline prefix-to-prefix iTTS with one morphoneme+look-2 (gray: average latency of 0.572 seconds) and (3-4): iTTS with two accent phrases as a synthesis chunk (yellow: average latency of 1.20 seconds). The bottom figure is (1-4):prefix-to-prefix iTTS with one morphoneme+look-3 (green: average latency of 0.667 seconds).	59
25	Relationship between MOS and its average latency. Note that MOS score of (1-0) is 1.14, (1-2) is 2.78, (1-3) is 2.83, (3-1) is 4.47, (3-3) is 3.45, and (3-4) is 3.87.	60

List of Tables

1	Utilization of Linguistic Features in This Dissertation.	31
2	MCD and f_0 error of cent unit without available next linguistic features for iTTS. Note that a synthesis chunk is sentence units, and linguistic features are used conditions of iTTS.	35
3	Average text length [characters] and latency [seconds]. Note that extracting latency is the time to extract linguistic features from an input text, generating latency is the time to synthesize speech from linguistic features, and playing latency is the time for speech duration in a synthesis chunk. Note that extracting latency is simulated values, and the text of the synthesis chunk converts to linguistic features with a sentence-based text processor.	37
4	Objective and subjective evaluation of iTTS systems with various chunk connections. Note that a synthesis chunk is an accent phrase	40
5	MCD and f_0 an error of cent unit with available next linguistic features for iTTS.	42
6	MOS score with the presence or absence of next linguistic features.	42
7	Evaluation of iTTS systems with various input chunks and available next feature. A ChunkCurr uses an accent phrase. Note that “Pho+POS+Accphr+Next+smoothing” uses the Hanning window function for smoothing each waveform.	43
8	Input feature types and embedding dimensions	54
9	Objective evaluations of proposed methods. Note that a full sentence in a column of the synthesis chunk means a sentence unit as the synthesis chunk, 1 morpheme+look- k means one morpheme as the synthesis chunk with a look-ahead of k length, and 1 accent phrase and 2 accent phrases mean one accent phrase as the synthesis chunk and two accent phrases as the synthesis chunk. . .	57

1. Introduction

1.1 Research Background

Multilingual speech communication is necessary to understand or speak each language for overseas travel, business, and education. To understand a language, it is necessary to learn its vocabulary, grammar, and the speech's phonological information. Learning an additional language other than the mother language requires much time and effort. One of the promising solutions is the construction of a machine Speech-to-Speech Translation (S2ST) system for multilingual speech communication. Figure 1 shows the architectures of a cascaded S2ST system. The Cascaded S2ST system concatenates automatic speech recognition (ASR), machine translation (MT), and text-to-speech (TTS) synthesis [3]. ASR recognizes a source language's text in speech, and MT translates the text of a target language. A TTS system generates the target speech from the translated text. Each cascaded model uses a neural end-to-end system, and these models were recently reported to outperform ASR [4], MT [5], and TTS [1]. The S2ST system works sentence-by-sentence. In other words, the S2S system cannot translate incrementally while the speaker speaks and generates speech. Since spoken speech is long, the S2ST system will cause significant latency in the resulting translation and lack quick responses like human communication.

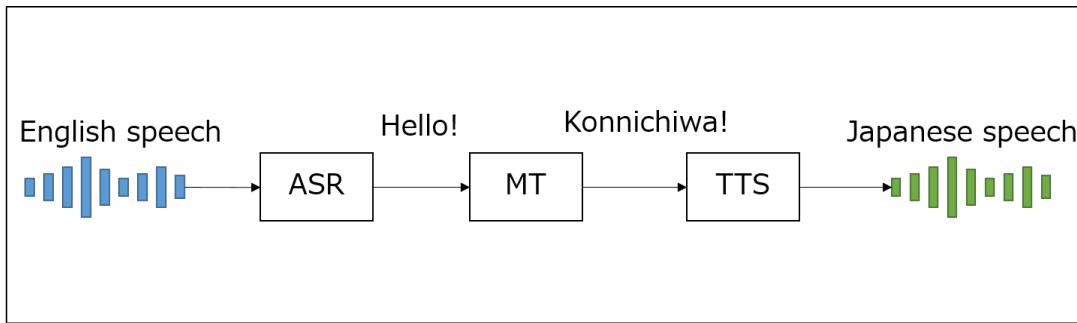


Figure 1. Machine Speech-to-Speech Translation systems.

Human interpreters generally break sentences into smaller chunks, resulting in a shorter latency [6]. As a solution to incrementally generate a speech, several

studies aimed to construct an incremental S2ST system [7, 8, 9, 10, 11] that could produce high-quality speech translations with minimizing the latency of the translation process. Figure 2 shows an incremental S2ST system. The system uses three models: incremental ASR (iASR), incremental MT (iMT), and incremental TTS (iTTS). Each model works chunk-by-chunk at each time step. For instance, a chunk of incremental ASR is the speech segment, and ASR outputs the text word-by-word. After that, iMT translates the source’s one word to target words, and an iTTS system synthesizes a speech for target words. The incremental S2ST system can produce the output’s speech without waiting for the input of a sentence at each time step (Figure 2).

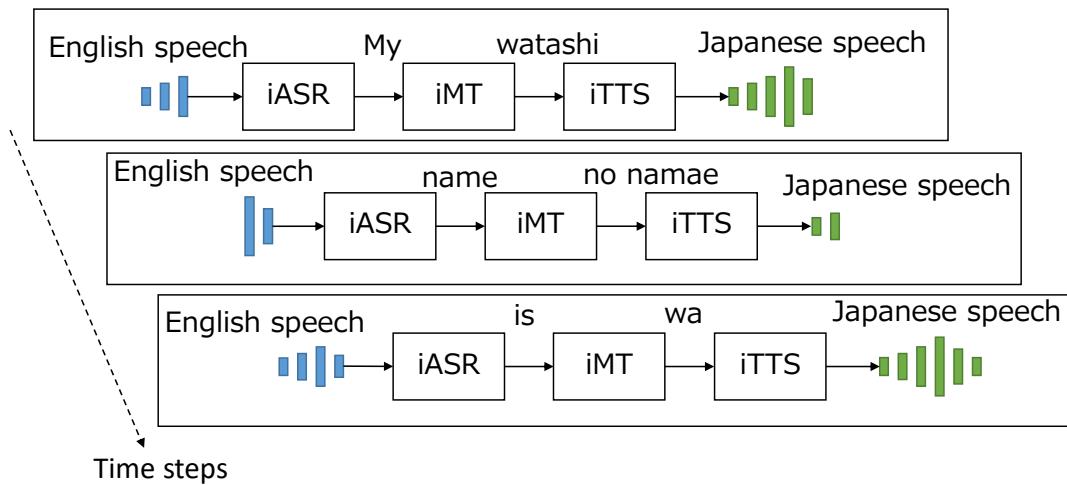


Figure 2. Machine incremental Speech-to-Speech Translation system.

The challenging task of the incremental approach is to reduce latency without affecting the model’s performance. If an input chunk is full-sentence, the model can be given important information (e.g., speech’s pronunciation for ASR, word’s inflection or word order for MT, and speech’s pronunciation or intonation for TTS). However, the incremental approach does not have available information from the next time step, and the performance is lower than the non-incremental model. Vital information differs between tasks, and each incremental model independently uses a different input chunk. Consequently, developing iASR, iMT, and iTTS systems are necessary. In this dissertation, we focus on developing the Japanese iTTS system. An iTTS system that synthesizes a speech in a shorter synthesis chunk is required to construct the incremental S2ST. To construct S2ST for the Japanese language, we proposed the Japanese incremental text-to-speech synthesis based on an accent phrase unit.

Existing iTTS technologies based on a Hidden Markov Model (HMM) [12, 13, 14] and a neural end-to-end framework [15] generate a speech from a word unit as a synthesis chunk. At the first step of iTTS, most investigations for the iTTS framework focused only on German, English, and French languages. English and German are stress-timed languages, and French is a syllable-timed language [16].

The Japanese language is a mora-timed and pitch-accent aspect language. The accent of each mora, which indicates the relative pitch change, plays an important role in prosody, which resembles tone types in tonal languages [17]. One mora is approximately equivalent to one hiragana character. The Japanese word “ha shi” can mean either a bridge or a pair of chopsticks. If the pitch changes from high to low, it means a bridge; if the pitch changes from low to high, it means a pair of chopsticks.

Figure 3 shows accent types in the Japanese language, and the standard Japanese dialect has four patterns. (1) Heibangata does not have pitch changes from a high pitch to a low pitch. (2) Atamadakagata has a high pitch for the first mora and a low pitch for others. (3) Nakadakagata has a high pitch for middle mora and a low pitch for others. (4) Odakagata has a low pitch for the first mora and a high pitch for others, and the pitch for the next mora is changed from a high pitch to a low pitch.

An accent phrase is constructed by accent sandhi. Figure 4 shows an exam-

Accent types : a blue circle represents low pitch, and an orange circle represents high pitch.

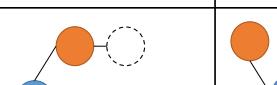
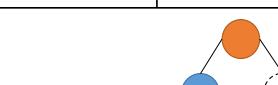
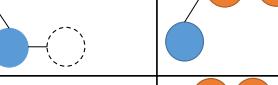
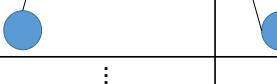
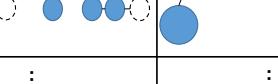
	(1) Heibangata	(2) Atamadakagata	(3) Nakadakagata	(4) Odakagata
One mora			Undefined pattern	Undefined pattern
Two moras				
Three moras				
Four moras				
⋮	⋮	⋮	⋮	⋮

Figure 3. Accent types for the Japanese language. Note that a circle represents mora units, an orange circle represents a high pitch, a blue circle represents a low pitch, and a dashed circle represents the next mora units.

ple of the Japanese accent sandhi. The second accent type is modified by the pitch of the first accent type, and one accent phrase, which consists of words, is constructed by accent sandhi. One accent phrase has only one accent type. Such pitch information is represented not in words or phonemes but in accent phrase units, which have one accent type that changes a pitch from high to low¹. The accent type is critical for representing the meaning in a given context, although it does not represent a phoneme or word sequence. Accent phrase information is used in the Japanese TTS system to represent the meaning in the context, and we must use the accent phrase information for the Japanese iTTS system. Therefore, we need the accent phrase units for the Japanese iTTS system.

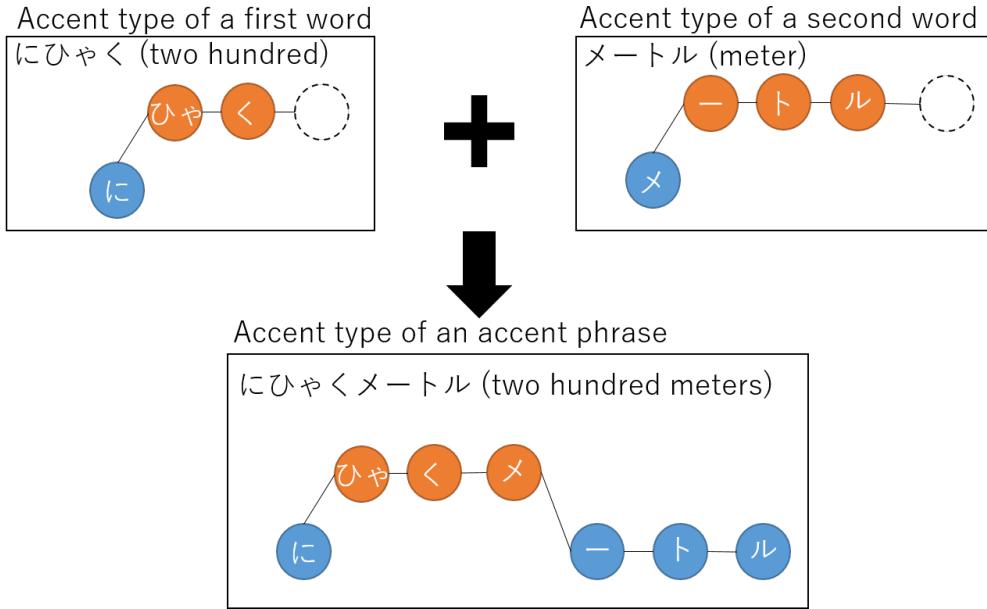


Figure 4. An example of the Japanese accent sandhi. Note that a circle represents mora units, an orange circle represents a high pitch, a blue circle represents a low pitch, and a dashed circle represents the next mora units.

¹Strictly speaking, although Heibangata and Odakagata are different, a TTS system treats them as identical.

1.2 Research Scopes and Future Directions in This Dissertation

Figure 5 shows my dissertation’s research scope and future direction. First, we introduce an HMM-based TTS system that cascades to many models. After that, we introduce an end-to-end neural TTS approach (Chapter 2). Some existing works adopt the HMM-based TTS system to an HMM-based iTTS system. However, these works focus only on the word unit as a synthesis chunk. A TTS system is a language-dependent technique, so we may consider a more extended unit than a word unit since the word unit can not aim for enough information to synthesize. Therefore, we use the Japanese language to clarify this research point. We adopt an HMM-based Japanese TTS system to an HMM-based Japanese iTTS system. We evaluate the quality and latency using a more extended unit than a word (Chapter 3).

Although the HMM-based iTTS system is standard, the quality did not reach that of human speech. There are many modules in an HMM-based TTS system. The current research topic is the neural end-to-end iTTS system for a more natural synthesis speech. The neural end-to-end architecture can simplify the model and let neural networks directly map efficiently from the input feature to the output spaces of speech acoustics. Moreover, the speech quality reaches human speech [1]. We first proposed an iTTS approach concept without waiting for the look-ahead synthesis chunk in the neural end-to-end iTTS system (Chapter 4), and we used a neural vocoder to reconstruct speech. The neural vocoder has a more powerful ability to reconstruct the speech than a vocoder based on a digital filter [18].

Recently, an end-to-end neural iTTS with a neural vocoder [15] also uses word units as a synthesis chunk, and this iTTS needs to wait for 1-2 look-ahead words to improve speech quality. However, since we do not evaluate both approaches in the Japanese language, we also investigate an accent phrase as an optimal unit in the neural end-to-end iTTS approach (Chapter 4). The evaluation of the optimal synthesis chunk of an end-to-end neural iTTS is conducted by speech quality and latency analysis in Chapter 4.

At last, we discuss future works (Chapter 5) for adopting the proposed methods to a simultaneous speech translation system.

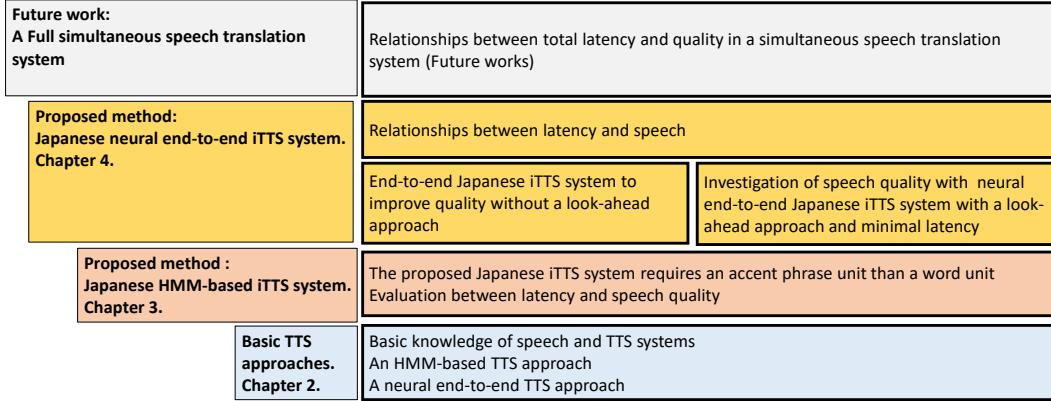


Figure 5. Relationships between research scope and this dissertation.

1.3 Contribution of this Dissertation

The following are the contributions of this dissertation.

- We proposed an HMM-based Japanese iTTs system. The proposed methods evaluate synthesized speech quality and latency of iTTs systems. Using the proposed approach, we show that an accent phrase, a longer unit than a word unit, is essential for the Japanese iTTs system (Chapter 3).
- We proposed the first concept of the neural end-to-end incremental TTS system without a look-ahead approach to investigate the relationship between the optimal synthesized speech quality and its latency for Japanese neural iTTs (Chapter 4). The proposed methods are a strategy to synthesize a speech for accent phrase units as a synthesis chunk and a strategy to use the previous information of the model. Additionally, we proposed additional input features for the Japanese neural end-to-end iTTs. We compare the proposed methods to a prefix-to-prefix end-to-end iTTs with a neural vocoder [15].

2. Speech and Text-to-speech Technology

2.1 Introduction

This chapter introduces basic knowledge of the speech (section 2.2), an HMM-based TTS system (section 2.3), a neural end-to-end neural TTS system (section 2.4), and summary of this chapter (section 2.5).

2.2 Process of Producing Speech and Acoustic Features

Figure 6 shows the production process of human speech. Speech is uttered by a sound source generation and an articulation to construct a speech pronunciation.

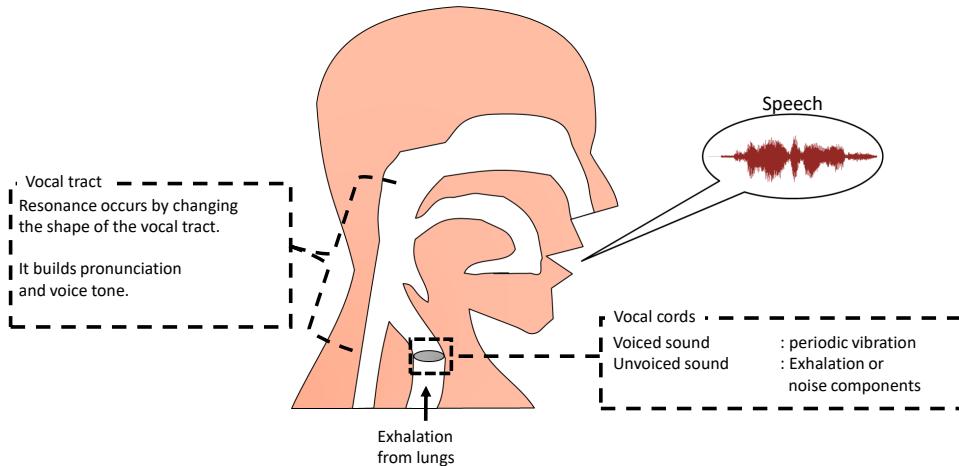


Figure 6. A producing process of human speech.

The vocal cords are vibrated by exhalation from the lungs, and sound sources are generated. Sound sources are constructed as voiced sound and unvoiced sound. Voiced sound is the periodic vibration of the vocal cords, and unvoiced sound is aperiodic components like the turbulent flow of exhaled breath. The lowest frequency in periodic vibration is called a fundamental frequency (F0). The fundamental frequency affects a pitch in sound and relates to speech prosody.

Sound sources are resonated by changing a shape of a vocal tract, and pronunciations in speech are constructed. Pronunciations are also involved in lips,

tongue position, teeth, and exhaled breath movements. These processes that construct speech are called articulation.

Acoustic features represent speech characteristics related to prosody and articulation. Acoustic features such as speech prosody use fundamental frequencies and aperiodic components. Acoustic features representing the property of articulation utilize a Mel cepstrum, a Mel generalized cepstrum [19], and a Mel spectrum. These features are analyzed by a Fourier Transformation or STRAIGHT [20].

A property of speech has language dependent. It represents an accent and a rhythm in speech. English has a stress accent that controls the prosody by the stress of the sound. In contrast, Japanese is a pitch aspect accent that controls the prosody by the relative pitch change. The rhythm is also different. The English rhythm is based on stress, and the Japanese rhythm is based on mora units [21].

Mora units are sub-unit syllables consisting of a consonant and a vowel or one vowel. In this dissertation, a mora unit uses one Japanese Hiragana character except for the contracted sound. As described in section 1.1, such pitch information is represented not in words or phonemes but in accent phrase units with mora units. An accent phrase has one accent nuclear, which changes a pitch from high to low. This is called an accent type. One accent phrase has only the accent type depending on context. The accent type is essential to represent the meaning of context, but it does not represent a phoneme sequence. The Japanese TTS system needs another input for accent phrases.

2.3 HMM-based Text-to-speech

This section describes technologies for an HMM-based TTS system. Figure 7 shows the HMM-based TTS processing outline. The upper part of Figure 7 shows the processing at the training step, and the bottom part shows the processing at the synthesizing step.

At the training step, a text processor extracts linguistic features to represent the language property of speech (section 2.3.1), and acoustic features are extracted from speech (section 2.2). A pair of linguistic features and acoustic features are used for training an HMM-based acoustic model (section 2.3.2, sec-

tion 2.3.3, and section 2.3.4).

The text processor extracts linguistic features at the synthesizing step, and HMM states are selected from linguistic features. A sequence of HMMs generates acoustic features with a generation algorithm (section 2.3.5), and a digital filter that imitates human products generates speech from acoustic features (section 2.3.6). The filter is called a vocoder.

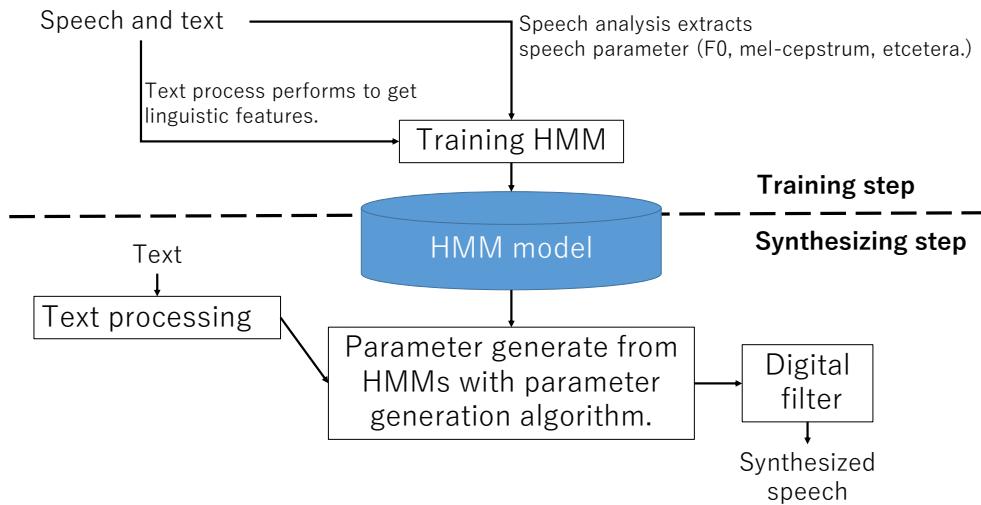


Figure 7. An overview of the HMM-based TTS system.

2.3.1 Text Processing

Language-dependent linguistic features are extracted from a text to represent language-dependent speech (section 2.2) for training a TTS system. Figure 8 shows an example of Japanese linguistic features. A phoneme is the shortest unit in linguistic features. Linguistic features consist of phoneme label sequences. Word units are piled up phoneme sequences. Accent phrase units include some words, and breath group units have some accent phrases. A breath group is detected by punctuation in a sentence. Linguistic features have hierarchical and temporal architectures. An example of temporal architecture is phonemes' forward or backward location information.

Figure 9 shows a converting process from surface text to inputs: a phoneme and an accent type in the accent phrase. The Japanese surface text has no word boundary, such as a space character in English, and a morpheme unit is useful to obtain the phoneme and the accent type. Consequently, morpheme analysis detects morphemes in the text. Each morpheme includes pronunciation, a part-of-speech tag, accent information of the morpheme, and other features like the original form of the morpheme, etc. Accent information in the morpheme includes the morpheme’s accent type and the number of moras. To obtain features for an accent phrase, pronunciations in morpheme units are reconstructed by mora units. Furthermore, the accent phrase boundary is detected by the part-of-speech tags, and the accent type of the accent phrase is obtained from rules with part-of-speech and accent information of the mora unit. Finally, pronunciations are converted to a sequence of phonemes with a pronunciation dictionary. Phonemes, part-of-speech tags, and accent types are used as inputs for the Japanese TTS. Breath group units are detected by punctuation in the sentence and are used as inputs for the Japanese TTS. Sentence features are used as inputs to count the number of words, moras, and breath groups. We use Open Jtalk² as a text processor. Open Jtalk detects accent phrase boundaries with the rules and part-of-speech (POS) tag. After detecting the accent phrase boundary, accent type estimates Sagisaka’s rule [22].

		Text: 今日は、良い天気です (It is a good weather today.)									
Sentence		Total breath group=2, Total accent phrase = 3									
Breath group		Breath group=1				Unk.	Breath group=2				
Accent phrase						Unk.					
Word (Part-of-Speech tag)		N.		PP		Unk.	Adj.	N.	Auxil.v.		
Phoneme		ky	l	o		wa	pause	y	o	i	
							t	e	n	k	i
							d	e	s	u	

Figure 8. An example of Japanese linguistic features for TTS.

²Open Jtalk – <http://open-jtalk.sourceforge.net/>

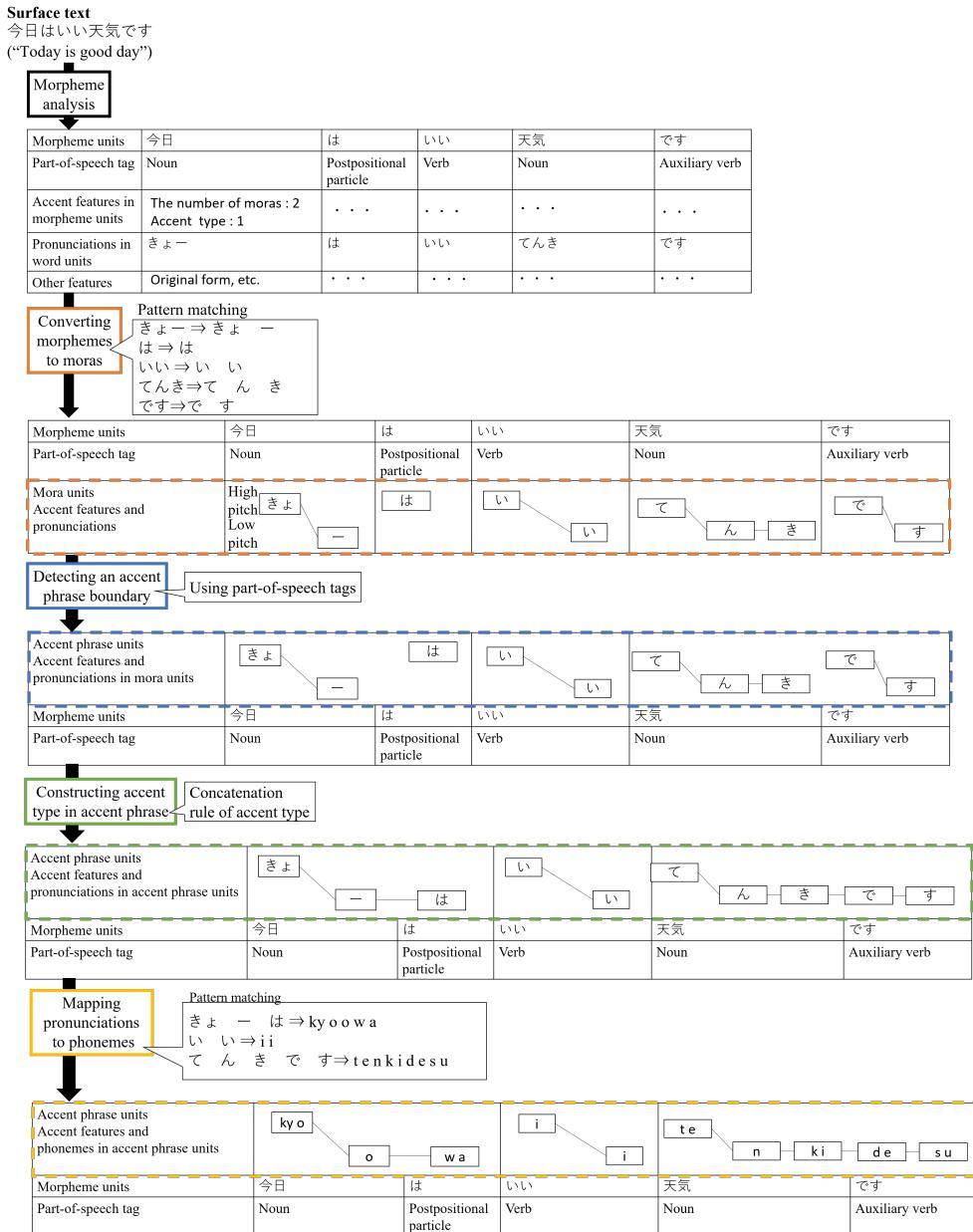


Figure 9. Flowchart of extracting accent phrase information for Japanese TTS inputs .

2.3.2 Definition of HMM

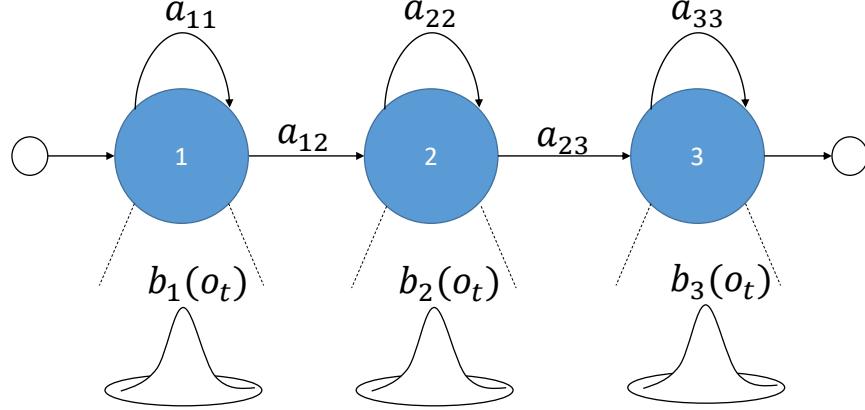


Figure 10. An illustration of a left-to-right HMM model.

Figure 10 shows a left-to-right HMM. As shown in Figure 10, the HMM is defined as the probability distribution $b_i(\mathbf{o}_t)$ to output the vector \mathbf{o}_t and the state transition probability $a_{ij} = P(q_t = j | q_{t-1} = i)$. Here, t is an index of time. i and j are the index of the HMM state, respectively. The left-to-right HMM models each variation of speech and frequency. A parameter set $\boldsymbol{\lambda}$ of N -state HMM is given by the set of transition probability $\mathbf{A} = \{a_{ij}\}_{i,j=1}^N$, the set of output probability distribution $\mathbf{B} = \{b_i(\cdot)\}_{i=1}^N$, and the set of initial state probability $\boldsymbol{\pi} = \{\pi_i\}_{i=1}^N$, that is, the parameter set $\boldsymbol{\lambda}$ of HMM is $\{\mathbf{A}, \mathbf{B}, \boldsymbol{\pi}\}$.

When a HMM state transitions to a sequence $\mathbf{q} = [q_1, q_2, \dots, q_T]$, the output probability for the sequence of output vectors $\mathbf{O} = [\mathbf{o}_1^T, \dots, \mathbf{o}_T^T]^T$ is given by the following equation:

$$P(\mathbf{O}, \mathbf{q} | \boldsymbol{\lambda}) = \prod_{t=1}^T a_{q_{t-1}q_t} b_{q_t}(\mathbf{o}_t), \quad (1)$$

where, $a_{q_0q_1} = \pi_{q_0}$ is the transition probability from the initial state to q_0 , and T is represented in the transposition of a matrix or a vector. The probability that

\mathbf{O} is output from the $\boldsymbol{\lambda}$ is represented in the following equation:

$$P(\mathbf{O}|\boldsymbol{\lambda}) = \sum_{\text{all } \mathbf{q}} P(\mathbf{O}, \mathbf{q}|\boldsymbol{\lambda}), \quad (2)$$

where “all \mathbf{q} ” is represented in all combinations of state transitions.

At the training step, the HMM parameter set $\boldsymbol{\lambda}$ is optimized by maximizing the likelihood $P(\mathbf{O}|\boldsymbol{\lambda})$, and the optimized parameter set $\hat{\boldsymbol{\lambda}}$ is given by the following equation:

$$\hat{\boldsymbol{\lambda}} = \underset{\boldsymbol{\lambda}}{\operatorname{argmax}} P(\mathbf{O}|\boldsymbol{\lambda}), \quad (3)$$

where \mathbf{O} is a set of the sequence of output vectors in training data and $\operatorname{argmax}_{\boldsymbol{\lambda}}$ is represented in an argument of the maximum for the parameter set. The learning algorithm for maximizing the likelihood can be solved by the Expectation–Maximization algorithm called the Baum-Welch algorithm.

2.3.3 Acoustic Model of HMM

To use an HMM model as an acoustic model, one HMM models one phoneme unit. The transition probability of section 2.3.2 is the duration of acoustic features, T is the total frame of acoustic features, and the output probability of section 2.3.2 is the probability of the sequence of acoustic feature vectors.

A multidimensional Gaussian distribution $b_i(\mathbf{o}_t)$ is used as an output probability of acoustic features. The probability is the following equation:

$$b_i(\mathbf{o}_t) = \mathcal{N}(\mathbf{o}_t; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}_i|}} \exp\left\{-\frac{1}{2} (\mathbf{o}_t - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{o}_t - \boldsymbol{\mu}_i)\right\}, \quad (4)$$

where D is the number of dimensions in the Gaussian distribution, and $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are the mean vector and covariance matrix of the Gaussian distribution for state i , respectively.

In an HMM-based TTS, an output vector \mathbf{o}_t is represented in a matrix and defined as $\mathbf{o}_t = [\mathbf{c}_t^T, \Delta\mathbf{c}_t^T, \Delta^2\mathbf{c}_t^T]^T$. Here, $\mathbf{c}_t = [c_t^1, \dots, c_t^{D_a}]^T$ is an acoustic feature vector with D_a dimentions. $\Delta\mathbf{c}_t$ and $\Delta^2\mathbf{c}_t$ are dynamic features [23]. Dynamic features are calculated by static feature vectors around \mathbf{c}_t with weighted coefficients for dynamic features. Dynamic features represent variation among acoustic

feature vectors, and first-order and second-order differential values among acoustic feature vectors are used as dynamic features.

For the fundamental frequency of acoustic features, voiced speech takes continuous values, and unvoiced speech indicating a symbol of silence takes discrete values. Therefore, it can not directly model the fundamental frequency using continuous distribution HMMs or discrete distribution HMMs. A Multi-Space probability Distribution HMM (MSD-HMM) is used for modeling the fundamental frequency [24]. The MSD-HMM can mix discrete and continuous distributions in the HMM and simultaneously model the fundamental frequency's continuous and discrete values.

A control for the duration of acoustic features is based on the state continuation length of the HMM. The HMM's state continuation length is modeled by using the state transition probability and the state duration distribution [25].

2.3.4 Context Clustering

After training phoneme HMM, a context-dependent HMM to consider temporal phonological information of the speech is trained by using a combination of linguistic features. The context-dependent HMM can learn a more accurate acoustic model by considering the temporal fluctuation of speech; however, the combination of linguistic features increases exponentially. Therefore, the training data of the context-dependent model decreases. Moreover, preparing training data corresponding to all linguistic feature combinations is challenging. The context-clustering technique is used to solve these problems while training the context-dependent HMM.

The context-clustering technique builds a decision tree using questions about linguistic features, and the decision tree classifies the HMM parameters. The lack of training data is prevented by sharing HMM parameters based on linguistic features and decision tree classification.

2.3.5 Generation Algorithm for Acoustic Features

A decision tree and linguistic features select the HMM parameter $\hat{\lambda}$ for generating acoustic features. A sequence of HMM states $\hat{q} = [\hat{q}_1, \dots, \hat{q}_t, \dots, \hat{q}_T]$ is

determined by a likelihood maximization and duration models. After determining HMM state sequences, statistic acoustic features are generated by constraints based on dynamic features [23]:

$$\hat{\mathbf{C}} = \underset{\mathbf{O}}{\operatorname{argmax}} P(\mathbf{O} | \hat{\mathbf{q}}, \hat{\boldsymbol{\lambda}}) \text{ subject to } \mathbf{O} = \mathbf{W}\mathbf{C}, \quad (5)$$

where \mathbf{W} represents a weighted matrix to convert $\mathbf{O} = [\mathbf{o}_1^T, \dots, \mathbf{o}_T^T]^T$ from $\mathbf{C} = [\mathbf{c}_1^T, \dots, \mathbf{c}_T^T]^T$ and $\mathbf{o}_t = [\mathbf{c}_t^T, \Delta\mathbf{c}_t^T, \Delta^2\mathbf{c}_t^T]^T$ in section 2.3.3.

When the output vector \mathbf{o}_t uses as only the statistic feature vector \mathbf{c}_t at the training step, generated statistic feature vectors $\hat{\mathbf{c}}$ are analytically known to be a sequence of mean vectors in output probabilities $[\mathbf{u}_{\hat{q}_1}^T, \dots, \mathbf{u}_{\hat{q}_T}^T]^T$. An over-smoothness, which generates a mean vector, makes synthesized speech muffled. Dybanuc feature vectors can be avoided over-smoothness.

2.3.6 Speech Generation by a Digital Filter

Generated acoustic feature vectors are reconstructed to speech using a discrete-time digital filter. Figure 11 shows an illustration of the discrete-time filter on the basis of a source-filter model. An excitation signal $e(n)$ is constructed to pulse signals for a voiced sound and white noise signals for an unvoiced sound. The excitation signal is chosen by a switch to control voiced or unvoiced sounds. The f0 values determine the period in the pulse sequence. A linear time-invariant system has an impulse response to represent a vocal tract, and speech signals $x(n)$ can be computed by the following equation:

$$x(n) = h(n) * e(n),$$

where $h(\cdot)$ is the impulse response to represent a vocal tract, and $*$ is a discrete convolution function. In the digital filter, a Mel-Los Spectrum Approximation (MLSA) filter is used as an approximation digital filter to synthesize a speech waveform from acoustic feature vectors [26], and aperiodic components are also used for representing noise components in the voiced sound.

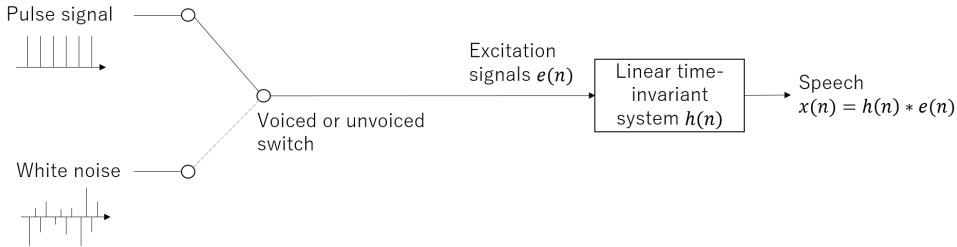


Figure 11. An illustration of a discrete-time filter on the basis of a source-filter model.

2.4 End-to-end Based Text-to-speech

This section introduces technologies about neural networks and a neural end-to-end TTS system. At the training step, a text processor extracts linguistic features, and acoustic features are extracted from speech. A pair of linguistic and acoustic features are used for training an end-to-end neural acoustic model. The one neural network model can directly map a relationship between linguistic and acoustic features. The text processor extracts linguistic features at the synthesizing step, and one neural network generates frame-wisely an acoustic feature. Synthesized speech is reconstructed from acoustic features with a vocoder on the basis of neural networks.

2.4.1 Text Processing for End-to-end Text-to-speech

Text processing is the same as the HMM-based TTS system, as described in section 2.3.1. In the HMM-based TTS system, many combinations of linguistic features on the basis of various units, such as a phoneme, a word, an accent phrase, a breath group, and a sentence, are used for training context-dependent HMM and constructing a decision tree (section 2.3.4). In the Japanese neural end-to-end TTS system, a sequence of phonemes and a sequence of accent phrase information are enough inputs to the end-to-end TTS system [27, 28, 29].

2.4.2 Definition of Neural Network Framework

This section introduces basic neural network frameworks: a linear layer, a convolution layer, a recurrent network, and combinations of these layers.

The linear layer performs an affine transformation to inputs. The linear layer is defined by following an equation:

$$\mathbf{y} = f(\mathbf{x}\mathbf{A}^T + \mathbf{b}) \quad (6)$$

where \mathbf{y} and \mathbf{x} are an output vector and an input vector, respectively. \mathbf{A} is a transforming matrix, \mathbf{b} is a vector for the bias term, and f is an activation function.

Multiple neural network frameworks use activation functions to perform non-linear transformations and represent a complex distribution. *Sigmoid*, *Tanh*, and *ReLU* functions are used as activation functions to perform the non-linear transformation. These activation functions are the following equations.

$$Sigmoid(x) = \sigma(x) = \frac{1}{1 + \exp(-x)},$$

$$Tanh(x) = \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)},$$

$$ReLU(x) = relu(x) = \max(0, x),$$

where $\exp(\cdot)$ is an exponential function, $\max(\cdot)$ is represented in a ramp function, and x is a scalar input. When the input is a vector or matrix, these functions calculate element-wise.

Convolutional Neural Network (CNN) is also one type of linear layer. The most significant difference of the linear layer is in computing an affine transformation to partial inputs. CNN utilizes location information about input and uses the same weight matrix to calculate outputs. An end-to-end TTS uses a 1D CNN with one-dimensional sequences as an input. The equation is the following:

$$a_i = \sum_c \sum_{k=0}^{K-1} w_k^c x_{i+k}^c + b_i, \quad (7)$$

where c is the size of the input channel, K is a size of kernel, w_k^c is an element of the weight matrix \mathbf{W} in a channel c , x_i^c is an input element of i in channel c , and b_i is a bias element of i in \mathbf{b} . Other parameters in CNN are stride, padding, dilation, and output channels. The stride controls the weight position to multiply an input sequence. The stride equals 1 in equation (7). The padding parameter decides the amount of padding to the input because it aligns a length between the input and output. The dilation controls a gap between kernel elements when it multiplies to an input. Output channels are a number of output channels that determine the number of weight matrices, bias terms, and output dimensions.

There are two types of CNN: a causal CNN and a non-causal CNN. Figure 12 shows an example of the non-causal CNN with one output channel. The non-causal CNN calculates the output element a_i by scanning the same weight matrix \mathbf{W} to an input sequence considering left to right. The non-causal CNN inserts padding sequences on the left and right sides to uniformize the series length. Figure 13 shows an example of causal CNN with one output channel. The causal CNN does not use the following input elements $x_{i=1,2}^c$, when calculating the output a_0 (Figure 13). Therefore, padding sequences are inserted on the left sides of the inputs.

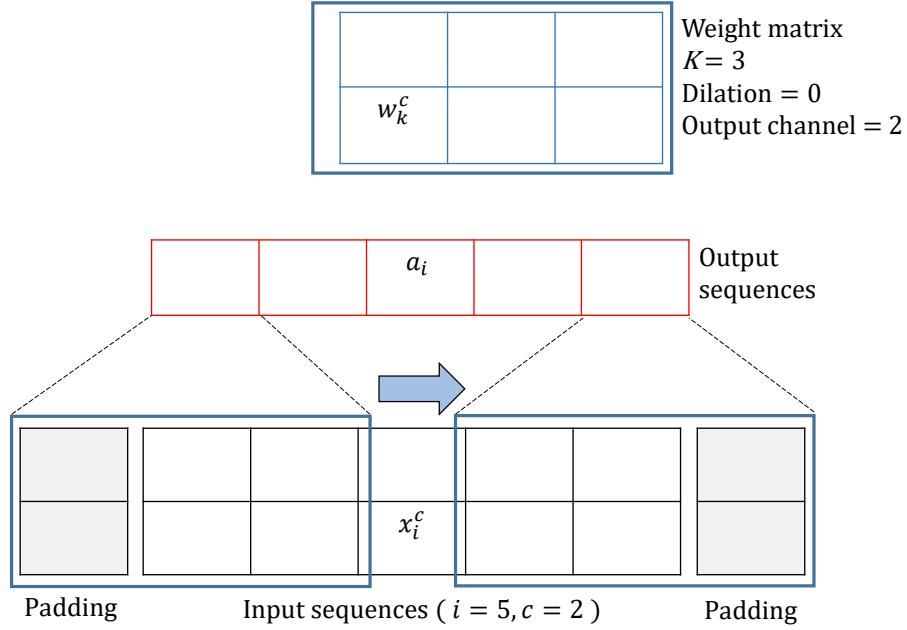


Figure 12. Visualizing a non-causal CNN.

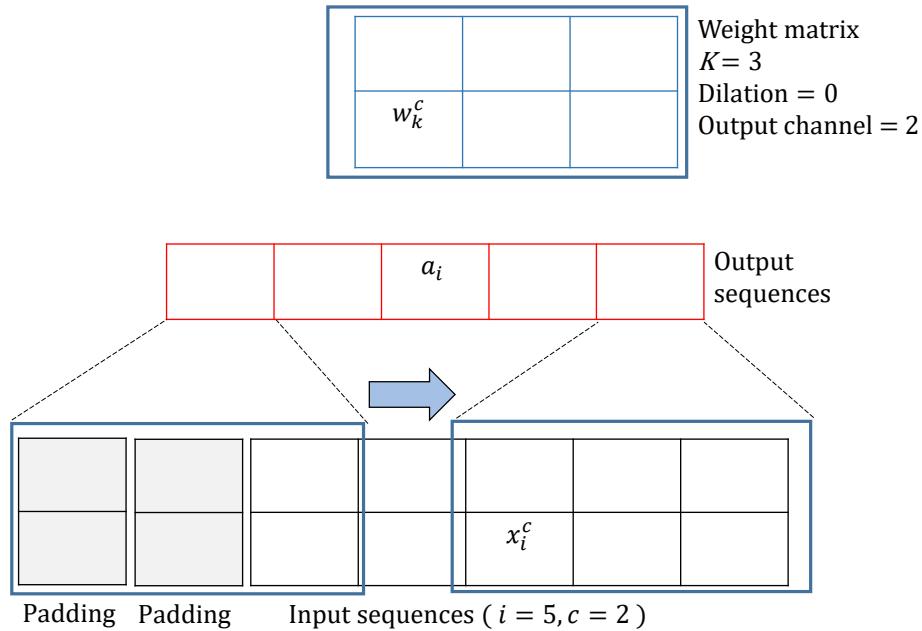


Figure 13. Visualizing a causal CNN.

Recurrent Neural Networks (RNN) model a temporal sequence. The basic RNN computes an output vector \mathbf{h}_t at time t . \mathbf{h}_t is the following equations:

$$\mathbf{h}_t = \tanh(\mathbf{W}_{ih}\mathbf{x}_t + \mathbf{b}_{ih} + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_{hh}),$$

where \mathbf{h}_t is the output vector of RNN at time t , \mathbf{x}_t is the input vector to RNN at time t , \mathbf{W}_{ih} is a transforming matrix to \mathbf{x}_t , \mathbf{W}_{hh} is a transforming matrix to the previous output vector \mathbf{h}_{t-1} , \mathbf{b}_{ih} is a vector for a bias term of the current input, and \mathbf{b}_{hh} is a vector for the bias term of the previous output state. The RNN output vector relates to all previous inputs. So RNN has a problem: no strongly related previous information is used to compute the current output. RNN should not represent a model which is not necessary for such long previous sequences.

Long Short-Term Memory (LSTM) is a type of RNN to solve it. The LSTM computes \mathbf{h}_t as the following equations:

$$\begin{aligned}\mathbf{i}_t &= \sigma(\mathbf{W}_{ii}\mathbf{x}_t + \mathbf{b}_{ii} + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_{hi}), \\ \mathbf{f}_t &= \sigma(\mathbf{W}_{if}\mathbf{x}_t + \mathbf{b}_{if} + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_{hf}), \\ \mathbf{g}_t &= \tanh(\mathbf{W}_{ig}\mathbf{x}_t + \mathbf{b}_{ig} + \mathbf{W}_{hg}\mathbf{h}_{t-1} + \mathbf{b}_{hg}), \\ \mathbf{o}_t &= \sigma(\mathbf{W}_{io}\mathbf{x}_t + \mathbf{b}_{io} + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_{ho}), \\ \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t, \\ \mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t),\end{aligned}$$

where \mathbf{h}_t is the output vector at time t . \mathbf{c}_t , \mathbf{o}_t , \mathbf{g}_t , \mathbf{f}_t , and \mathbf{i}_t vectors are called a cell state, an output gate, a cell gate, a forget gate, and an input gate, respectively, \odot is the Hadamard product. \mathbf{W} and \mathbf{b} with subscripts are transforming matrixes and vectors of bias terms for each gate. LSTM performs to forget not necessary previous sequence by each gate function. However, there are more parameters than RNN to use many gates.

Unidirectional models such as RNN and LSTM calculate outputs from \mathbf{x}_1 (initial input) to \mathbf{x}_t . We also use bi-directional models to get rich information about time sequences. The output vector in a bidirectional LSTM is calculated by a direction from \mathbf{x}_1 to \mathbf{x}_t and a direction from \mathbf{x}_t to \mathbf{x}_1 . The bidirectional LSTM has twice the parameters as the unidirectional LSTM to calculate each direction output vector. So, it needs more memory than the unidirectional LSTM.

A normalization approach is used to prevent over-fitting neural networks. A TTS uses Dropout [30], Batch Normalization [31], and Zoneout [32].

2.4.3 Attention-based Encoder-decoder for Acoustic Model

In the end-to-end neural approach, we defined the notation $x_{1:N} = [x_1, \dots, x_S]$ as the sequence of x with length S . An encoder-decoder directly models conditional probability between two sequences $p(y_{1:T}|x_{1:S}, \boldsymbol{\theta})$, where $y_{1:T} = [y_1, \dots, y_T]$ is the sequence of the output with length T , $x_{1:S} = [x_1, \dots, x_S]$ is the sequence of the input with length S , and $\boldsymbol{\theta}$ is parameter set of neural networks such as weighted matrix in all networks.

The encoder-decoder model with attention is on the basis of stacked neural networks. It has three networks in one model: an encoder, a decoder, and an attention mechanism. The encoder transforms the input sequence into another sequence of the encoder's hidden states $\mathbf{h}_{1:S} = [\mathbf{h}_1, \dots, \mathbf{h}_S]$. After getting the encoder's hidden states, the context vector \mathbf{c}_t represents the important information from $\mathbf{h}_{1:S}$ to the decoder at the decoding step t . A context vector \mathbf{c}_t and alignment weight α_t^s are computed as the following equations:

$$\alpha_t^s = \frac{\exp(score(\mathbf{h}_t, \mathbf{h}_s))}{\sum_s \exp(score(\mathbf{h}_t, \mathbf{h}_s))}, \quad (8)$$

$$\mathbf{c}_t = \sum_{s=1}^S \alpha_t^s \mathbf{h}_s \quad (9)$$

Here, \mathbf{h}_t is a decoder's hidden state which usually uses an RNN hidden state in the networks of the decoder. A *score* function is used as the following:

$$score(\mathbf{h}_t, \mathbf{h}_s) = \begin{cases} \mathbf{h}_t^T \mathbf{h}_s & dot \\ \mathbf{h}_t^T \mathbf{W}_\alpha \mathbf{h}_s & general \\ \mathbf{v}_\alpha^T \tanh(\mathbf{W}_{\alpha 1} \mathbf{h}_t + \mathbf{W}_{\alpha 2} \mathbf{h}_s) & content \end{cases} \quad (10)$$

,where T is a transposition, $\mathbf{W}_{\alpha 1}$, $\mathbf{W}_{\alpha 2}$, and \mathbf{v}_α are matrix to align the two dimensions. After calculating the context vector, y_t is estimated by the decoder network with the \mathbf{c}_t and the decoder's input.

A task of neural end-to-end TTS by using an encoder-decoder with attention models the conditional probability between $p(\mathbf{y}_{1:T}|x_{1:S}, \boldsymbol{\theta})$, where $x_{1:S} =$

$[x_1, \dots, x_S]$ is the sequence of inputs with length S and $\mathbf{y}_{1:T} = [\mathbf{y}_1, \dots, \mathbf{y}_T]$ is the sequence of the acoustic features with length T . The neural end-to-end TTS generates the sequence of acoustic features from one neural network. Acoustic features are generated frame-wise.

In the dissertation, we use Tacotron2 [1] as an architecture. Figure 14 shows the architecture of Tacotron2. The encoder has an embedding layer, three 1D CNN layers with Dropout and ReLU activation functions, and Bi-directional LSTM. The decoder has two linear layers (Pre-net) with Dropout and ReLU, two layers of Uni-directional LSTM with attention, a linear layer to project a stop flag from an LSTM output vector, a linear layer to project mel spectrum from the LSTM output vector, and five 1D CNN layers (Post-processing net) to estimate a difference of mel spectrum.

The decoding step is stopped by detecting the stop flag that 0 means not to stop the decoder, and 1 means to stop the decoder. Tacotron2 can be controlled to terminate the decoder by a stop flag.

The model parameter $\boldsymbol{\theta}$ of Tacotron2 trains from the dataset, and the training process is to decrease a value for the loss function. A loss function without a neural vocoder is the following equation:

$$\begin{aligned} Loss(\mathbf{y}_{1:T}, \hat{\mathbf{y}}_{1:T}, y_{1:T}^{stop}, \hat{y}_{1:T}^{stop}) = & \\ & \frac{1}{T} \sum_1^T \{MSE(\mathbf{y}_t, \hat{\mathbf{y}}_t)\} \\ & + \frac{1}{T} \sum_1^T \{\hat{y}_t^{stop} \log(y_t^{stop}) + (1 - \hat{y}_t^{stop}) \log(1 - y_t^{stop})\}. \end{aligned} \quad (11)$$

Here, $\hat{\mathbf{y}}_{1:T} = [\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_T]$ is the sequence of generated acoustic feature vectors $\hat{\mathbf{y}}_t$, $\hat{y}_{1:T}^{stop} = [\hat{y}_1^{stop}, \dots, \hat{y}_T^{stop}]$ is the sequence of probability for the predicted stop flag \hat{y}_t^{stop} , $y_{1:T}^{stop} = [y_1^{stop}, \dots, y_T^{stop}]$ is the sequence of the truth label of the stop flag y_t^{stop} , y_t^{stop} has a value for 0 or 1, and $MSE(\cdot)$ is a function of Mean Squared Error.

Tacotron2 trains the encoder-decoder with attention for predicting the mel spectrum and a neural vocoder for generating speech at the same time. In our experiment, we have trained the model separately due to having a limitation of GPU memory. The training method of a neural vocoder introduces later. The speech quality of Tacotron2 with phoneme inputs is reached to a speech quality of a human, and Tacotron2 indicates that the mel spectrogram is enough information

to synthesize a speech [1].

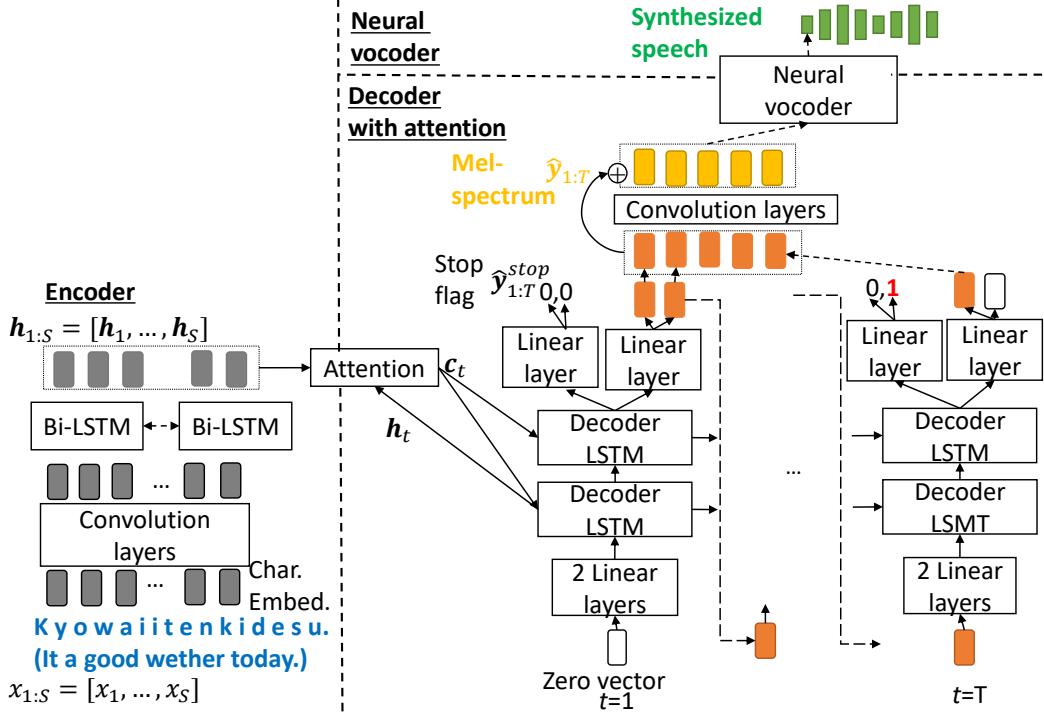


Figure 14. An architecture of Tacotron2 [1].

2.4.4 Forward Attention in Acoustic Model for Text-to-speech

An attention model can automatically learn the durations between input and output sequences at the training step. A Forward Attention [33] can get faster convergence for the attention mechanism of end-to-end TTS. The Japanese end-to-end TTS also uses it [28].

The initial weights for the Forward Attention are defined as $\hat{\alpha}_0^1 = 1, \hat{\alpha}_0^2 = 0, \dots, \hat{\alpha}_0^S = 0$. Here, $\hat{\alpha}_t^0$ is defined as 0. The new alignment weight $\hat{\alpha}_t'^s$ at a time step t is given by the following equations:

$$\begin{aligned}\hat{\alpha}_t'^s &= (\hat{\alpha}_{t-1}^s + \hat{\alpha}_{t-1}^{s-1})\alpha_t^s \\ \hat{\alpha}_t^s &= \frac{\hat{\alpha}_t'^s}{\sum_{s=1}^S \hat{\alpha}_t'^s}.\end{aligned}\tag{12}$$

Here, α_t^s is given by equation (8). The context vector for forward attention \mathbf{c}'_t is the following equation:

$$\mathbf{c}'_t = \sum_{s=1}^S \hat{\alpha}_t^s \mathbf{h}_s \quad (13)$$

We use Forward Attention with Transit Agent [33]. The agent outputs a scalar value u_t between 0 and 1 at a timestep t , and the value of u_t controls the ratio of mixing the $\hat{\alpha}_{t-1}^{s-1}$ and $\hat{\alpha}_{t-1}^{s-1}$. The transit agent uses a linear layer with the *Sigmoid* function. The following equations give attention weight to use the transit agent and u_t :

$$\begin{aligned}\hat{\alpha}'_t^s &= ((1 - u_{t-1})\hat{\alpha}_{t-1}^s + u_{t-1}\hat{\alpha}_{t-1}^{s-1})\alpha_t^s \\ u_t &= FF(\mathbf{c}'_t, \mathbf{h}_t, \hat{\mathbf{y}}_t),\end{aligned}$$

where $FF(\cdot)$ is the linear layer, \mathbf{h}_t is a decoder's hidden state vector, $\hat{\mathbf{y}}_t$ is generated acoustic feature vector, and the initial value of u_0 is 0.5.

2.4.5 Neural Vocoder for Generating a Speech

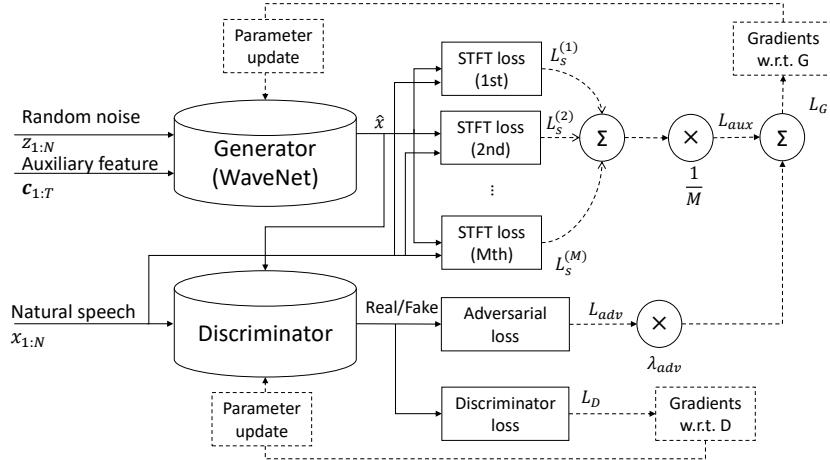


Figure 15. An architecture of Parallel WaveGAN [2].

A neural vocoder is a vocoder based on neural networks. We use a Parallel WaveGAN [34] to reconstruct a speech from generated acoustic feature sequences with Tacotron2 as the acoustic model. Figure 15. shows a process of Parallel WaveGan. Parallel WaveGan uses Generative Adversarial Networks (GAN) to represent a good distribution of a waveform without recurrent inputs. GANs consist of two neural networks: Generator (G) and Discriminator (D). Parallel WaveGan is a model conditioned on an auxiliary feature like mel spectrogram. Parallel WaveGan uses the same network architecture in Wavenet, but the model uses no non-causal CNNs instead of causal CNN. Inputs to Parallel WaveGan are a mel spectrum sequence and a white noise sequence \mathbf{z} , and a Generator parallelly synthesizes the waveform. The mel spectrogram is entered as an auxiliary feature. At this time, the mel spectrogram is used with interpolation and CNN to unify the length of the audio waveform. The Generator learns the distribution of natural speech from random noise and auxiliary features. The Generator is trained by minimizing the adversarial loss:

$$L_{adv}(G, D) = E[(1 - D(G(z_{1:N})))^2], \quad (14)$$

where $z_{1:N}$ is the sequence of white noises, $E[\cdot]$ is a function to calculate expected value, $D(\cdot)$ represents a Discriminator's process, and $G(\cdot)$ represents a Generator's process. The Discriminator trains classifications of either fake or real speech from the Generator's output and natural speech. The loss used for training the Discriminator is the following equation:

$$L_D(G, D) = E[(1 - D(x_{1:N}))^2] + E[(D(G(z_{1:N})))^2], \quad (15)$$

where $x_{1:N}$ is the sequence of a natural speech sample.

Additional loss is used as a multi-resolution STFT auxiliary loss (STFT loss in Figure 15) for improving a training process. A single STFT loss $L_s(\cdot)$ is the following equation:

$$L_s(G) = E[L_{sc}(x_{1:N}, \hat{x}_{1:N}) + L_{mag}(x_{1:N}, \hat{x}_{1:N})], \quad (16)$$

$$L_{sc}(x_{1:N}, \hat{x}_{1:N}) = \frac{\| |STFT(x_{1:N})| - |STFT(\hat{x}_{1:N})| \|_F}{\| |STFT(x_{1:N})| \|_F}, \quad (17)$$

$$L_{mag}(x_{1:N}, \hat{x}_{1:N}) = \frac{1}{N_s} \| \log |STFT(x_{1:N})| - \log |STFT(\hat{x}_{1:N})| \|_1, \quad (18)$$

where $\|\cdot\|_F$ and $\|\cdot\|_1$ are Frobenius and L_1 norms, $|STFT(\cdot)|$ is the STFT magnitude spectrum, N_s is the number of elements in the magnitude, and $\hat{x}_{1:N}$ is a sequence of a synthesized speech sample from the Generator.

Then, multi-resolution STFT loss is the summation of the STFT losses with different STFT parameter sets. Multi-resolution STFT loss is defined as follows:

$$L_{aux}(G) = \frac{1}{M} \sum_{m=1}^M L_s^{(m)}(G). \quad (19)$$

Here, M is a number of STFT losses with a different STFT parameter, such as FFT size, window size, and the dimension of magnitude spectrum. Using multi-resolution STFT loss, the Generator is made to learn the time-frequency characteristics of speech [2, 35].

The loss function to train the generator is a combination of the multi-resolution STFT loss and the adversarial loss. The equation is defined as follows:

$$L_G(G, D) = L_{aux}(G) + \lambda_{adv} L_{adv}(G, D), \quad (20)$$

where λ_{adv} is the weighted parameter for two losses.

The generator can effectively learn the natural speech distribution by optimizing the adversarial loss for the waveform domain and the multi-resolution STFT loss for the time frequency domain [2].

2.5 Summary

This chapter has introduced outlines in the section 2.1, basic knowledge of the speech (section 2.2), HMM-based TTS system (section 2.3), and neural end-to-end based TTS (section 2.4).

The HMM-based approach (section 2.3) has the following characteristics in techniques: (1) Many HMMs are used for representing a relationship between text and its acoustic features, and acoustic feature vectors are generated on HMM state units. (2) To consider speech for time-series changes, combinations of linguistic features are used for context-dependent HMMs with a decision tree clustering. (3) Output vectors consisting of statistic and dynamic acoustic feature vectors are used for the generation algorithm. (4) Synthesized speech is reconstructed from generated acoustic feature vectors with a discrete-time digital filter.

The end-to-end TTS (section 2.4) has the following characteristics compared to the HMM-based approach: (1) One neural network framework models relationships between text and its acoustic feature vectors, and the acoustic feature vector is generated by neural networks framewise. (2) The sequence of linguistic features, such as phonemes and accent phrases, are used for inputs to end-to-end TTS, and full-context labels in linguistic features are not required. (3) To generate acoustic features, dynamic and statistic features are not necessary for conditions in the neural end-to-end TTS. The framewise generation can predict detailed acoustic feature vectors. (4) Synthesized speech is reconstructed from generated acoustic feature vectors with neural networks.

3. HMM-based Japanese Incremental TTS

3.1 Introduction

Many contextual linguistic features (e.g., phoneme identity and word stress) are needed to improve synthesized speech quality because linguistic factors can represent the prosodic characteristics of speech. Specifically, in statistical parametric speech synthesis based on HMM, the following two processes are executed: (a) Text processing performs to extract linguistic features, and the process performs sentence-by-sentence; (b) HMM model is selected by linguistic features and estimating acoustic features to generate a speech waveform while performing global optimization so that acoustic features could be changed smoothly [23, 36]. Despite its ability to produce high-quality speech, a conventional TTS system can only synthesize speech sentence by sentence. It requires language-dependent contextual linguistic information of a complete sentence and parameter smoothing.

An iTTS system attempts to produce a natural-sounding speech waveform “on the fly” before receiving a complete sentence. The main challenge is estimating the target prosody online from partial knowledge of the sentence’s syntactic structure. In contrast to the process in the conventional TTS system, for part (a), the iTTS system has to extract linguistic features in a situation where some (next part-of-speech (POS) tag, the next word, et cetera.) are unknown. In processing (b), a limited HMM sequence must be constructed from a limited number of linguistic features, local optimization must be performed, and acoustic features must be estimated. Unfortunately, speech quality may deteriorate due to the limited number of linguistic features and local optimization.

3.2 Related Work of HMM-based Incremental TTS

Various studies on the HMM-based iTTS system have proposed ways to address the above problems. Baumann et al. investigated how unknown linguistic features influence prosodic estimation in English and German iTTS systems [37, 13]. Pauget et al. proposed an iTTS training strategy based on HMM with unknown linguistic features [14]. In addition, they also proposed an approach to predict the POS of the next word and use it as a linguistic feature [38]. By adopt-

ing the above approaches, iTTS quality improves. However, most investigations into the iTTS framework focus only on German, English, and French. English and German are stress-timed languages, and French is a syllable-timed language [16]. This dissertation focuses on the Japanese language, with different prosodic characteristics to German, English, and French.

3.3 Proposed Method

Figure 16 shows the different scopes of contextual linguistic features. The solid line box in the figure shows a conventional TTS system, given a complete text sentence and its linguistic information. In contrast, an iTTS system generates acoustic features for a speech waveform given only a partial text sentence (Figure 16, dashed line box).

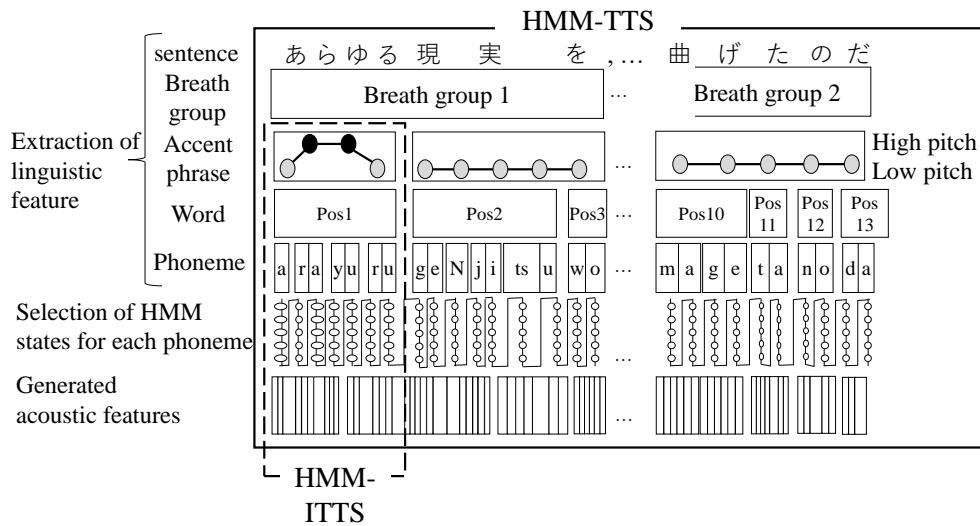


Figure 16. A differential process scopes in TTS and iTTS systems.

Table 1 shows linguistic features for standard TTS and iTTS systems. A breath group is a more extended unit than an accent phrase. A punctuation mark detects breath group into text and silence part into a speech. In iTTS systems, underlined linguistic features are the available case. For an example of iTTS with a word as the synthesis chunk, the next phoneme’s identification can be used in the current word.

The main difference between TTS and iTTS systems is the following two points. First, the sentence unit’s linguistic features cannot be used for iTTS because it requires a sentence as input.

Second, pause insertion between accent phrases is not used by the iTTS system. Since we aim to synthesize speech in real-time, we do not use linguistic features that are difficult to detect within the current time step, such as “pause insertion occurrence between accent phrases.”

Table 1. Utilization of Linguistic Features in This Dissertation.

Synthesis chunk	normal TTS	iTTS (An underlined feature used if it is available)
Phoneme	Identity of {past, current, next} phoneme.	Identity of {past, current, <u>next</u> } phoneme.
Word	{past, current, next} POS tag information of the word.	{past, current, <u>next</u> } POS tag information of the word.
Accent phrase	The relative position between an accent nucleus and a mora unit. The number of mora in {past, current, next} accent phrase. {past, current, next} accent type of the accent phrase. Pause insertion occurrence between accent phrases. {forward, backward} position of mora in the current accent phrase.	The relative position between an accent nucleus and a mora unit. The number of mora in {past, current, <u>next</u> } accent phrase. {past, current, <u>next</u> } accent type of the accent phrase. {forward, backward} position of mora in the current accent phrase.
Breath group	The number of accent phrases in the {past, current, next} breath group. Position of the accent phrase in the current breath group. {forward, backward} position of breath group in the sentence.	The number of accent phrases in the {past, current} breath group. A position of the accent phrase in the current breath group. {forward} position of breath group in the sentence.
Sentence	The number of {mora, accent phrases, breath groups} in the sentence. {forward, backward} position of {mora, accent phrase} in the sentence.	

3.3.1 Linguistic Features

For an iTTS system, the more features we use, the better the quality can be. However, this approach becomes less incremental, and the latency gets longer. Therefore, it is essential to investigate the optimum linguistic and temporal locality.

First, we classify several possible linguistic locality choices to define the level of linguistic abstraction or the granularity of the “current” chunk. These choices are as follows:

Pho: Only phoneme features.

Pho+POS: Phoneme and word POS.

Pho+Accphr: Phoneme and accent phrase.

Pho+Bre: Phoneme and breath group.

Pho+POS+Accphr: Phoneme, word POS, and accent phrase.

Pho+POS+Bre: Phoneme, word POS, and breath group.

Pho+Accphr+Bre: Phoneme, accent phrase, and breath group.

Pho+POS+Accphr+Bre: Phoneme, word POS, accent phrase, and breath group.

+Next: An underlined feature used if it is available.

As an example of a combination of language features, in the case of “Pho+POS,” linguistic features within a phoneme unit and a word unit are used, but underlined linguistic features are not used. On the other hand, in the case of “Pho+POS+Next,” the next phoneme (an underlined linguistic feature of the phoneme unit in Table 1) is used because it can use any latency. However, the next word POS isn’t used because waiting for the next word is necessary.

3.3.2 Synthesis Chunk

Next, we change the chunks for iTTS systems according to the above experimental result, investigate its speech quality via objective evaluation, and visualize acoustic features in each chunk. Furthermore, to maintain the smoothness between chunks, we investigate possible ways of connecting several chunks while synthesizing text “on-the-fly” (Figure 17). Chunks include:

ChunkCurr: Synthesize the current chunk (Figure 17 (a)).

ChunkCurr+wPast: Synthesize the current chunk by connecting one chunk of a previous one (Figure 17 (b)).

ChunkCurr+wAllPast: Synthesizing the current chunk by connecting all chunks from the previous chunk (Figure 17 (c)).

ChunkCurr+wPast+wNext: Synthesize the current chunk by connecting one chunk of a previous chunk and one chunk of the next chunk (Figure 17 (d)).

A current synthesis chunk is “Accphr-3”.

- (a) ChunkCurr
- (b) ChunkCurr+wPast
- (c) ChunkCurr+wAllPast
- (d) ChunkCurr+wPast+wNext

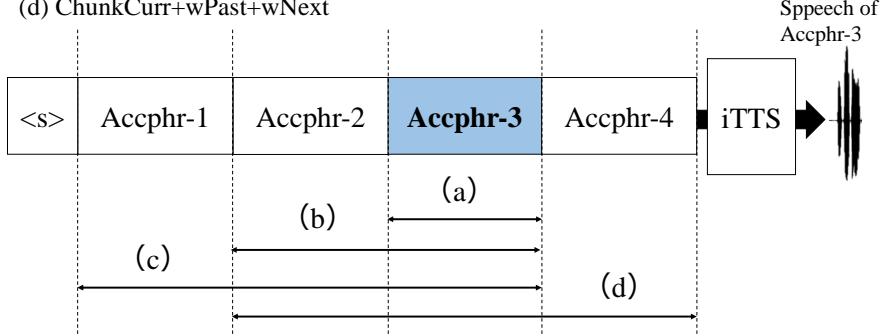


Figure 17. Possible ways of connecting several chunks while synthesizing a text. ChunkCurr shows a current synthesis chunk as an accent phrase.

3.4 Experiment

3.4.1 Dataset and Experimental Conditions

We used the ATR 503 phonetically balanced sentences [39] of the HTS demo [40] as the dataset, from which 450 sentences were used as the training set while the rest were used for the test set. The speech features include 39-dimensional Mel cepstrum coefficients, 1-dimensional fundamental frequency, 5-dimensional aperiodic components, and their respective dynamic features. We also used STRAIGHT [20] to extract speech features and the HTS engine for speech synthesis.

First, we synthesized speech using a TTS system and used the results as a reference. After that, we synthesized speech using various iTTS systems. A perceptual-based measure of the differences in terms of fundamental frequency (F0) between TTS and iTTS systems was calculated as follows:

$$C_{f_0} = \frac{1}{T} \sum_{t=1}^T 1200 \log_2 \frac{|f_{0,t}^{tar}|}{|f_{0,t}^{src}|}, \quad (21)$$

where $f_{0,t}^{tar}$ is the F0 of TTS system at t frame, $f_{0,t}^{src}$ is the F0 of iTTS system at t frame. F0 is evaluated in cent between two speeches, and 1200 cents represents

the difference of 1 octave [14]. Furthermore, we also calculated the accuracy of the estimated spectrum using a Mel cepstral distortion [41] in dB, which is defined as follows:

$$MCD = \frac{1}{T} \frac{10}{\log_{10}} \sum_{t=1}^T \sqrt{2 \sum_{d=1}^D (y_t^d - \hat{y}_t^d)^2}, \quad (22)$$

where t and d are the number of frames and the Mel cepstrum dimensions, respectively. y_t^d is a Mel cepstrum component with TTS systems. \hat{y}_t^d is a Mel cepstrum component with iTTS systems. In addition to an objective evaluation, we also conducted a mean opinion score (MOS) test as a subjective evaluation [42]. Subjects listened to each presented speech and were required to rate the overall quality of naturalness. We use a 5-point MOS scale, where five indicated “excellent” (the speech utterance sounds very clear and perfectly natural) and one indicated “bad” (the speech utterance sounds unclear and unnatural).

3.4.2 Evaluation without Next Linguistic Feature

We investigated the classes we defined in section 3.3.1 that represent linguistic abstraction or the granularity of the “current” chunk. The objective and subjective evaluations were done based on a comparison with the TTS model using full-context linguistic information. The global optimization by dynamic features was affected in these evaluations because the “current” chunks are sentences. In this experiment, generated acoustic features were considered the upper limit speech quality of a Japanese iTTS system. In other words, we assume it has a lower speech quality than that of a conventional TTS system.

Table 2 shows the F0 difference average in the objective evaluation, the MCD average in the objective evaluation, and the MOS average in the subjective evaluation. The results show that there is not much difference when we use only phonemes or phonemes with the word POS as the level of linguistic abstraction (see “Pho” vs. “Pho+POS”). However, features in an accent phrase level could improve the prosody quality (see “Pho+POS+Accphr”). As expected, the best performance was the most extended context information (see “Pho+Accphr+Bre”). Nevertheless, the MCD value of those systems is almost the same.

Table 2. MCD and f_0 error of cent unit without available next linguistic features for iTTS. Note that a synthesis chunk is sentence units, and linguistic features are used conditions of iTTS.

Combination of linguistic features	C_{f_0} [cent]	MCD[dB]	MOS
Pho	242.5	3.5	-
Pho+POS	211.2	3.5	2.25
Pho+Accphr	178.8	3.4	-
Pho+Bre	186.8	3.5	
Pho+POS+Accphr	141.1	3.4	3.16
Pho+POS+Bre	175.3	3.4	-
Pho+Accphr+Bre	83.9	3.3	3.33
Pho+POS+Accphr+Bre	84.2	3.3	-
Topline TTS	0.0	0.0	3.66

Yokomizo reported that phonemes and accent phrase features are enough to decrease F0 errors in the Japanese HMM-based TTS system [17]. The HMM-based TTS system can use the next linguistic information, and phonemes and accent phrases effectively improve speech quality. The Japanese iTTS with phonemes, part-of-speech tags, and accent phrases (“Pho+POS+Accphr”) is a lower F0 error than the Japanese iTTS with phonemes and accent phrases (“Pho+Accphr”). The Japanese HMM-based iTTS system can not use the next linguistic features, and part-of-speech tags can help to improve speech quality.

As a result of the results of Table 2, we selected three variations of contextual linguistic features (“Pho+POS,” “Pho+POS+Accphr,” and “Pho+Accphr+Bre”) and performed a subjective evaluation. Sixteen native Japanese speakers conducted the subjective evaluation. There were 45-60 speech utterances (15 utterances per system) presented in random order. Each speech utterance could be played as many times as the subjects wished.

We also included the MOS test for a TTS model with full-context linguistic information. The result of the iTTS system with only “Pho+POS” linguistic information was the lowest.

“Pho+POS” and “Pho+POS+Accphr” significantly differ by less than 1%. The

results revealed that it is challenging to construct the Japanese iTTS system that produces synthesized speech word by word. The MOS scores of iTTS with “Pho+POS+Accphr” versus “Pho+Accphr+Bre” were quite close. This result indicates that the minimum level of linguistic abstraction would be in the accent phrase. Surprisingly, the MOS scores of “Pho+POS+Accphr” and “Pho+Accphr+Bre” are not that different from the TTS system, and these systems do not have any information on the next linguistic features. However, this might be the effect of the global optimization by dynamic features. We will thus choose the accent phrase or breath group as the synthesis chunk and further investigate its effect in the next experiment.

To determine a synthesis chunk for the next experiment, we analyze the total latency of iTTS systems for a synthesizing speech. The total latency consists of three parts: (1) extracting latency is the time to extract linguistic features from an input text, (2) generating latency is the time to synthesize speech from linguistic features, and (3) playing latency is the time for speech duration in a synthesis chunk. Table 3 shows the test datasets’ average text length and average latency. The synthesis chunk selects an accent phrase, a breath group, and a sentence for the standard TTS system. Note that (1) is simulated values, and the text of the synthesis chunk converts to linguistic features with a sentence-based text processor.

The latency reduces to 2.6 seconds compared to the sentence’s and accent phrase’s latency. Therefore, an iTTS system with a chunk as an accent phrase is more suitable for real-time applications than a TTS system. Moreover, a breath group is also more suitable than the TTS system because the latency of the breath group reduces by 1.66 seconds. However, some test data do not have a breath group, so the latency is the same as the TTS system in some test data. On the other hand, in the case of an iTTS system with a chunk as the accent phrase, the dataset always contains two or more accent phrases, and the latency is always shorter than the TTS system. From this result, an accent phrase is essential from a latency viewpoint. Therefore, the next experiment evaluated a synthesis chunk as an accent phrase.

In the next experiment, we focused on an iTTS system based on the accent-phrase unit with “Pho+POS+Accphr.” We investigated the speech quality of

Table 3. Average text length [characters] and latency [seconds]. Note that extracting latency is the time to extract linguistic features from an input text, generating latency is the time to synthesize speech from linguistic features, and playing latency is the time for speech duration in a synthesis chunk. Note that extracting latency is simulated values, and the text of the synthesis chunk converts to linguistic features with a sentence-based text processor.

ChunkCurr /Linguistic features	Text length	Extracting latency	Generating latency	Playing latency
Sentence/normal TTS	20.36	0.015	0.119	4.224
Breath group /Pho+POS+Accphr+Bre	9.76	0.015	0.071	2.610
Accent phrase /Pho+POS+Accphr	3.96	0.014	0.045	1.668

Japanese iTTS temporal locality choices to define an accent phrase as the synthesis chunk. Since we synthesized text accent phrase by accent phrase, a prosody break would occur between accent phrases. After that, we connected the chunk-based synthesized speech to a synthesized sentence-based speech.

Table 4 shows the averages for F0 difference, MCD, and MOS for iTTS with only an accent phrase as the synthesis chunk. As expected, the F0 difference and MCD values are worse than those for “Pho+POS+Accphr” in Table 2.

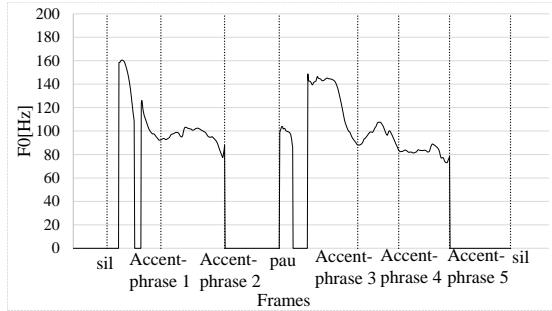
Figure 18 shows a comparison of the generated F0 sequences based on different linguistic and temporal localities: (a) F0 sequences based on full-context linguistic features (TTS); (b) F0 sequences based on “Pho+POS+Accphr” linguistic context given the sentence as a synthesis chunk; (c) F0 sequences based on “Pho+POS+Accphr” and a current accent phrase as a synthesis chunk (“Chunk Curr”). Here, “sil,” “pau,” and “accent phrase” indicate boundaries of silence, pause, and accent phrase, respectively. The results of Figure 18 indicate that smooth f0 sequences generate using a longer temporal locality (see Figure 18(a) and Figure 18(b)). However, a prosody break occurs using only the synthesis chunk as the current accent phrase (see Figure 18(c)).

The results reveal that “ChunkCurr+wPast+wNext” gave the best performance. A smoother prosodic estimate could be made among the accent phrases

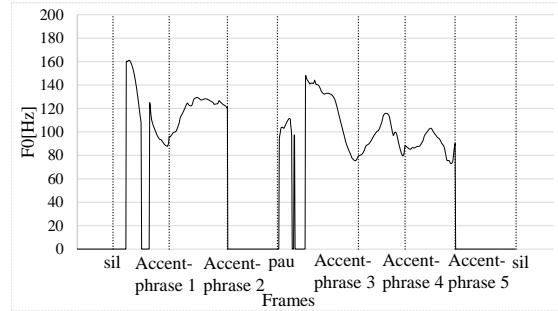
by considering the past and next accent phrase units. The subjective evaluation results show a similar tendency to the objective evaluation. Connecting all past accent phrase chunks could improve naturalness. Nevertheless, the best system is “ChunkCurr+wPast+wNext.” This result suggests waiting for one chunk before starting the synthesis process. Figure 18(d) shows f0 sequences of “ChunkCurr+wPast+wNext.” As expected, f0 sequences are also much smoother than those for iTTS systems(Figure 18(c)).

“ChunkCurr+wPast+wNext” needs to use the previous accent phrase and to wait for the next accent phrase, except for the accent phrase at the end of the sentence. The extracting latency for “ChunkCurr+wPast+wNext” would be as follows: (extracting latency for “ChunkCurr” of an accent phrase in Table 3) \times 3 = $0.014 \times 3 = 0.042$. The generating latency would be as follows: (generating latency for “ChunkCurr” of an accent phrase in Table 3) \times 3 = $0.045 \times 3 = 0.135$. The playing latency for “ChunkCurr+wPast+wNext” would be the same as “ChunkCurr” in Table 3 since a speech chunk is one accent phrase. The playing latency is 1.668 seconds. The total latency for “ChunkCurr+wPast+wNext” would be as follows: $0.042 + 0.135 + 1.668 =$ about 1.85 seconds.

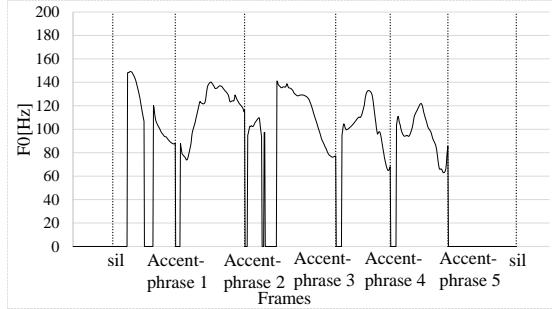
“ChunkCurr+wPast+wAllPast” uses a current accent phrase from previous accent phrases, so the latency for “ChunkCurr+wPast+wAllPast” would range from one accent phrase to the number of accent phrases in a sentence. The latency simulates the minimal latency for “ChunkCurr+wPast+wAllPast” as latency for “ChunkCurr” of an accent phrase in Table 3. The minimal latency is about 1.72 seconds. The maximum latency is simulated as follows: 5.15 (average the number of accent phrases in the test dataset) \times (extracting latency for “ChunkCurr” of an accent phrase in Table 3)+ $5.15 \times$ (generating latency for “ChunkCurr” of accent phrase in Table 3) + the playing latency for “ChunkCurr” of an accent phrase in Table 3. The maximum latency is about 1.98 seconds. Relationships between MOS and its latency are shown in Figure 19.



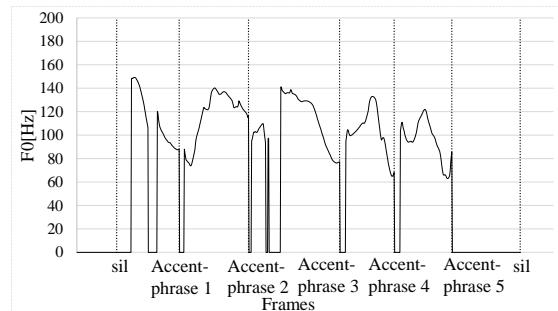
(a) F0 sequences based on full-context linguistic context (standard TTS)



(b) F0 sequences based on “Pho+POS+Accphr” linguistic context given the whole chunk sequences



(c) F0 sequences based on “Pho+POS+Accphr” linguistic context and current accent phrase chunk (“ChunkCurr”)



(d) F0 sequences based on “Pho+POS+Accphr” linguistic context. Synthesis chunk is past and next accent phrase chunks (“ChunkCurr+wPast+wNext”)

Figure 18. Comparison of generated F0 sequences based on different linguistic and temporal localities.

Table 4. Objective and subjective evaluation of iTTS systems with various chunk connections. Note that a synthesis chunk is an accent phrase

Current synthesis chunk	C_{f_0} [cent]	MCD[dB]	MOS
ChunkCurr	232.6	5.2	2.7
ChunkCurr +wPast	170.5	4.5	-
ChunkCurr +wAllPast	160.8	4.2	2.83
ChunkCurr +wPast +wNext	157.3	4.0	3.29

3.4.3 Evaluation with Next Linguistic Feature

In this section, we evaluate the effect of the next linguistic context (+Next) proposed in section 3.3.1. A dataset, a subjective evaluation method, and an objective evaluation method are the same in section 3.4.2. Ten native Japanese speakers conducted the subjective evaluation. There were 45-60 utterances, 15 utterances per system, presented in random order. Each speech utterance could be played as many times as the subjects wished. In the latency for the next available linguistic features, a text process extracts all available features in a current synthesis chunk, and we select whether to use it. For this reason, the latency for extracting available features will be the same as the input without available next linguistic features. As a result, the total latency will be the same as the latency of section 3.4.2.

First, we investigated the classes we defined in section 3.3.1 that represent the +Next or the granularity of the “current” chunk. The objective and subjective evaluations were done by comparing the TTS model with full-context linguistic features. Table 5 shows the F0 difference average and MCD average in the objective evaluation. The value in brackets shows the difference without +Next in Table 2. The results show that all cases could improve the MCD from around 0.2 to 0.5. An error of F_0 decreased using accent phrases and breath groups (see. “Pho+Accphr+Bre+next” and “Pho+POS+Accphr+Bre+Next”). In the case of “Pho+POS+Accphr”, the error of F_0 is slightly increasing (0.9↑).

Second, a subjective evaluation performs TTS systems with full-context lin-

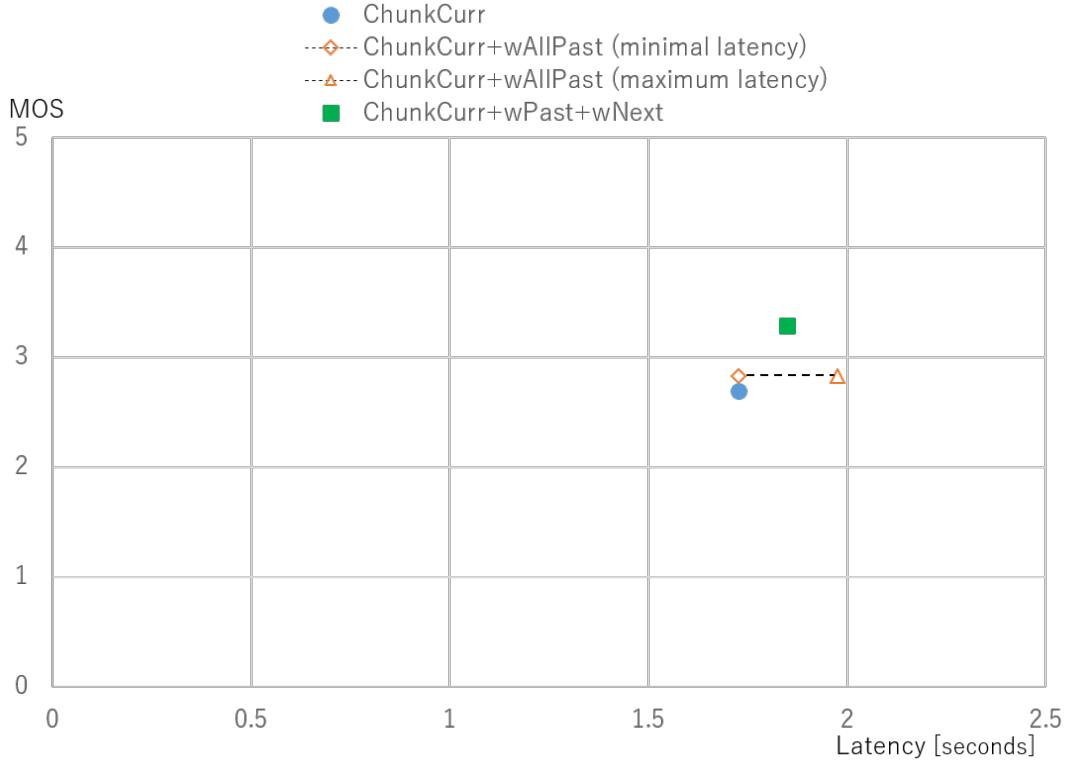


Figure 19. MOS scores of iTTS systems with various chunk connections and its latency.

Note that The total latency for “ChunkCurr+wPast+wNext” would be 1.85. “ChunkCurr+wPast+wAllPast” uses a current accent phrase from previous accent phrases,

so the latency for “ChunkCurr+wPast+wAllPast” would range from 1.72 to 1.98 seconds. Latency for ChunkCurr is an actual value (1.72 seconds). Latency for “ChunkCurr+wPast+wNext” and “ChunkCurr+wPast+wAllPast” are simulated values by the latency of “ChunkCurr.”

guistic features (a topline TTS system), “Pho+Pos+Accphr,” and “Pho+Pos+Accphr+Next.” Table 5 shows the result. The difference in MOS between “Pho+POS+Accphr” and “Pho+POS+Accphr+Next” is slight (0.03). Moreover, there is no significance. Therefore, we conclude that the available next features (+Next) are no natural improvement.

Table 5. MCD and f_0 an error of cent unit with available next linguistic features for iTTS.

Linguistic features	C_{f_0} [cent]	MCD[dB]
Pho+POS+Next	207.0 ($\downarrow 4.2$)	3.3 ($\downarrow 0.2$)
Pho+Accphr+Next	172.4 ($\downarrow 6.4$)	3.2 ($\downarrow 0.2$)
Pho+Bre+Next	182.5 ($\downarrow 4.3$)	3.0 ($\downarrow 0.5$)
Pho+POS+Accphr+Next	142.0 ($\uparrow 0.9$)	3.1 ($\downarrow 0.3$)
Pho+POS+Bre+Next	177.2 ($\uparrow 1.9$)	3.0 ($\downarrow 0.3$)
Pho+Accphr+Bre+Next	74.7 ($\downarrow 9.2$)	2.8 ($\downarrow 0.5$)
Pho+POS+Accphr+Bre+Next	73.4 ($\downarrow 10.8$)	2.8 ($\downarrow 0.5$)

Table 6. MOS score with the presence or absence of next linguistic features.

Linguistic features	MOS
Pho+POS+Accphr	3.60
Pho+POS+Accphr+Next	3.63
Topline TTS	3.84

At last, we perform a speech quality evaluation in that we have proposed a synthesis chunk with “Pho+Pos+Accphr+Next”. The synthesis chunk use sentence, “ChunkCurr,” and “ChunkCurr+Past+Next.” We used the sentence as a synthesis chunk in a TTS and the accent phrase as the synthesis chunk in iTTS systems. Moreover, as a first step, we attempted to smooth the waveforms by applying a Hanning window function when connecting waveforms of synthesis chunks. Table 7 shows the subjective and objective results. MOS improves by about 0.6 from the synthesis chunk as one accent phrase to use the next accent phrase (see, “ChunkCurr+wPast+wNext” and “ChunkCurr”). “ChunkCurr+wPast+wNext” and “ChunkCurr” have a significant (<0.001). Results mean that the next input chunk is an important approach to improve quality, and even if the iTTS system uses the next linguistic features, the speech quality could not improve. Moreover, the TTS system and the iTTS system with “ChunkCurr+wPast+wNext” have a huge MOS score difference (almost 1.0 points). Simple smoothing with the

Hanning window can improve an F0 error and an MCD value, but the Hanning window decreases the MOS score. There are some unnatural sounds between each chunk. Another problem is that it is far from the real-time application because the HMM-based iTTS system is far from the HMM-based TTS system or natural speech. Relationships between MOS and its latency are shown in Figure 20.

Table 7. Evaluation of iTTS systems with various input chunks and available next feature. A ChunkCurr uses an accent phrase. Note that “Pho+POS+Accphr+Next+smoothing” uses the Hanning window function for smoothing each waveform.

Synthesis chunk/linguistic features	C_{f_0} [cent]	MCD[dB]	MOS
Sentence/normal TTS	0.0	0.0	3.76
ChunkCurr+wPast+wNext/Pho+POS+Accphr+Next	154.8	3.71	2.81
ChunkCurr+wPast+wNext/Pho+POS+Accphr+Next+smoothing	152.8	3.52	2.67
ChunkCurr/Pho+POS+Accphr+Next	201.7	5.61	2.21

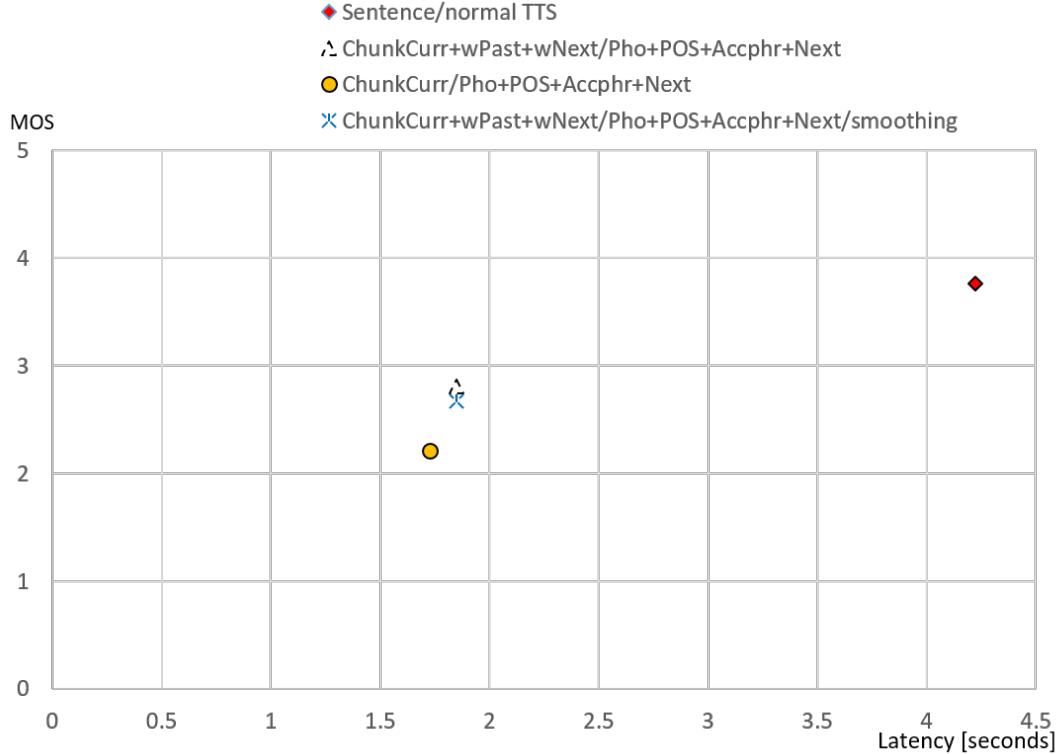


Figure 20. Relationships between MOS scores with available next features, various chunks, and its latency.

Note that latency for “ChunkCurr” and a TTS are actual values in Table 3. Latency for “ChunkCurr+wPast+wNext” is simulated values by “ChunkCurr” in Table 3.

3.5 Summary

In this chapter, we investigated a combination of linguistic features for HMM-based Japanese iTTS systems and a relationship between a synthesis chunk and its latency. We proposed an HMM-based iTTS system with accent phrase units as the synthesis chunk. Evaluations performed not only a speech quality but also a latency analysis. As a result, features of phonemes, words, and accent phrases could be training context-dependent HMMs for an HMM-based iTTS system, and accent phrase units were essential for a synthesis chunk. The quality was improved using the next accent phrase units as the synthesis chunk, and

the available linguistic features in the synthesis chunk did not improve speech quality. This result showed that using the next accent phrase was essential for the Japanese iTTS system. However, the latency using the next accent phrase unit would be huge for one accent phrase, and the latency would be longer than the breath group. For example, the latency would be the same as a sentence unit when the sentence was in two accent phrases. So we must consider improving quality without the next accent phrase.

4. Neural End-to-end Incremental TTS

4.1 Introduction

Recently, end-to-end TTS systems have been proposed [43, 1], based on a sequence-to-sequence model with attention [5]. Unlike an HMM-based TTS system, the neural end-to-end architecture simplifies models so that the neural network directly maps input features to speech or acoustic feature outputs. Developments of neural vocoders [18, 44, 2] to reconstruct speech from acoustic features or a noise sequence have also made remarkable progress. Therefore, the end-to-end architecture’s speech quality has reached the human speech level [1], and the end-to-end TTS system is better speech quality than HMM-based TTS [40, 43, 45, 46].

In this chapter, we take an initial step toward constructing a Neural iTTS and propose a quality improvement method. To the best of our contribution, this is the first study that attempts to synthesize speech in real-time using neural iTTS. We also investigated the effects of various incremental units on the quality of end-to-end neural iTTS in the Japanese language.

4.2 Related Works

Taking into account the performance improvement by end-to-end TTS systems, sequence-to-sequence modeling has also been applied to iTTS recently [47, 15, 48, 49, 50, 51]. In English, a prefix-to-prefix framework [15] was proposed that allows waits for a look-ahead of 1-2 words in iTTS. Although this prefix-to-prefix iTTS used phoneme sequences as input and produced good speech quality, it could not automatically control the look-ahead length. Another work [49] proposed a prefix-to-prefix iTTS with reinforcement learning to control the tradeoff between look-ahead words and speech quality. Other research [48] analyzed the look-ahead effects in a prefix-to-prefix iTTS approach and found that the look-ahead word length significantly affected quality. From this analysis, Stephenson et al. [50] proposed a method that predicts look-ahead text using a language model. A similar method [51] was then also proposed. These related works [15, 48, 49, 50, 51] use phoneme sequences as input features and word units for a synthesis chunk.

In a Japanese iTTS system, based on the HMM framework in the chapter 3, the input features are phonemes, words, and accent phrases. The synthesis chunk is an accent phrase unit, and the following accent phrase can improve speech quality. Since accent phrases are longer than a word, a word-based, prefix-to-prefix neural network cannot be simply applied to a Japanese iTTS system. When we apply the prefix-to-prefix neural network to the Japanese iTTS system, the look-ahead length is 1-2 accent phrases. Using 1-2 accent phrases with the look-ahead approach does not produce an unacceptable latency. We previously presented a preliminary result of a neural Japanese iTTS system [47] that uses accent phrases and phonemes without a look-ahead approach. This chapter consists of newer modeling, a deeper analysis of the synthesis unit, and comparisons of latency and quality with those of the related works. Furthermore, we propose an additional method, using various accent features in addition to accent phrases, and use Parallel WaveGan [2] and Tacotron2 [1] to improve speech quality. For the Japanese baseline prefix-to-prefix iTTS systems, we use a morpheme unit as the synthesis chunk to minimize latency and compare it to our proposed approaches. The latency and quality of the proposed iTTS systems are analyzed and compared to the results of prefix-to-prefix iTTS systems.

4.3 End-to-end Japanese ITTS Inputs

As described in section 1.1, the Japanese language is a pitch-accent aspect language. Using features in the accent phrase for Japanese end-to-end TTS systems is known to improve speech quality [27, 29, 52]. Therefore, we use a sequence of phonemes, and accent phrase features as inputs for the neural end-to-end iTTS system. Features of the accent phrase are assigned to each phoneme and defined on the mora unit. Therefore, the same feature of the accent phrase will be assigned to each phoneme. Figure 21 shows accent phrase features of “kyo o wa” (“Today is”) on mora units in the accent phrase. We use five features in the accent phrases. A1 is the difference between the position of the phoneme in mora units and the positions of the accent type. For example, A1 of the phoneme “o” in the second mora unit is 1 because the difference between the position of the phoneme “o” in the mora unit and the mora’s position of the accent type is $2 - 1 = 1$. We expect A1 to increase the number of features related to the accent

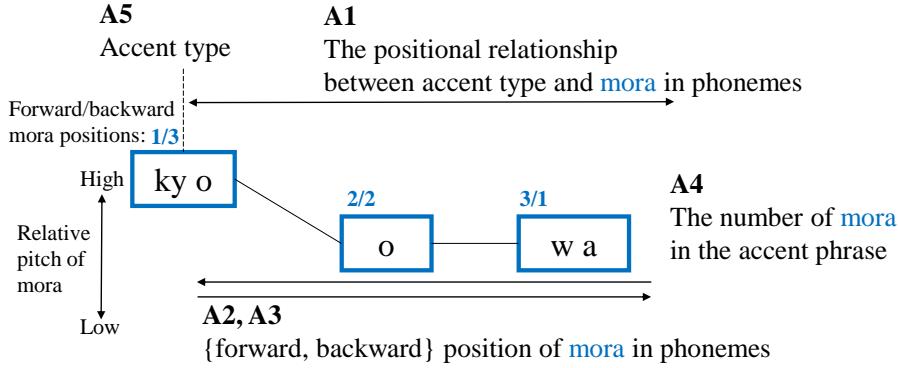


Figure 21. Japanese accent features in an accent phrase

type and mora units. A2 and A3 are the forward and backward positions of the mora in the accent phrase, that is, A2 of the phoneme “o” in the second mora unit is 2, and A3 of the phoneme “o” in the second mora unit is also 2. A4 is the number of moras in the accent phrase, and A5 is the accent type of the accent phrase.

4.4 Methods

This chapter deals with the neural TTS system, which consists of two steps: a neural encoder-decoder model to infer acoustic features from input sequences and a neural vocoder to synthesize speech from acoustic features.

The baseline prefix-to-prefix iTTS system uses a word unit as the synthesis unit [15]. As described in section 2.3.1, a morpheme unit is useful in obtaining inputs for Japanese TTS systems. In later experiments, the Japanese baseline prefix-to-prefix iTTS system used morpheme units as the synthesis chunk instead of word units.

4.4.1 Sentence-based TTS

A sentence-based TTS system processes a text sentence by sentence; a synthesis chunk is a full sentence. A full sentence containing N words is represented with a sequence of words $x_{1:N} = [\bar{x}_1, \dots, \bar{x}_N]$, where the word $\bar{x}_\tau = [x_\tau^1, \dots, x_\tau^{s_\tau}]$ includes

A baseline TTS and a baseline prefix-to-prefix iTTS
Relationship between chunks and input sequences

Chunk of $x_{1:N}$ word (morpheme) units	\bar{x}_1	\bar{x}_2	...	\bar{x}_τ	$\bar{x}_{\tau+1}$...	\bar{x}_N
Elements of \bar{x}_t	[ky, o]	[w, a]	[i, i]	[h, i]	[n, i]	[n, a, r, i]	[m, a, s, u]
Indexes of \bar{x}_t	$x_1^1 x_1^{s_1}$	$x_2^1 x_2^{s_2}$...	$x_\tau^1 x_\tau^{s_\tau}$	$x_{\tau+1}^1 x_{\tau+1}^{s_{\tau+1}}$...	$w_N^1 \dots w_N^{s_N}$

Proposed Japanese iTTS

Relationship between chunks and input sequences

Chunk of $x'^{1:M}$ accent phrase units	\bar{x}'_1	...	\bar{x}'_τ	\bar{x}'_M
Elements of \bar{x}'_t	[ky, o, w, a]	[i, i]	[h, i, n, i]	[n, a, r, i, m, a, s, u]
Indexes of \bar{x}'_t	$x'_1^1 \dots x'_1^{r_1}$	$x'_\tau^1 \dots x'_\tau^{r_\tau}$	$x'_M^1 \dots x'_M^{r_M}$	

Figure 22. Synthesis chunks and their elements for a sentence-based TTS, a prefix-to-prefix iTTS, and the proposed accent-phrase-based iTTS

an input sequence of phonemes with a length s_τ ³.

The encoder transforms the input sequence into another feature sequence as hidden states $\mathbf{h}_{1:N} = Enc(x_{1:N}) = [\bar{\mathbf{h}}_1, \dots, \bar{\mathbf{h}}_N] = [\mathbf{h}_1^1, \dots, \mathbf{h}_1^{s_1}, \dots, \mathbf{h}_N^1, \dots, \mathbf{h}_N^{s_N}]$, where $Enc(\cdot)$ represents the encoder's process.

After getting the encoder's hidden states, the decoder infers acoustic features. The chunk of acoustic features for the word $\bar{\mathbf{y}}_\tau = [\mathbf{y}_\tau^1, \mathbf{y}_\tau^2, \dots]$ is estimated by $\mathbf{h}_{1:N}$ and $\mathbf{y}_{<\tau}$, where $\mathbf{y}_{<\tau} = [\bar{\mathbf{y}}_1, \dots, \bar{\mathbf{y}}_{\tau-1}]$ are sequences of acoustic features until the previous words. More specifically, the i -th frame for any word is as follows:

$$\mathbf{y}_t^i = Dec(\mathbf{h}_{1:N}, \mathbf{y}_{<\tau} \circ \mathbf{y}_{\tau, < i}), \quad (23)$$

where $Dec(\cdot)$ denotes the decoder's process, $\mathbf{y}_{\tau, < i} = [\mathbf{y}_\tau^1, \dots, \mathbf{y}_\tau^{i-1}]$, and \circ is the concatenation of two sequences. Finally, the sentence's speech waveform $w_{1:N} = [\bar{w}_1, \dots, \bar{w}_N] = [w_1^1, w_1^2, \dots]$ is as follows:

$$w_{1:N} = \phi(\mathbf{y}_{1:N}), \quad (24)$$

where $\phi(\cdot)$ represents the neural vocoder's process, $\mathbf{y}_{1:N}$ are acoustic features of the full-sentence.

³Note that τ represents a step for the iTTS process on the basis of the synthesis chunk.

4.4.2 Word-based iTTS

Unlike a sentence-based TTS system, an iTTS system uses a partial synthesis chunk instead of a full sentence. We use a prefix-to-prefix iTTS approach as a baseline, where the synthesis chunk is one word (top table in Figure 22). The prefix-to-prefix iTTS approach uses a look-ahead approach to account for the following speech changes. The look-ahead approach waits for k words before the encoder process. The look-ahead length is determined by the following function:

$$g(\tau) = \min\{\tau + k, |\overline{x_{1:N}}|\}, \quad (25)$$

where $|\overline{x_{1:N}}|$ indicates the total number of words in the sentence.

Under the condition of the look-ahead approach, the sequence of hidden states for the word is represented by $\mathbf{h}_{1:g(\tau)} = Enc(x_{1:g(\tau)}) = [\overline{\mathbf{h}_1}, \dots, \overline{\mathbf{h}_{g(\tau)}}]$. In other words, the sequence of hidden states is conditioned by the $g(\tau)$ words. Therefore, the i -th acoustic feature for the word and the speech waveform of the word are as follows:

$$\mathbf{y}_\tau^i = Dec(\mathbf{h}_{1:g(\tau)}, \mathbf{y}_{<\tau} \circ \mathbf{y}_{t,<\tau}), \quad (26)$$

$$\overline{w_\tau} = \phi(\overline{\mathbf{y}_\tau}). \quad (27)$$

4.4.3 Proposed Accent-phrase-based iTTS

As described earlier, an accent phrase is important for representing Japanese intonation and meaning. Therefore, we propose the Japanese iTTS system on the basis of the accent phrase as the synthesis chunk. In contrast to subsection 4.4.2, the full sentence containing M accent phrases is represented with a sequence of the accent phrase $\mathbf{x}'_{1:M} = [\overline{\mathbf{x}'_1}, \dots, \overline{\mathbf{x}'_M}]$, where the accent phrase $\overline{\mathbf{x}'_\tau} = [\mathbf{x}'_\tau^1, \dots, \mathbf{x}'_\tau^{r_\tau}]$ includes an input sequence of phonemes with length r_τ and a sequence of accent features (bottom table in Figure 22). We propose two methods to estimate acoustic features for the accent phrase.

The first method is **dec+in**. The encoder’s hidden states of the accent phrase are observed:

$$\overline{\mathbf{h}'_\tau} = Enc(\overline{\mathbf{x}'_\tau}) = [\overline{\mathbf{h}'_\tau^1}, \dots, \overline{\mathbf{h}'_\tau^{r_\tau}}]. \quad (28)$$

Acoustic features in the accent phrase $\overline{\mathbf{y}'_\tau} = [\mathbf{y}'_\tau^1, \mathbf{y}'_\tau^2, \dots]$ are estimated by $\overline{\mathbf{h}'_\tau}$ and the last acoustic feature for the previous accent phrase $\mathbf{y}'_{\tau-1}^p$. The i -th acoustic

feature for the accent phrase is the following:

$$\mathbf{y}'^i_\tau = Dec(\overline{\mathbf{h}'_\tau}, \mathbf{y}'^{ip}_{\tau-1} \circ \mathbf{y}'_{\tau,< i}), \quad (29)$$

where $\mathbf{y}'_{\tau,< i} = [\mathbf{y}'^1_\tau, \mathbf{y}'^2_\tau, \dots, \mathbf{y}'^{i-1}_\tau]$. The $\overline{\mathbf{h}'_\tau}$ does not use the hidden vectors in the previous accent phrase $\overline{\mathbf{x}'_{\tau-1}}$.

Figure 23 (a) shows how **dec+in** functions in the Japanese iTTS system. The first accent phrase starts from the beginning of the sentence, and other accent phrases start from its middle. We set the initial decoder's inputs as the Mel spectrogram's last frame from the previous accent phrase.

The second method, **dec+in+hidden**, connects not only the last acoustic feature but also the previous states of the model to the current states of the model (Figure 23 (b)). Therefore, the encoder's hidden states and the acoustic feature are as follows:

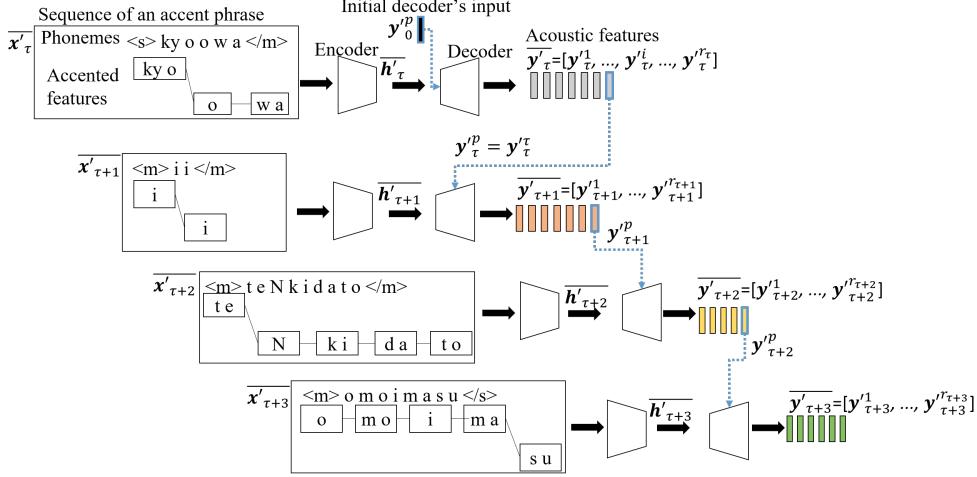
$$\overline{\mathbf{h}'_\tau} = Enc(\mathbf{x}'_{1:\tau-1} \circ \overline{\mathbf{x}'_\tau}), \quad (30)$$

$$\mathbf{y}'^i_\tau = Dec(\overline{\mathbf{h}'_\tau}, \mathbf{y}'_{<\tau} \circ \mathbf{y}'_{\tau,< i}), \quad (31)$$

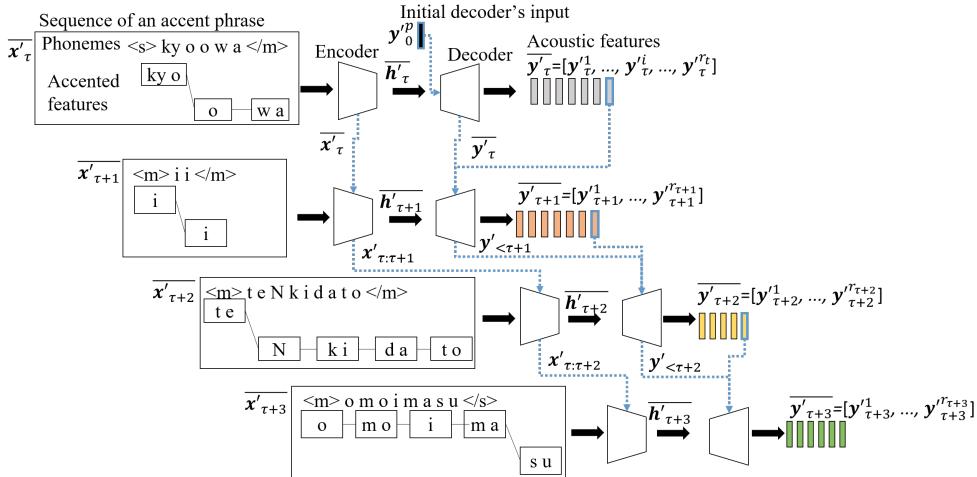
where $\mathbf{y}'_{<\tau} = [\overline{\mathbf{y}'_1}, \dots, \overline{\mathbf{y}'_{\tau-1}}]$. $\mathbf{y}'^{ip}_{\tau-1}$ in (29) and $\mathbf{y}'_{<\tau}$ in (31) are different due to encoding processes in (28) and (30). In using the second method, we expect it to learn the acoustic feature time-series and the model's internal state change.

Finally, the speech waveform of the accent phrase is the following:

$$\overline{w_\tau} = \phi(\overline{\mathbf{y}'_\tau}). \quad (32)$$



(a) **dec+in:** connecting the last synthesis vector to the decoder's initial input



(b) **dec+in+hidden:** connecting not only the last synthesis vector but also the model's internal states to the encoder and decoder

Figure 23. Proposed approaches to the Japanese iTTS system

4.5 Experiment

4.5.1 Dataset and Models

We used the JSUT dataset (version 1.1), which has 7,696 sentences (10 hours of audio sampled at 48-kHz, which we down-sampled to 22.05-kHz) spoken by a single native female speaker [53]. The data were divided into 7,196 pairs (speech and input sequences) for training, 250 pairs for the development set, and 250 pairs for the test set. We used Open Jtalk⁴ for extracting the phoneme and accent features from the text and files with speech duration⁵. We used a Geforce RTX TITAN with a memory of 24 gigabytes.

The acoustic features were extracted by Fourier transform, and our final set was composed of 80 dimensions of log Mel spectrogram features. The size of the Fourier transform was 2,048 points. The frameshift and frame lengths were 10 and 50 milliseconds, respectively. We used Tacotron2 [1] to estimate acoustic features from inputs and a Parallel WaveGan [2] to reconstruct speech from the acoustic features. Unlike the original Tacotron2, we used a uni-directional LSTM to connect hidden states of the model and Forward Attention with Transit Agent [33] to quickly converge the attention. We used an Adam [54] optimizer with a 32-batch size. The learning rate was 1e-3.

We used the prefix-to-prefix model as a baseline iTTS. The input feature is a phoneme sequence (**Pho**). To accommodate such pitch information, we used two input types of Japanese iTTS to improve speech quality. **Pho+AccType** uses phonemes and only accent types (A5) in accent phrases. We used two embedding layers for the phonemes and the accent types. Then we concatenated two embedding outputs as single input. **Pho+AccFeats** uses phonemes and both accent types (A5) and many accent features (A1, A2, A3, and A4) in the accent phrases. We used six embedding layers for the phonemes and many accent features and concatenated the embedding outputs as one input.

⁴Open Jtalk – <http://open-jtalk.sourceforge.net/>

⁵<https://github.com/r9y9/jsut-lab>

Table 8. Input feature types and embedding dimensions

Input feature types	Input and embedding dimensions
Pho	Phoneme feature dimension: 44, embedding dimension: 512
Pho+AccType	Phoneme feature dimension: 44 (TTS) or 46 (iTTS), embedding dimension: 480 A5-feature dimension: 23, embedding dimension: 32
Pho+AccFeats	Phoneme feature dimension: 44 (TTS) or 46 (iTTS), embedding dimension: 432 A1-feature dimension: 26, embedding dimension: 16 A2-feature dimension: 20, embedding dimension: 16 A3-feature dimension: 20, embedding dimension: 16 A4-feature dimension: 23, embedding dimension: 16 A5-feature dimension: 23, embedding dimension: 16

Table 8 shows the size of each embedding layer in our experiment. The first column indicates input features, and the second indicates the size of each embedding layer. We replaced low-frequency input features with an unknown symbol using a threshold to deal with unknown inputs. In Japanese iTTS, the vocabulary size is increased by two due to the special characters that indicate the middle.

Baseline TTS and iTTS systems were trained in sentence-based units. The speech was synthesized using each synthesis chunk by adding location symbols to differentiate the unit’s location: <s> is the sentence’s start and </s> is the sentence’s end. The terminating process of the decoder differs in each model. The decoding process in the TTS is controlled by the stop flag[1]. The iTTS uses the stop flag and the alignment distribution to stop the decoder in order to synthesize Mel spectrogram frames[15].

On the other hand, our proposed method was trained on the accent-phrase-based units, and the decoding process was controlled only by the stop flag. The speech was synthesized using the accent phrase by adding location symbols to differentiate the unit’s location: <s>, </s>, <m> is the middle sentence’s start, and </m> is the middle sentence’s end. When we used **dec+in**, we connected only the Mel spectrogram to each synthesis chunk. When we used

dec+in+hidden, we connected not only the Mel spectrogram but also the RNN hidden states on each synthesis chunk.

4.5.2 Evaluation Indexes

We used natural speech as a reference in our objective evaluation of speech quality. Then, we synthesized the speech using various iTTS systems. We calculated a perceptual-based measure in terms of the fundamental frequency (F0) between synthesized speech as follows:

$$C_{f_0} = \frac{1}{T} \sum_{t=1}^T 1200 \log_2 \frac{|f_0^{tar}|}{|f_0^{src}|}, \quad (33)$$

where f_0^{tar} is F0 of synthesized speech with a topline sentence-based TTS, f_0^{src} is F0 of synthesized speech with target systems, and 1200 cents represents a difference of 1-octave [14]. We also calculated the accuracy of the estimated spectrum using Mel cepstrum distortion [41] in dB, defined as follows:

$$MCD = \frac{1}{T} \frac{10}{\ln(10)} \sum_{t=1}^T \sqrt{2 \sum_{d=1}^D (y_t^d - \hat{y}_t^d)^2}, \quad (34)$$

where t and d are the number of frames and the Mel cepstrum dimensions, respectively. y_t^d is a Mel cepstrum component with a TTS system. \hat{y}_t^d is a Mel cepstrum component with an iTTS system.

In our subjective evaluation of speech quality, we calculated a mean opinion score (MOS) test [42] for the naturalness of changing lengths of incremental units. Subjects listened to each presented bit of speech audio and rated the overall quality based on its naturalness. We used a 5-point MOS scale, where 5 indicated excellent speech utterances (very clear and completely natural), and 1 indicated bad speech utterances (unclear and completely unnatural). We conducted subjective evaluations in Japanese with 13 native speakers. Synthesized speech samples were evaluated from 10 speech utterances per model.

We used a re-speaking system to analyze latency that synthesized the same speech after playing natural speech with each chunk. We measured the latency time from starting an input sequence to finishing each bit of synthesis speech. Then we calculated the frequency of latency in each model. The input sequence is

prepared in advance, and the analysis does not include latency for a text process. An incremental text process is the future direction, and We will work for it in future works.

4.5.3 Objective Evaluation of Methods

As described above, we used two types of input features in Figure 21 and a proposed approach that uses the previous decoder’s input and hidden states of a model. In this section, we evaluate the differences in the input types and the effectiveness of the models. The iTTS’s speech quality is objectively evaluated for the differences in input and method.

Table 9 shows the results of the objective evaluations with methods, input features, models, and synthesis chunks. We made four observations regarding the Japanese iTTS. First, the objective evaluation in F0 and MCD demonstrated that the proposed iTTS system approaches the sentence-based TTS (see (3-3) or (3-4)). Second, the proposed Japanese iTTS systems with **dec+in+hidden** are more efficient than word-based iTTS systems that we reimplemented ourselves. Third, regarding the different input types in the accent phrase, **Pho+AccFeats** is more efficient than **Pho+AccType**, that is, the speech quality is improved by using accent types and position of mora units in accent phrases. Input features in accent types and mora units are well represented in the Japanese language property as a tonal-aspect accent and a mora-timed rhythm to the Japanese iTTS systems. Finally, our proposed method **dec+in+hidden** is better than **dec+in**, that is, using a large amount of previous information is efficient.

Moreover, increasing the number of input features related to the accent phrase can improve speech quality with the Japanese iTTS systems using the same accent phrase unit (see (2-3) and (3-3), or (2-4) and (3-4)). The Japanese iTTS system with two accent phrases using many accent phrase features has better speech quality than one accent phrase; the longer synthesis chunk with many accent phrase features can improve speech quality (see (3-3) and (3-4)).

Table 9. Objective evaluations of proposed methods. Note that a full sentence in a column of the synthesis chunk means a sentence unit as the synthesis chunk, 1 morpheme+look- k means one morpheme as the synthesis chunk with a look-ahead of k length, and 1 accent phrase and 2 accent phrases mean one accent phrase as the synthesis chunk and two accent phrases as the synthesis chunk.

Methods	Models	Synthesis chunk	F0 error [cent]	MCD [dB]
Input type: Pho				
(1-1): Baseline TTS	Sentence-based TTS	Full sentence	259.71	5.76
(1-2): Baseline prefix-to-prefix iTTS	Word-based iTTS	1 morpheme+look-1	302.31	6.12
(1-3): Baseline prefix-to-prefix iTTS	Word-based iTTS	1 morpheme+look-2	300.00	6.13
(1-4): Baseline prefix-to-prefix iTTS	Word-based iTTS	1 morpheme+look-3	293.55	6.08
(1-5): Baseline prefix-to-prefix iTTS	Word-based iTTS	1 morpheme+look-4	286.47	6.03
(1-6): Baseline prefix-to-prefix iTTS	Word-based iTTS	1 morpheme+look-5	289.98	6.00
(1-7): Baseline prefix-to-prefix iTTS	Word-based iTTS	1 morpheme+look-10	268.17	5.85
Input type: Pho+AccType				
(2-1): Baseline TTS	Sentence-based TTS	Full sentence	207.04	5.57
(2-2): Japanese iTTS	Accent phrase-based iTTS with dec+in	1 accent phrase	300.14	6.30
(2-3): Japanese iTTS	Accent phrase-based iTTS with dec+in+hidden	1 accent phrase	252.27	6.04
(2-4): Japanese iTTS	Accent phrase-based iTTS with dec+in+hidden	2 accent phrases	266.56	5.98
Input type: Pho+AccFeats				
(3-1): Topline TTS	Sentence-based TTS	Full sentence	0.00	0.00
(3-2): Japanese iTTS	Accent phrase-based iTTS with dec+in	1 accent phrase	261.57	6.23
(3-3): Japanese iTTS	Accent phrase-based iTTS with dec+in+hidden	1 accent phrase	225.43	5.92
(3-4): Japanese iTTS	Accent phrase-based iTTS with dec+in+hidden	2 accent phrases	208.43	5.70

4.5.4 Objective Evaluation of Relationship between Speech Quality and Latency

An accent phrase is longer than a morpheme. Therefore, we must compare the latency of the baseline iTTS and our proposed iTTS before subjectively evaluating our proposed method. We analyzed the latency of the iTTS models with five methods: (1-2), (1-3), (1-4), (3-3), and (3-4).

Figure 24 shows latencies and their frequencies. The latencies of the proposed iTTS were 0.655 seconds with one accent phrase and 1.20 seconds with two accent phrases. While the latencies of the baseline methods are 0.572 seconds with a look-ahead of two morphemes and 0.667 seconds with a look-ahead of three morphemes. The latency of the proposed method with one accent phrase is slightly slower but has lower F0 error and MCD.

4.5.5 Subjective Evaluation of Naturalness

Next, we conducted a MOS test as a subjective evaluation of naturalness under six experimental conditions: a baseline iTTS with one morpheme as a synthesis chunk and no look-ahead approach (1-0), a baseline prefix-to-prefix iTTS with one morpheme as a synthesis chunk and a look-ahead-1 (1-2), a baseline prefix-to-prefix iTTS with one morpheme as a synthesis chunk and a look-ahead-2 (1-3), a topline TTS (3-1), our proposed iTTS with one accent phrase as a synthesis chunk (3-3), and our proposed iTTS with two accent phrases as a synthesis chunk (3-4). Method (1-0) used one morpheme as a short sentence with the baseline TTS model.

The subjective evaluation results and each average latency are shown in Figure 25. The baseline iTTS that used each morpheme as a short sentence showed a lower quality than the baseline prefix-to-prefix iTTS systems since that model did not use look-ahead inputs [15]. The best model was the topline TTS with all phonemes and accent features in the sentence; the latency was 4.63 seconds and longer than the others. Although the objective results showed that the proposed iTTS system approached the level of the topline TTS, the subjective results of the proposed iTTS showed room for improvement. The intonation between accent phrases might attract strong attention from the evaluators, or the Japanese iTTS system might need more comprehensive information. The baseline prefix-to-prefix iTTS approach additionally proposed multiple words as an input chunk instead of one word. The prefix-to-prefix iTTS approach with phonemes assumes that the quality of the sentence-based TTS approach with phonemes is the upper bound. The approach does not consider the features of accent phrase units, and our proposed iTTS systems can have a better objective result than the sentence-based TTS with phonemes. As a result, our proposed methods would produce higher

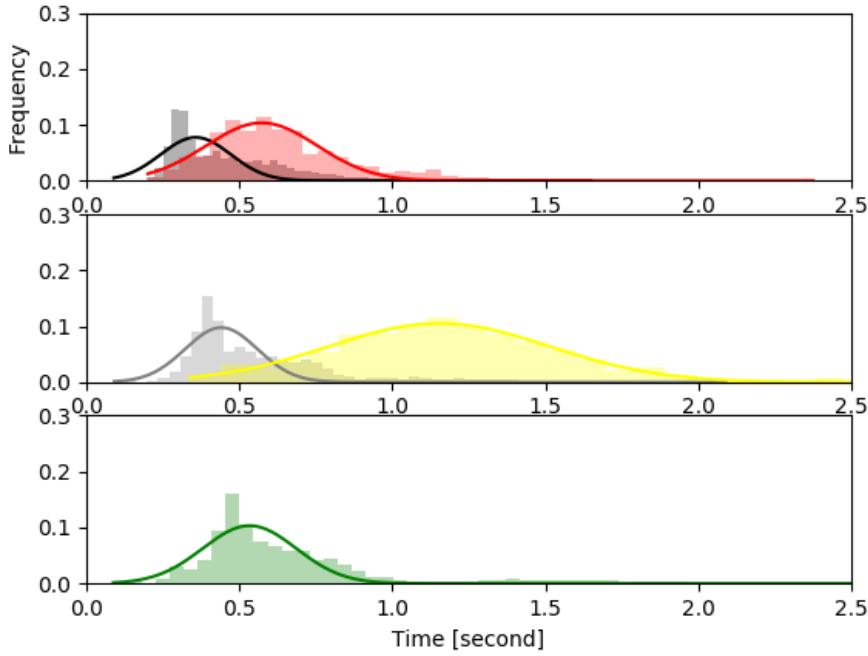


Figure 24. Relationship between latency and its frequency. The top figure is (1-2): baseline prefix-to-prefix iTTS with one morphoneme+look-1 (black: average latency of 0.464 seconds) and (3-3): iTTS with one accent phrase as a synthesis chunk (red: average latency of 0.655 seconds). The middle figure is (1-3): baseline prefix-to-prefix iTTS with one morphoneme+look-2 (gray: average latency of 0.572 seconds) and (3-4): iTTS with two accent phrases as a synthesis chunk (yellow: average latency of 1.20 seconds). The bottom figure is (1-4):prefix-to-prefix iTTS with one morphoneme+look-3 (green: average latency of 0.667 seconds).

quality than the baseline prefix-to-prefix iTTS approach with multiple words and a look-ahead approach. When the prefix-to-prefix approach uses phonemes as inputs and accent phrase units as the synthesis chunk, the speech quality would be better than the prefix-to-prefix approach with one or two-word units and lower than our proposed Japanese iTTS systems with accent phrase units.

Our iTTS systems with accent phrases showed better quality than baseline iTTS systems. Furthermore, a statistical significance test was conducted between the baseline prefix-to-prefix iTTS with look-ahead-2 and each proposed Japanese iTTS system, and a significant difference was confirmed ($p < 0.001$).

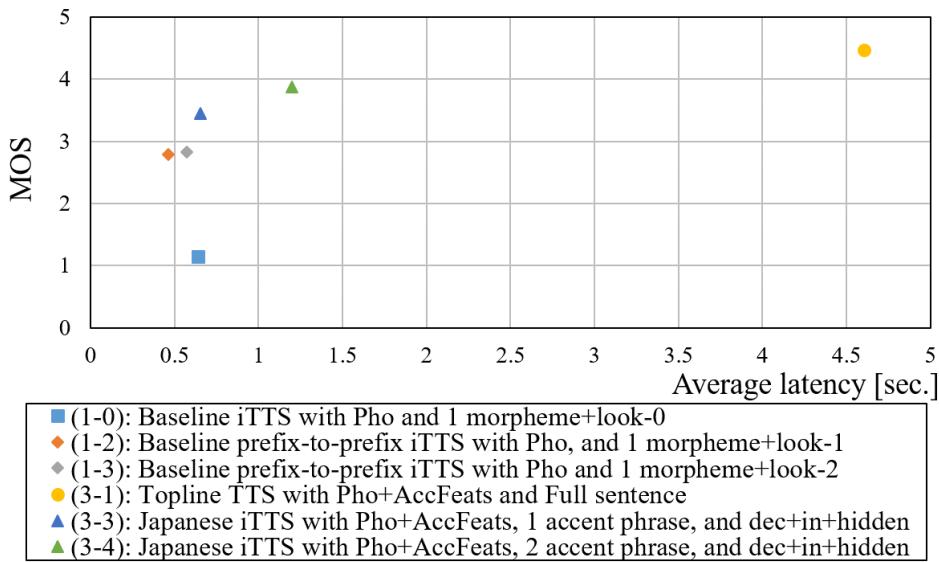


Figure 25. Relationship between MOS and its average latency. Note that MOS score of (1-0) is 1.14, (1-2) is 2.78, (1-3) is 2.83, (3-1) is 4.47, (3-3) is 3.45, and (3-4) is 3.87.

4.6 Summary

In this chapter, we proposed a novel Japanese end-to-end neural iTTS architecture without a look-ahead approach, and our proposed methods used phonemes and features based on accent phrases. We presented a method to connect the initial

input by considering the acoustic time series and connecting the model’s internal state. Moreover, we used two types of input features for the accent phrase unit. We experimentally investigated the latency of various iTTS systems with different modeling and synthesis chunks.

We objectively evaluated the speech quality regarding the differences in input and method. The objective evaluation in F0 and MCD demonstrates that the proposed iTTS system approaches the sentence-based TTS system, while the MOS score of the proposed iTTS is still lower. The proposed Japanese iTTS systems using the previous initial input and the previous model’s internal state are more efficient than word-based iTTS systems. The speech quality is improved by using input features in accent types and mora units in accent phrases. Input features in accent types and mora units are well represented in the Japanese language property as a tonal-aspect accent and a mora-timed rhythm to Japanese iTTS systems. Moreover, using a large amount of previous information is also efficient.

Furthermore, we also subjectively evaluated speech quality. Our results reveal that the proposed method with one accent phrase had better MOS scores, with a similar latency range between the baseline with a look-ahead of two morphemes and the baseline with a look-ahead of three morphemes. A method with two accent phrases improved speech quality, although the latency is slightly longer than in a baseline with a two-morpheme look-ahead approach.

5. Conclusion and Future Direction

5.1 Conclusion

In this dissertation, we proposed the Japanese iTTS on the basis of accent phrase units. Existing iTTS works were word units as a synthesis chunk. The Japanese iTTS systems were used in accent phrase units that were longer than word units.

In Chapter 2, we have described basic speech knowledge, HMM-based TTS technologies, and neural end-to-end TTS technologies. In the HMM-based TTS, many HMMs are used for representing a relationship between text and its acoustic features, and acoustic feature vectors are generated on HMM state units. Output vectors consisting of statistic and dynamic acoustic feature vectors are used for the generation algorithm. Synthesized speech is reconstructed from generated acoustic feature vectors with a discrete-time digital filter. In the end-to-end TTS, one neural network framework models relationships between text and its acoustic feature vectors, and the acoustic feature vector is generated by neural networks framewise. To generate acoustic features, dynamic and statistic features are not necessary for conditions in the neural end-to-end TTS, and the framewise generation can predict detailed acoustic feature vectors. Synthesized speech is reconstructed from generated acoustic feature vectors with neural networks.

In Chapter 3, we have proposed the Japanese HMM-based iTTS system. We have investigated a combination of linguistic features for iTTS, a relationship between synthesis chunk and its latency. We have performed speech quality of our iTTS systems with latency. The experimental results show that the HMM-based Japanese iTTS system uses accent phrases as input chunks, and input features were used as part-of-speech tags and accent phrases. Using the next accent phrase improved speech quality, but the latency would be close to an HMM-based TTS. Therefore, we have proposed the first concept of the neural end-to-end TTS system without look-ahead inputs in Chapter 4.

In Chapter 4, we have proposed a novel Japanese end-to-end neural iTTS architecture without a look-ahead approach, and our proposed methods used phonemes and features on the basis of accent phrases. Our proposed method could synthesize a speech with various chunks, and two accent phrases were optimal for synthesizing a speech without a look-ahead synthesis chunk. Our proposed neural

iTTS systems could improve speech quality using information from only previous chunks. The proposed approach was as follows: the model connected the previous acoustic features and the model’s hidden states to current states; the model used many accented features and a neural vocoder. This approach could improve the quality of Japanese iTTS. The result indicated that a baseline prefix-to-prefix iTTS with a look-ahead of two words and proposed iTTS with one accent phrase are the same latency range. A proposed iTTS with two accent phrases are better speech quality than the prefix-to-prefix iTTS, but the latency range is huge from others. The experimental results show that the proposed Japanese iTTS with one accent phrase is able to synthesize better speech quality with a similar latency range than that of the conventional baseline prefix-to-prefix neural iTTS with word units.

Finally, we have discussed our proposed methods in terms of input features and a synthesis chunk. In the input features, the Japanese HMM-based iTTS uses phonemes, part-of-speech tags, and accent phrase information for effectively training many HMMs of iTTS. A Japanese neural end-to-end TTS can train a single model from phonemes and accent phrase information. The end-to-end TTS’ speech quality has reached the human speech level, and the end-to-end TTS system is better speech quality than HMM-based TTS. Therefore, the Japanese end-to-end iTTS can generate a better speech than the HMM-based iTTS, and the number of features of Japanese end-to-end iTTS is fewer features than the HMM-based iTTS. In the synthesis chunk, both HMM-based iTTS and a baseline neural prefix-to-prefix iTTS use a look-ahead approach for improving continuity among acoustic features. However, a look-ahead approach can increase latency for look-ahead synthesis chunks. In conclusion, the proposed neural iTTS can synthesize high-quality speech with a low latency range between two words. Our end-to-end neural iTTS could be applied to other languages having the same properties, such as pitch-aspected accent and mora-timed rhythm.

5.2 Future Direction

We used a sentence for inputs to a text processor, and linguistic features were extracted by sentence units. The dissertation states that accent phrase units are essential for acoustic models to generate acoustic features from input features

and for a vocoder to synthesize speech from acoustic features. To implement the complete Japanese incremental TTS, an incremental text process must extract accent phrases and their features from a partial sentence. An incremental text process is a future direction; we will work for it in future works. We will model an incremental text processor as a wait-k approach [55]. An incremental machine translation task uses the wait-k approach, translating target words after waiting for k’s source words. The text processor with the wait-k approach would output phonemes and accent phrase features after waiting for the k Japanese input words such as Hiragana, Katakana, and Chinese characters. However, the Japanese incremental text processor would wait for the end of Chinese characters instead of k characters to estimate the pronunciations of compound words in Chinese characters, and latency would be longer than the other Japanese iTTS processes.

Another direction is that we need an approach to keep the latency to one accent and the quality to two accents. English iTTS proposed using predicted future inputs with a language model, improving the speech quality without waiting for the next input [50, 51]. The same approach may use our Japanese iTTS without the input of two accent phrases. However, predicting for Japanese iTTS is more challenging than for English iTTS. English iTTS successfully reaches human speech with only phoneme inputs, and the prediction task for the next input needs only phoneme sequences. On the other hand, our Japanese iTTS successfully improves speech quality with phonemes and accented features like the accent type. Therefore, the prediction task for the next inputs of the Japanese iTTS system becomes multi-task predictions. Future work will propose to predict both future features (i.e., next phonemes and accent features) for Japanese iTTS. Fine-tuning to train a neural vocoder with iTTS systems outputs could also improve speech quality without increasing latency.

Future work investigates the latency and the quality of iTTS under simultaneous speech translation. We will further investigate the performance of the neural iTTS system, given the partial output from the iMT systems in incremental machine speech translation systems. In this case, each incremental system’s input has a prediction error. In other words, the incremental simultaneous speech translation system has a prediction error between each ASR, incremental MT, and iTTS system. Moreover, to decrease the latency of the Japanese iTTS system,

we suggest using an iTTS system with one accent phrase and a predictor for the next accent phrase information. However, by adding a predictor to an incremental simultaneous speech translation, there are more complex relationships between the error of each model, latency, and output quality. The investigation of relationships will be a future direction.

References

- [1] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, Rif A. Saurous, Yannis Agiomvrgiannakis, and Yonghui Wu. Natural TTS Synthesis by Conditioning Wavenet on MEL Spectrogram Predictions. In *Proc. ICASSP 2018*, pages 4779–4783, 2018.
- [2] Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel WaveGAN: A Fast Waveform Generation Model Based on Generative Adversarial Networks with Multi-Resolution Spectrogram. In *Proc. ICASSP 2020*, pages 6199–6203. IEEE, 2020.
- [3] Satoshi Nakamura, Konstantin Markov, Hiromi Nakaiwa, Gen-ichiro Kikui, Hisashi Kawai, Takatoshi Jitsuhiko, J-S Zhang, Hirofumi Yamamoto, Eiichiro Sumita, and Seiichi Yamamoto. The ATR Multilingual Speech-to-Speech Translation System. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(2):365–376, 2006.
- [4] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-Based Models for Speech Recognition. In *Proc. NIPS*, volume 28, pages 577–585, 2015.
- [5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proc. ICLR 2015*, 2015.
- [6] Frieda Goldman-Eisler. Segmentation of Input in Simultaneous Translation. *Journal of Psycholinguistic Research*, 1(2):127–140, 1972.
- [7] Christian Fügen, Alex Waibel, and Muntsin Kolss. Simultaneous translation of lectures and speeches. *Machine Translation*, 21(4):209–252, 2007.
- [8] Srinivas Bangalore, Vivek Kumar Rangarajan Sridhar, Prakash Kolan, Ladan Golipour, and Aura Jimenez. Real-time Incremental Speech-to-Speech Translation of Dialogs. In *Proc. NAACL 2012*, pages 437–445, 2012.

- [9] Tomoki Fujita, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. Simple, lexicalized choice of translation timing for simultaneous speech translation. In *Proc. Interspeech 2013*, pages 3487–3491, 2013.
- [10] Hiroaki Shimizu, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. Constructing a Speech Translation System Using Simultaneous Interpretation Data. In *Proc. IWSLT 2013*, pages 212–218, Heidelberg, Germany, 2013.
- [11] Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. Optimizing Segmentation Strategies for Simultaneous Speech Translation. In *Proc. ACL 2014*, pages 551–556, 2014.
- [12] Timo Baumann and David Schlangen. Evaluating Prosodic Processing for Incremental Speech Synthesis. In *Proc. Interspeech 2012*, pages 438–441, 2012.
- [13] Timo Baumann. Decision Tree Usage for Incremental Parametric Speech Synthesis. In *Proc. ICASSP, 2014*, pages 3819–3823. IEEE, 2014.
- [14] Maël Pouget, Thomas Hueber, Gérard Bailly, and Timo Baumann. HMM training strategy for incremental speech synthesis. In *Proc. Interspeech 2015*, pages 1201–1205, 2015.
- [15] Mingbo Ma, Baigong Zheng, Kaibo Liu, Renjie Zheng, Hairong Liu, Kainan Peng, Kenneth Church, and Liang Huang. Incremental Text-to-Speech Synthesis with Prefix-to-Prefix Framework. In *Proc. Findings of ENLNP*, pages 3886–3896, 2020.
- [16] Esther Grabe and Ee Ling Low. Durational Variability in Speech and the Rhythm Class Hypothesis. *Papers in Laboratory Phonology*, 7(515-546), 2002.
- [17] Shuji Yokomizo, Takashi Nose, and Takao Kobayashi. Evaluation of Prosodic Contextual Factors for HMM-based Speech Synthesis. In *Proc. Interspeech 2010*, pages 430–433, 2010.

- [18] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. In *Proc. SSW 9*, page 125, 2016.
- [19] Keiichi Tokuda, Takao Kobayashi, Takashi Masuko, and Satoshi Imai. Mel-Generalized Cepstral Analysis - a Unified Approach to Speech Spectral Estimation. In *Proc. ICSLP 1994*, volume 94, pages 18–22, 1994.
- [20] Hideki Kawahara, Ikuyo Masuda-Katsuse, and Alain De Cheveigne. Restructuring Speech Representations Using a Pitch-Adaptive Time-Frequency Smoothing and an Instantaneous-Frequency-Based F0 Extraction: Possible Role of a Repetitive Structure in Sounds. *Speech Communication*, 27(3):187–207, 1999.
- [21] Esther Grabe and Ee Ling Low. Durational Variability in Speech and the Rhythm Class Hypothesis. In *Laboratory phonology 7*, pages 515–546. De Gruyter Mouton, 2008.
- [22] Yoshinori Sazisaka and Hirokazu Sato. Accentuation Rules for Japanese Word Concatenation. *IEICE TRANSACTIONS on Information and Systems (Japanese Edition)*, J66-D(7):849–856, 1983.
- [23] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura. Speech Parameter Generation Algorithms for HMM-Based Speech Synthesis. In *Proc. ICASSP 2000*, pages 1315–1318, 2000.
- [24] Keiichi Tokuda, Takashi Masuko, Noboru Miyazaki, and Takao Kobayashi. Multi-Space Probability Distribution HMM. *IEICE TRANSACTIONS on Information and Systems*, 85(3):455–464, 2002.
- [25] Heiga Zen, Keiichi Tokuda, Takashi Masuko, Takao Kobayashi, and Tadashi Kitamura. Hidden Semi-Markov Model Based Speech Synthesis. In *Proc. Interspeech 2004*, pages 1393–1396, 2004.
- [26] Satoshi Imai, Kazuo Sumita, and Chieko Furuichi. Mel Log Spectrum Approximation (MLSA) Filter for Speech Synthesis. *Electronics and Communications in Japan*, 66(2):10–18, 1983.

- [27] Takato Fujimoto, Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku, and Keiichi Tokuda. Impacts of Input Linguistic Feature Representation on Japanese End-to-End Speech Synthesis. In *Proc. SSW 10*, pages 166–171, 2019.
- [28] Yusuke Yasuda, Xin Wang, and Junichi Yamagishi. Investigation of Learning Abilities on Linguistic Features in Sequence-to-Sequence Text-to-Speech Synthesis. *Computer Speech & Language*, 67:101183, 2021.
- [29] Kiyoshi KURIHARA, Nobumasa SEIYAMA, and Tadashi KUMANO. Prosodic Features Control by Symbols as Input of Sequence-to-Sequence Acoustic Modeling for Neural TTS. *IEICE Transactions on Information and Systems*, E104.D(2):302–311, 2021.
- [30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [31] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proc. ICML 2015*, page 448–456, 2015.
- [32] David Krueger, Tegan Maharaj, János Kramár, Mohammad Pezeshki, Nicolas Ballas, Nan Rosemary Ke, Anirudh Goyal, Yoshua Bengio, Aaron C. Courville, and Christopher J. Pal. Zoneout: Regularizing RNNs by Randomly Preserving Hidden Activations. In *Proc. ICLR 2017*, 2017.
- [33] Jing-Xuan Zhang, Zhenhua Ling, and Lirong Dai. Forward Attention in Sequence- To-Sequence Acoustic Modeling for Speech Synthesis. *Proc. ICASSP 2018*, pages 4789–4793, 2018.
- [34] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Proc. NIPS 2014*, volume 27, 2014.

- [35] Xin Wang, Shinji Takaki, and Junichi Yamagishi. Neural Source-Filter-Based Waveform Model for Statistical Parametric Speech Synthesis. In *Proc. ICASSP 2019*, pages 5916–5920, 2019.
- [36] Takayoshi Yoshimura, Keiichi Tokuda, Takashi Masuko, Takao Kobayashi, and Tadashi Kitamura. Simultaneous Modeling of Spectrum, Pitch and Duration in HMM-Based Speech Synthesis. In *Proc. Eurospeech 1999*, pages 2347–2350, 1999.
- [37] Timo Baumann. Partial Representations Improve the Prosody of Incremental Speech Synthesis. In *Proc. Interspeech 2014*, pages 2932–2936., 2014.
- [38] Maël Pouget, Olha Nahorna, Thomas Hueber, and Gérard Bailly. Adaptive Latency for Part-of-Speech Tagging in Incremental Text-to-Speech Synthesis. In *Proc. Interspeech 2016*, pages 2846–2850, 2016.
- [39] Akira Kurematsu, Kazuya Takeda, Yoshinori Sagisaka, Shigeru Katagiri, Hisao Kuwabara, and Kiyohiro Shikano. ATR japanese speech database as a tool of speech recognition and synthesis. *Speech Communication*, 9(4):357–363, 1990.
- [40] Heiga Zen, Takashi Nose, Junichi Yamagishi, Shinji Sako, Takashi Masuko, Alan W Black, and Keiichi Tokuda. The HMM-based speech synthesis system (hts) version 2.0. In *Proc. SSW workshop 2007*, pages 294–299, 2007.
- [41] R Kubichek. Mel-Cepstral Distance Measure for Objective Speech Quality Assessment. In *Communications, Computers and Signal Processing, 1993., IEEE Pacific Rim Conference*, volume 1, pages 125–128. IEEE, 1993.
- [42] Recommendation, ITU-T. P.800 Methods for Subjective Determination of Transmission Quality. *International Telecommunication Union*, 1996.
- [43] Yuxuan Wang, R.J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous. Tacotron: Towards End-to-End Speech Synthesis. In *Proc. Interspeech 2017*, pages 4006–4010, 2017.

- [44] Aaron Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, et al. Parallel WaveNet: Fast High-Fidelity Speech Synthesis. In *Proc. ICML 2018*, pages 3918–3926. PMLR, 2018.
- [45] Heiga Zen, Andrew Senior, and Mike Schuster. Statistical Parametric Speech Synthesis Using Deep Neural Networks. In *Proc. ICASSP*, pages 7962–7966, 2013.
- [46] Heiga Zen and Hasim Sak. Unidirectional Long Short-Term Memory Recurrent Neural Network with Recurrent Output Layer for Low-Latency Speech Synthesis. In *Proc. ICASSP 2015*, pages 4470–4474, 2015.
- [47] Tomoya Yanagita, Sakriani Sakti, and Satoshi Nakamura. Neural ITTS: Toward Synthesizing Speech in Real-Time with End-to-End Neural Text-to-Speech Framework. In *Proc. SSW*, pages 183–188, 2019.
- [48] Brooke Stephenson, Laurent Besacier, Laurent Girin, and Thomas Hueber. What the Future Brings: Investigating the Impact of Lookahead for Incremental Neural TTS. In *Proc. Interspeech 2020*, pages 215–219, 2020.
- [49] Devang S. Ram Mohan, Raphael Lenain, Lorenzo Foglianti, Tian Huey Teh, Marlene Staib, Alexandra Torresquintero, and Jiameng Gao. Incremental Text to Speech For Neural Sequence-to-Sequence Models Using Reinforcement Learning. In *Proc. Interspeech 2020*, pages 3186–3190, 2020.
- [50] Brooke Stephenson, Thomas Hueber, Laurent Girin, and Laurent Besacier. Alternate Endings: Improving Prosody for Incremental Neural TTS with Predicted Future Text Input. In *Proc. Interspeech 2021*, pages 3865–3869, 2021.
- [51] Takaaki Saeki, Shinnosuke Takamichi, and Hiroshi Saruwatari. Incremental Text-to-Speech Synthesis Using Pseudo Lookahead with Large Pretrained Language Model. *IEEE Signal Processing Letters*, 28:857–861, 2021.
- [52] Yusuke Yasuda, Xin Wang, Shinji Takaki, and Junichi Yamagishi. Investigation of Enhanced Tacotron Text-to-speech Synthesis Systems with Self-

- attention for Pitch Accent Language. In *Proc. ICASSP*, pages 6905–6909, 2019.
- [53] Ryosuke Sonobe, Shinnosuke Takamichi, and Hiroshi Saruwatari. JSUT Corpus: Free Large-Scale Japanese Speech Corpus for End-to-End Speech Synthesis. In *arXiv preprint arXiv:1711.00354*, 2017.
- [54] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proc. ICLR 2015*, 2015.
- [55] Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proc. 57th ACL, 2019*, pages 3025–3036, 2019.

Acknowledgements

Here, I thank my supervisor, Professor Satoshi Nakamura.
He has given me many inspirations and supported my research anytime.
I also thank Professor Taro Watanabe.
I also thank Associate Professor Sakriani Sakti.
She is an excellent researcher and gives me many things about my research life.
I also thank Lecturer Shinnosuke Takamichi.
I also want to thank the lab's students and assistant Ms. Manami Matsuda for helping the life of the AHC laboratory.
Finally, I thank my family for helping my life.

Publication List

Peer review journal paper

1. Tomoya Yanagita, Sakriani Sakti, Satoshi Nakamura. Synthesis Unit for Japanese Incremental Text-to-Speech. volume 63 No. 4, pp. 1149-1158, Apr.15, 2022, IPSJ journal (Japanese edition).
2. Tomoya Yanagita, Sakriani Sakti, Satoshi Nakamura. Japanese Neural Incremental Text-to-speech Synthesis Framework with an Accent Phrase Input. Volume 11, 22355-22363, Mar. 2, 2023, IEEE Access.

Peer review international conference

1. Tomoya Yanagita, Sakriani Sakti, Satoshi Nakamura. Incremental TTS for Japanese Language. Proc. Interspeech 2018, 902-906, 2-6 September 2018.
2. Tomoya Yanagita, Sakriani Sakti, Satoshi Nakamura. Neural iTTS: Toward Synthesizing Speech in Real-time with End-to-end Neural Text-to-Speech Framework. Proc. 10th ISCA Workshop on Speech Synthesis, 183-188, 20-22 September 2019.