Doctoral Dissertation

Language-aware Code-switching Speech Recognition

Sahoko Nakayama

March 16, 2022

Graduate School of Science and Technology
Nara Institute of Science and Technology

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

Sahoko Nakayama

Language-aware Code-switching Speech Recognition[*]

Sahoko Nakayama

Abstract

The phenomenon where a speaker mixes two or more languages within the same conversation is called code-switching (CS). Handling CS is challenging for automatic speech recognition (ASR) because it requires coping with multilingual input. There are many challenges for the CS ASR, but this thesis focuses on the following three problems for the CS ASR's development: language coverage, training mechanism, and usability.

The first is the language coverage. Most of the previous researches only focused on a single-pair CS. However, when we want to handle multi-pair CS beyond a single-pair CS, developing multiple systems per single-pair CS can be an exhausting task. Therefore, the unified system for multi-pair CS is desirable to simplify the process of training, deploying, maintaining, and the recognition task. To realize the multi-pair CS system, we introduce the language-aware mechanism by utilizing a language identification system. It enables to handle multi-pair CS better by providing language information. Various approaches utilizing language identification systems were investigated.

The second is the training mechanism. The datasets of CS speech and the corresponding CS transcriptions are hard to obtain. To solve the CS data problem, we utilize the framework called the machine speech chain. The machine speech chain is the mechanism inspired by the human communication mechanism called the speech chain. It enables ASR and TTS to assist each other when they receive unpaired data since it allows them to infer the missing pair and optimize

[*]Doctoral Dissertation, Graduate School of Science and Technology,
Nara Institute of Science and Technology, March 16, 2022.

the models with reconstruction loss. In addition, we also integrate language identification into the ASR and language embedding into the TTS of the CS machine speech chain.

The third problem is usability. The common aim of developing a CS ASR is merely for transcribing CS-speech utterances into CS-text sentences, where we assume only dialogues between the same CS speakers. In contrast, in this study, we address the situational context that happens during dialogues between CS and non-CS (monolingual) speakers and support monolingual speakers who want to understand CS speakers. We construct a system that can recognize code-switching speech and translate it into monolingual texts to support monolingual speakers who are trying to understand CS speakers. We investigate several approaches, including a cascade of ASR and neural machine translation (NMT), a cascade of ASR and a Bidirectional Encoder Representations from Transformers (BERT), a single-task speech translation, and a multi-task speech translation.

Keywords:

ASR, Code-switching, Language identification, Semi-supervised learning, TTS, Machine speech chain

Contents

List of Figures

List of Abbreviations

ASR	Automatic Speech Recognition
BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional Long Short Term Memory
BPE	Byte-pair Encoding
BTEC	Basic Travel Expression Corpus
CBHG	1-D convolution Bank + Highway network + Bidirectional GRU
CER	Character Error Rate
Chr	Character
CNN	Convolutional Neural Networks
CS	Code-switching
DNN	Deep Neural Network
EN	English
FR	French
JA	Japanese
JASSO	Japan Student Services Organization
LER	Language Error Rate
LID	Language Identification (Discrimination)
LM	Language Model
Lng	Language
L1	First language
LSTM	Long Short Term Memory
L2	Second language
mBERT	Multilingual BERT
MEXT	Ministry of Education, Culture, Sports, Science, and Technology
MHLW	Ministry of Health, Labour and Welfare
MLP	Multi-Layer Perceptron
Mono	Monolingual
MSE	Mean Squared Error
NMT	Neural Machine Translation
PASM	Pronunciation-Assisted Sub-word Modeling
POS	Part Of Speech
PRLM	Phone Recognition followed by Language Modeling

PPRLM	Parallel Phone Recognition followed by Language Modeling
RNN	Recurrent Neural Network
seq2seq	Sequence-to-sequence
SPKREC	Speaker Recognition
ST	Speech Translation
STFT	Short-Time Fourier Transform
TER	Token Error Rate
TTS	Text-To-Speech
WER	Word Error Rate
ZH	Chinese

Chapter 1

Introduction

1.1 Bilingualism and Code-switching Phenomena

The number of Japanese-English bilingual speakers continues to increase. According to a Ministry of Health, Labour and Welfare (MHLW) survey, the number of children in Japan with at least one non-Japanese parent has risen gradually over the past 25 years [1]. The Japanese Ministry of Education, Culture, Sports, Science, and Technology (MEXT) also reported that the number of school-age children who have lived abroad was reported that more than doubled in 2015 [2]. Moreover, the number of Japanese students who study abroad has nearly tripled in the past ten years, as reported by the Japan Student Services Organization (JASSO) [3]. These changes are affecting how people communicate with each other. Bilingual speakers sometimes mix two or more languages while speaking, which is called code-switching.

Code-switching (CS) is formally defined as the phenomenon of alternating or mixing two or more languages in discourse (often with no change of interlocutor or topic). It is a hallmark of bilingual communities worldwide [4]. CS has been studied for several decades. Most researchers agree that it plays a vital role in bilingualism and is more than a random phenomenon [5]. Nakamura (2005) surveyed the code-switching of a Japanese child who lived in the United States and found that 179 switches occurred during a one-hour conversation with his mother

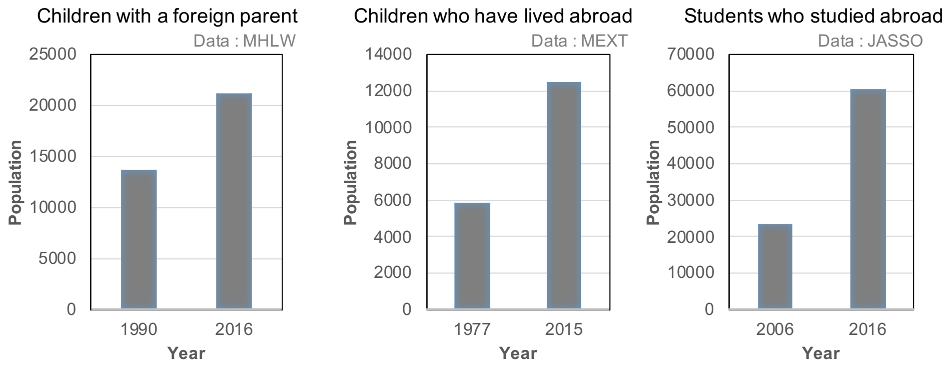


Figure 1.1: The statistics of bilingual population growth in Japan.

[6]. Fotos investigated four hours of conversations of four bilingual children in Japan with at least one American parent and observed 153 code-switchings [7]. Both reports reveal that people actually use Japanese-English CS in everyday life.

1.2 Challenges of Handling Code-switching Conversation

Handling code-switching speech is challenging for automatic speech recognition (ASR). There are the following difficulties.

1.2.1 Switching Positions

The units and locations of switches may vary widely. It can be categorized into two main categories: intra-sentential and inter-sentential. In intra-sentential CS, the language shift occurs within a sentence. The intra-sentential CS may be inserted from the length of a single word to phrases that exceed the loanwords. In inter-sentential CS, language switching occurs at the sentence boundaries. We show some Japanese-English CS examples collected from a bilingual CS user:

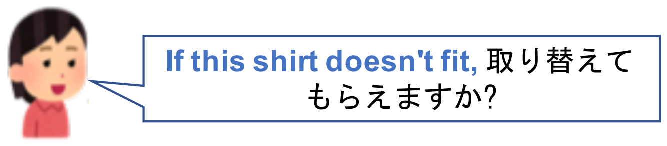
- **Intra-sentential CS:**
  - **Word-level CS:**
    "国会が the Equal Employment Opportunity Law に罰則を 設けなかったので、空文だという意見があります."
    (*Since the Diet did not put any teeth into the Equal Employment Opportunity Law, some believe that it is merely a scrap of paper.*)
  - **Phrase-level CS:**
    "If I could make a suggestion, この議題についての討議を昼食までに終えて頂ければと思いますが."
    (*If I could make a suggestion, why do not we finish discussing this subject by lunch?*)
- **Inter-sentential CS:**
  "In the end, he quit his job and followed in his father's footsteps, taking over the family business. やっぱりかえるの子はかえるだね."
  (*His son's a chip off the old block, all right. In the end, he quit his job and followed in his father's footsteps, taking over the family business.*)

However, there are some cases not to be considered as CS. Loanwords, which are borrowed from a foreign language, may not be included in intra-sentential CS. Quotations, which borrow part of another's text or speech, may not be included in the intra-sentential phrase-level CS.

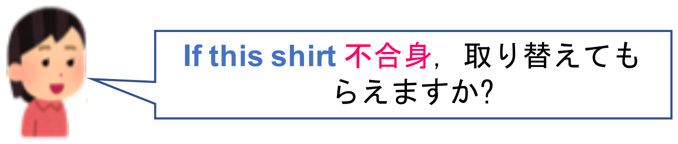
- **Loan words:**
  "中間言語を使った時の<u>メリット</u>に何があるか ?"
  (*What is the <u>merit</u> of using an interlingual?*)
- **Quotations:**
  "What do you think of the Japanese saying, うそつきは泥棒の始まり ?"
  (*What do you think of the Japanese saying, "Show me a liar and I'll show you a thief"?*)

A CS utterance can include not only two languages but also more than two languages. For example, Fig. 1.2 (a) shows the CS of two languages, English and Japanese, and Fig. 1.2 (b) shows the CS of more than two languages, English, Chinese and Japanese. The CS of two languages is more common than that of more than two languages. A survey [8] revealed that even trilingual speakers mostly use only the CS of two languages. It showed that only 33% of 60 trilingual speakers used the CS of three languages even though 100% of them used the CS of two languages.

Then a CS utterance is basically spoken by a single speaker. It is not switched by multi-speakers within one utterance.



(a)



(b)

Figure 1.2: One CS utterance of (a) two languages and (b) more than two languages.

1.2.2 Language Coverage and Proficiency

There are several kinds of bilingual speakers, such as English-Japanese, English-Korean, and Japanese-Chinese. Therefore, the system for single-pair CS, which covers only the same pairs of two languages (Fig. 1.3 (a)), is not enough to support all bilingual speakers. The ASR for multi-pair CS, which covers different pairs of two languages (Fig. 1.3 (b)), is required for covering all bilingual speakers. Besides, tackling the CS with different pairs of more than two languages (Fig. 1.3 (c)) would be more desirable for recognizing all CS conversations.

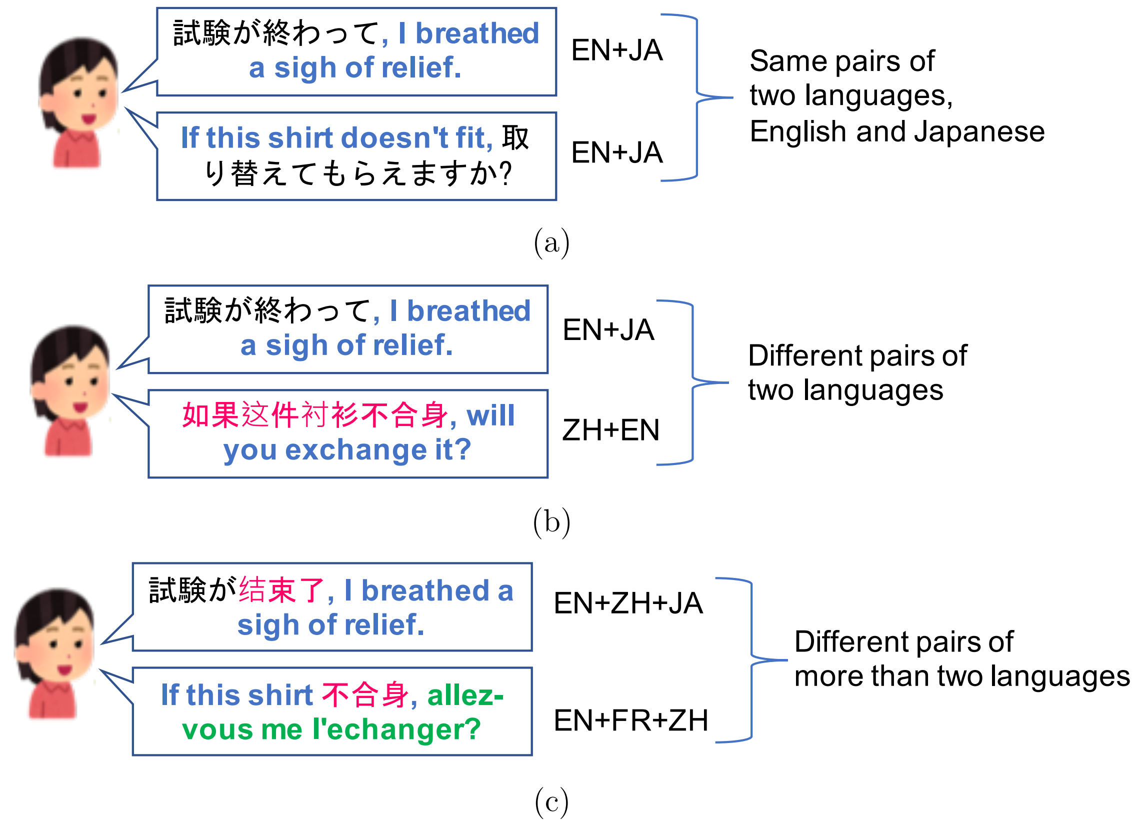

試験が終わって, I breathed a sigh of relief.
If this shirt doesn't fit, 取り替えてもらえますか?
EN+JA
EN+JA
Same pairs of two languages, English and Japanese
(a)
試験が終わって, I breathed a sigh of relief.
如果这件衬衫不合身, will you exchange it?
EN+JA
ZH+EN
Different pairs of two languages
(b)
試験が结束了, I breathed a sigh of relief.
If this shirt 不合身, allez-vous me l'echanger?
EN+ZH+JA
EN+FR+ZH
Different pairs of more than two languages
(c)

Figure 1.3: The overview of (a) single-pair CS with same pairs of two languages, (b) multi-pair CS with different pairs of two languages, and (c) CS with different pairs of more than two languages.

Then the CS proficiency level varies from beginners to near-native speakers since CS switches between the first language (L1) and the second language (L2), where only one of the languages is the mother tongue. We categorize CS into native CS and non-native CS based on the L2 proficiency level. The native CS is near-native speaker level, and the non-native CS holds non-native sounds. Handling them together may degrade the ASR performance since it causes a mismatch between speech and acoustic models [9]. Therefore, it is also necessary to develop the CS system to cover native and non-native.

1.2.3 Training Mechanism

The primary method for training CS ASR is supervised learning. Supervised learning is a more straightforward method since it can train with the target labels. However, this method requires a large amount of labeled data for training the model, which takes money and time. Especially, the datasets of CS speech and the corresponding CS transcriptions are difficult to obtain. The CS's existing corpus is limited to some language pairs and accents. Moreover, collecting new corpus is not easy since we have to prepare the special speakers and situations for CS conversations, and CS annotation requires high language skills. Therefore, we need to seek the training mechanism trainable with fewer labeled CS data.

1.2.4 Usability

The common aim of developing a CS ASR is merely for transcribing CS-speech utterances into CS-text sentences, where we assume only dialogues between the same CS speakers. However, we need to suppose the situational context during dialogues between CS and non-CS (monolingual) speakers and support monolingual speakers who want to understand CS speakers.

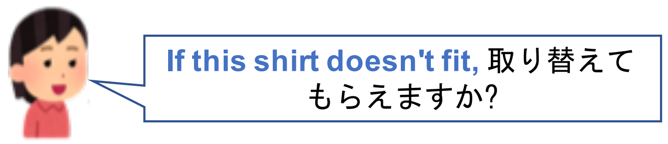
1.3 Scope of This Thesis

In this thesis, we handle the following:

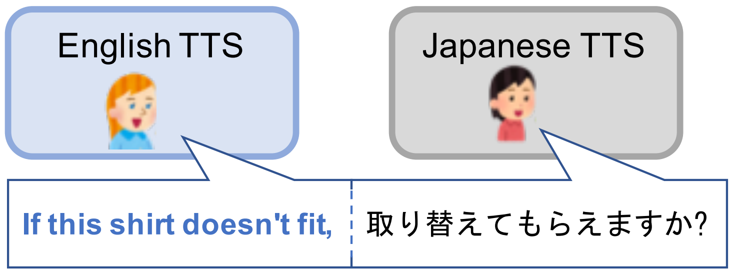
1. **Switching Positions**
   We handle several kinds of CS, including intra-sentential CS and inter-sentential CS. Although loan words and quotations may not be CS in principle, we include them in our CS targets because we want to recognize all words in multilingual conversations.

   This thesis assumes that the CS can allow only two languages for one utterance since we could not obtain the CS including more than two languages. In addition, one CS utterance should be basically spoken by a single speaker as Fig. 1.4 (a) shows, but we sometimes permit the CS to be switched by multi-speakers within an utterance to cover low-resourced CS data as

Fig. 1.4 (b) shows. It is the case when we utilize the synthesized speech generated using the corresponding language's TTS, where English words are synthesized with English TTS by an English speaker, and Japanese words are synthesized with Japanese TTS by a Japanese speaker.



(a)



(b)

Figure 1.4: One CS utterance by (a) a single speaker and (b) multi-speakers of TTS.

2. **Language Coverage and Proficiency**
   This thesis covers multi-pair CS as well as single-pair CS. Developing multiple systems per single-pair CS can be exhausting when handling multi-pair CS beyond a single-pair CS. Therefore, we developed a unified system for multi-pair CS to simplify the process of training, deploying, maintaining, and the recognition task. In our research, single-pair CS means one sentence has only two languages and there are only the same language pairs in that data. Fig. 1.3 (a) showed the single-pair CS, which had the same pairs of two languages, English and Japanese. The multi-pair CS means one sentence has only two languages and there are different language pairs in that data. Fig. 1.3 (b) showed the multi-pair CS, which had different pairs of two languages, English and Japanese or Chinese and English. Both single-pair CS and multi-pair CS have only two languages in one sentence.

This thesis does not handle the CS with different pairs of more than two languages (see Fig. 1.3 (c)). We also handle native CS and non-native CS, respectively.

3. **Training Mechanism**
   We seek an efficient training mechanism with fewer labeled CS data to solve the CS data problem. In this thesis, we investigate one of the semi-supervised learning methods, where we train the CS ASR with unpaired CS data (speech or text only). Pairs of CS speech and corresponding CS transcriptions are scarce and difficult to obtain, but either CS text or CS speech may be found on social media.

4. **Usability**
   This study addresses the situational context that happens during dialogues between CS and non-CS (monolingual) speakers and supports monolingual speakers who want to understand CS speakers.

1.4 Thesis Objective and Contribution

1. **Language Coverage**
   The objective is to increase the language coverage without reducing the performance. The contribution is developing the language-aware system. By incorporating the language-aware system, we expect to handle multi-pair CS better given the language information.

2. **Training Mechanism**
   The objective is to seek an efficient training mechanism with less CS data. The contribution is investigating a semi-supervised approach of the machine speech chain [10,11] and improving it for CS tasks with the language-aware mechanism.

3. **Usability**
   The objective is to consider the situational context during dialogues between CS and non-CS (monolingual) speakers and support non-CS speakers trying to understand CS speakers. The contribution is constructing a system that

can recognize CS speech and translate it into monolingual texts. We investigate several approaches, including a cascade of ASR and neural machine translation (NMT), a cascade of ASR and a Bidirectional Encoder Representations from Transformers (BERT), a single-task speech translation, and a multi-task speech translation.

1.5 Thesis Outline

This thesis is organized as follows:
In Chapter 2, we introduce the mechanism of spoken language technologies. We first see the overview of the learning algorithm and explain the end-to-end sequence-to-sequence modeling. Chapter 3 is about all of the datasets used in our experiments. Chapter 4 is about the language-aware code-switching ASR, where we compare the proposed LID approach with the other LID approaches. Chapter 5 explains the semi-supervised approach for CS ASR and CS TTS with the machine speech chain. Chapter 6 is about code-switching speech translation. We show the experimental results of several approaches for code-switching speech translations and discuss them. Finally, we close the thesis with a conclusion and a discussion on the future direction of the study in Chapter 7.

Chapter 2

Spoken Language Technologies

This chapter shows the overview of the learning algorithm and explains the end-to-end sequence-to-sequence modeling.

2.1 Overview of Learning Algorithm

2.1.1 Supervised Learning

Supervised learning is the primary method for machine learning. Fig. 2.1 shows the overview of supervised learning on classification. We are given a dataset with category labels, and the learning algorithm derives the function classifying the data and maps new input variables into the categories. Supervised learning problems can be applied to the regression problems as well as those classification problems. Supervised learning is a straightforward method to train but needs a lot of labeled data, so it costs money and time for preparing the datasets.

2.1.2 Unsupervised Learning

Unsupervised learning does not teach the target output given the input. Fig. 2.2 shows the data without any labels. The learning algorithm derives the structure by clustering the data based on relationships among the variables in the data. Unsupervised learning does not need any labels, thereby solving the time-consuming and expensive problems of data labeling. However, the performance

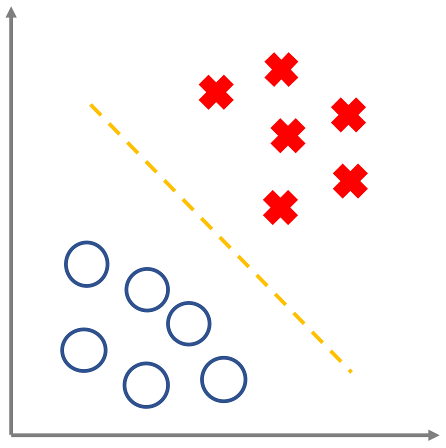


Figure 2.1: Supervised learning. We are given a dataset with category labels. The learning algorithm derives the function classifying the data and maps new input variables into the categories.

might be less accurate as it has to predict without any prior knowledge.

2.1.3 Semi-supervised Learning

Semi-supervised learning combines supervised and unsupervised learning. It requires some of the labeled data, but the others are allowed to be unlabeled data. While the labeled data costs time and money to collect, the unlabeled data is comparatively easier to obtain. Therefore, semi-supervised learning is useful for situations where we cannot prepare enough data with only labeled data but can be enough by adding unlabeled data. The semi-supervised approaches conduct supervised learning with labeled data and continue the training with unlabeled data in an unsupervised way. Common semi-supervised methods adopt pseudo-labels for unsupervised learning, where the pseudo-labels are inferred using the initially trained supervised model. It may make the performance unstable because of the unreliable labels. However, the semi-supervised approach is still helpful since it saves the effort for collecting labels and can be expected to perform better than unsupervised learning.

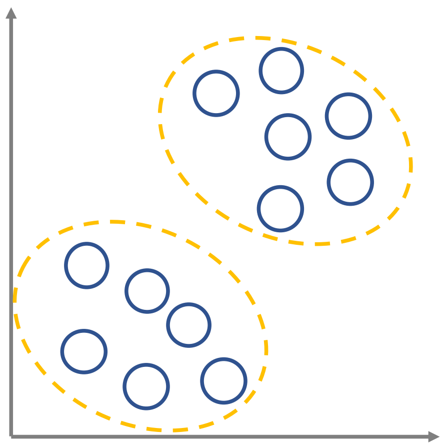


Figure 2.2: Unsupervised learning. The data does not have any labels. The learning algorithm derives the structure by clustering the data based on relationships among the variables in the data.

2.2 End-to-end Sequence-to-sequence Modeling

2.2.1 Overview

Sequence-to-sequence (seq2seq) is one of the deep learning approaches for spoken language technologies. We adopt this framework throughout our researches, so we describe it in detail in this chapter.

Sequence-to-sequence Framework

Sequence-to-sequence (seq2seq) model [12, 13] refers to the system that can convert an input sequence to a target sequence.The sequences will be lists of speech features or lists of characters, which have variable lengths. When we define the input sequence as $\boldsymbol{x} = (x_1, x_2, \cdots, x_S)$ and the target sequence as $\boldsymbol{y} = (y_1, y_2, \cdots, y_T)$, we can denote the seq2seq model as

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{t=1}^{T} p(y_t|y_1, y_2, \cdots, y_{t-1}, \boldsymbol{x}). \tag{2.1}$$

At time t, it predicts the target y_t with the previous output sequences $y_{<t}$ as well as the input.

The seq2seq model architecture consists of an encoder and a decoder. The encoder converts the input vector to a fixed dimensional vector representation, and the decoder predicts the output sequence from the vector. An attention mechanism [14] can map between the encoder and the decoder. It calculates context vector c_t using attention weights α_{ts}, which obtain the most relevant encoder representation to the decoder state:

$$c_t = \sum_{s=0}^{S} \alpha_{ts} h_s, \tag{2.2}$$

$$\alpha_{ts} = \frac{exp(e_{ts})}{\sum_{k=1}^{T} exp(e_{tk})}, \tag{2.3}$$

$$e_{ts} = Score(s_t, h_s). \tag{2.4}$$

The *Score* function determines how the encoder and decoder outputs are related, where s_t is the decoder's hidden vector and h_s is the encoder's hidden vector. One of the calculating methods is the multilayer perceptron (MLP) [15]:

$$Score(s_t, h_s) = w_a^{\mathsf{T}} tanh(W_a[s_t, h_s]), \tag{2.5}$$

where w_a, W_a is the weight vector and $tanh$ is an activation function.

Training and Decoding

The role of the encoder in the training and decoding phase is the same. Given the input sequence, it outputs the vector representation and sends it to the decoder. On the other hand, the decoder has a different way between the training and testing phase. The training approach called "Teacher forcing" uses the ground-truth of the previous time step y_{t-1}^{true} to predict the output of the current time step y_t^{pred} as Fig. 2.3 shows. The loss between the hypothesis y_t^{pred} and the ground-truth y_t^{true} is calculated at each time step, and the parameters are updated with back-propagation by the cumulative loss through time sequence.

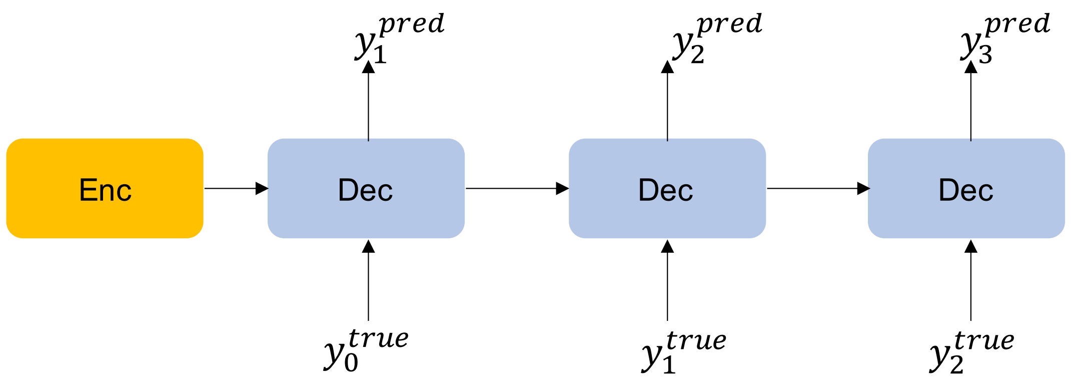


Figure 2.3: Teacher-forcing approach using the previous ground-truth y_{t-1}^{true} as input to predict y_t^{pred} at time t.

In the decoder of the test phase, since we can not use the ground-truth, we use the predicted output of the previous time step to input the current time step. The simplest decoding strategy is greedy search. The greedy search approach searches for the token with a maximum conditional probability per time step.

$$\hat{y}_t = \arg\max_{y_t} p(y_t|y_1, y_2, \cdots, y_{t-1}, \boldsymbol{x}). \tag{2.6}$$

The search continues until the output sequence has reached the length of target sequence T. The greedy search method (Fig. (a)) only considers the maximum conditional probability under each step, but the result sequence is not always optimal. If we want to guarantee the optimal sequence, we have to search by exhaustive search. The exhaustive search approach searches all the possible sequences with their conditional probabilities, and the sequences with the maximum conditional probability will be the final result. However, the computational cost is too high. When the number of possible sequences is S, the computational cost is $O(S^T)$. The computational cost of the greedy search is $O(T)$, so the computational cost of the exhaustive search is too expensive. On the other hand, the beam search (Fig. (b)) is a trade-off method between computational cost and accuracy. Using a hyperparameter called beam size, k, it selects the top-k tokens with high conditional probabilities at each time step. The following

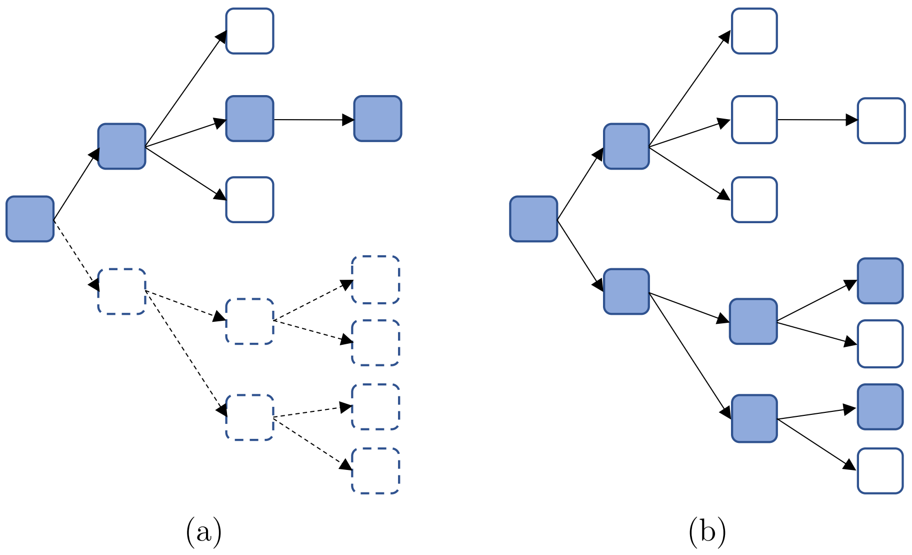


Figure 2.4: (a) greedy search algorithm considering the maximum conditional probability under each time step and (b) beam search algorithm selecting the top-k tokens with high conditional probabilities at each time step.

steps further search top-k tokens with high conditional probabilities based on the previous candidates. If we use the priority queue to sort the candidates, we can expand only top-k candidates at every step. Finally, the computational cost is $O(kT)$, much smaller than the exhaustive search. Therefore, we mainly adopt the beam-search decoding for our experiments of this thesis.

2.2.2 Sequence-to-sequence ASR

In the case of ASR, the input sequence $\boldsymbol{x}$ for the seq2seq model is speech features. Speech feature refers to the feature vector that extracts the important information for speech recognition from the speech signal. The speech feature has several variations, such as Mel-spectrogram and MFCC. They have usually multi-dimensional, not 1-dim. Then each time step in $\boldsymbol{x} = (x_1, x_2, \cdots, x_S)$ is each frame. It means that one input sequence $\boldsymbol{x}$ consists of speech feature dimension and speech frame lengths. Output $\boldsymbol{y}$ is speech transcription, where each time step equals to be each token.

One of the seq2seq ASR model is an attention-based encoder-decoder model

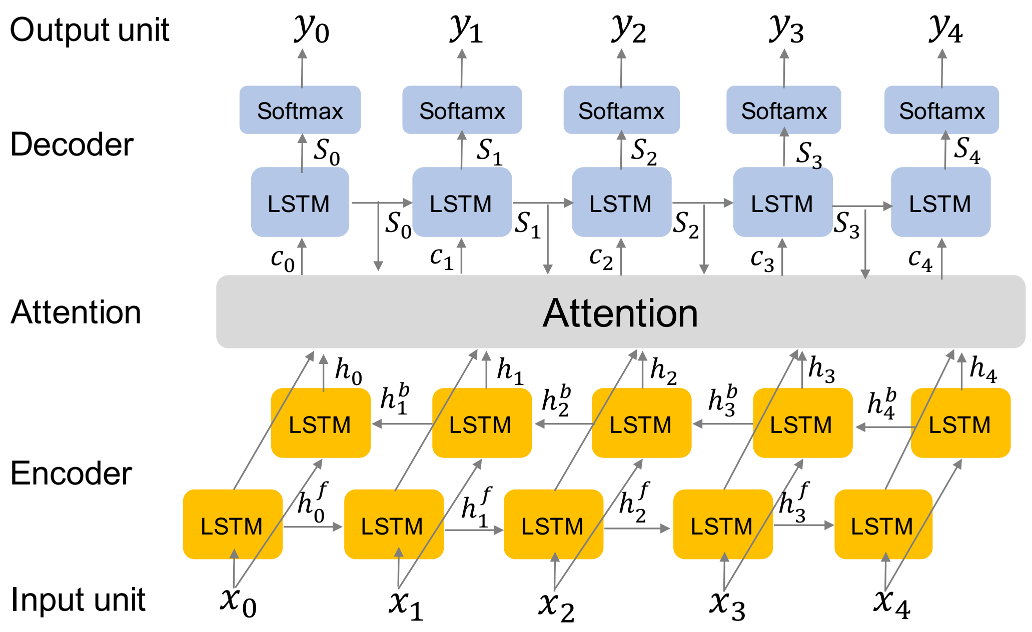


Figure 2.5: Attention-based encoder-decoder.

[16, 17]. It is composed of encoder, decoder, and attention. Fig. 2.5 shows the basic architecture. In the encoder, from the input sequences of speech features $\boldsymbol{x} = (x_1, x_2, \cdots, x_S)$, it outputs hidden vectors through bidirectional LSTM layers:

$$h_s^f = LSTM(h_{s-1}^f, x_s), \tag{2.7}$$

$$h_s^b = LSTM(h_{s+1}^b, x_s), \tag{2.8}$$

where h_s^f is a forward hidden vector at time s and h_s^b is a backward hidden vector h_s^b at time s. The final hidden vector concatenated h_s^f and h_s^b, which is then denoted as h_s at time s. Then an attention module described in Sect. 2.2.1 produces context vector.

The decoder generates output y_t with all the previously predicted words $y_1, y_2, \cdots, y_{t-1}$ and context vector c_t:

$$p(y_t|y_1, y_2, \cdots, y_{t-1}, c_t) = g(s_t, y_{t-1}, c_t), \tag{2.9}$$

where g is an activation function that calculates the probability of y_t. The decoder hidden vector s_t is calculated with the LSTM layer:

$$s_t = LSTM(s_{t-1}, y_{t-1}, c_t). \tag{2.10}$$

Finally, the softmax layer outputs the probability score for the target label.

$$p(y_t|y_{<t}, \boldsymbol{x}) = Softmax(s_t). \tag{2.11}$$

For optimizing ASR, we attempt to decrease the cross entropy loss function to maximize the probability of target sequence $\boldsymbol{y}$ with the context vector of decoder input $\boldsymbol{c}$ and previous output $y_{1:t-1}$:

$$L_{ASR} = -\frac{1}{T}\sum_{t=1}^{T} \log P(y_t|c_t, y_{1:t-1}), \tag{2.12}$$

where posterior probability $P(y_t|c_t, y_{1:t-1})$ is calculated by a softmax function.

2.2.3 Sequence-to-sequence TTS

Seq2seq model for TTS takes sequences of tokens as input and outputs sequences of speech features, which is just inverse with input and output of the seq2seq ASR. Tacotron [18] is one of the popular seq2seq TTS models. Fig. 2.6 shows the overview architecture. Given the input text consisting of character (or phoneme) sequences, it embeds it to character vector, and the character vector goes through the CBHG (1-D **C**onvolution **B**ank + **H**ighway network + bidirectional **G**RU). Then the CBHG module produces the final encoder state.

The attention module plays the role of bridging between encoder and decoder as we described in Sect. 2.2.1. On the decoder side, the recurrent neural network (RNN) such as GRU or LSTM produces a Mel-scale spectrogram, and then the decoder CBHG module converts it to a linear-scale log magnitude spectrogram.

The decoder can have a process that predicts the speech's end frame when it produces the Mel-scale spectrogram. It helps the Tacotron determine the end of the speech. The speech's end frame is decided by the binary prediction of the Mel-spectrogram and the context vector from the attention module. The

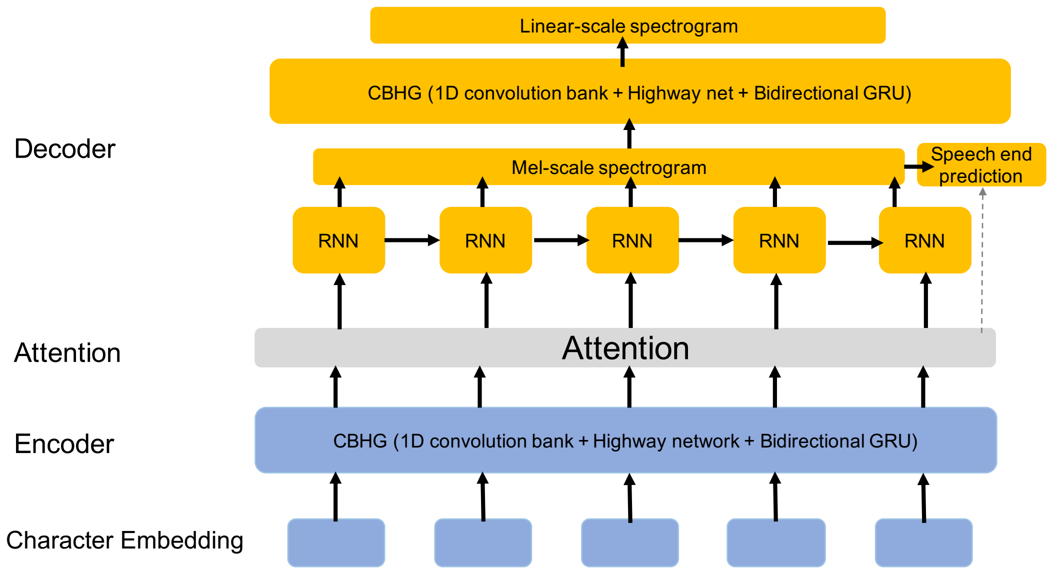


Figure 2.6: Tacotron-based TTS with speech's end frame prediction.

loss function for training TTS used a combination of mean squared error (MSE) in the Mel-spectrogram and MSE in the log magnitude spectrogram and binary cross-entropy in the prediction for the speech's end frame as follows:

$$\begin{aligned} L_{TTS} =& \frac{1}{T}\sum_{t=1}^{T}\{(m_t - \hat{m}_t)^2 + (r_t - \hat{r}_t)^2 && (2.13) \\ & - (b_t \log(\hat{b}_t) + (1-b_t)\log(1-\hat{b}_t))\}, && (2.14) \end{aligned}$$

where the first term of the summation is the MSE between target Mel-spectrogram m_t and predicted Mel-spectrogram $\hat{m}_t$ at time t, the second term is the MSE between target log magnitude spectrogram r_t and predicted log magnitude spectrogram $\hat{r}_t$ at time t, and the third term is the binary cross-entropy between the target probability end frame of speech b_t and predicted probability end frame of speech $\hat{b}_t$ at time t.

Since the original Tacotron is a single speaker model and cannot deal with multi-speakers, we generate speaker vectors with the DNN-based speaker recognition (SPKREC) DeepSpeaker [19] and take them into Tacotron. In DeepSpeaker, the DNN architectures extracted the frame features from the utterances. After converting the frame features to a speaker representation for an utterance unit, it

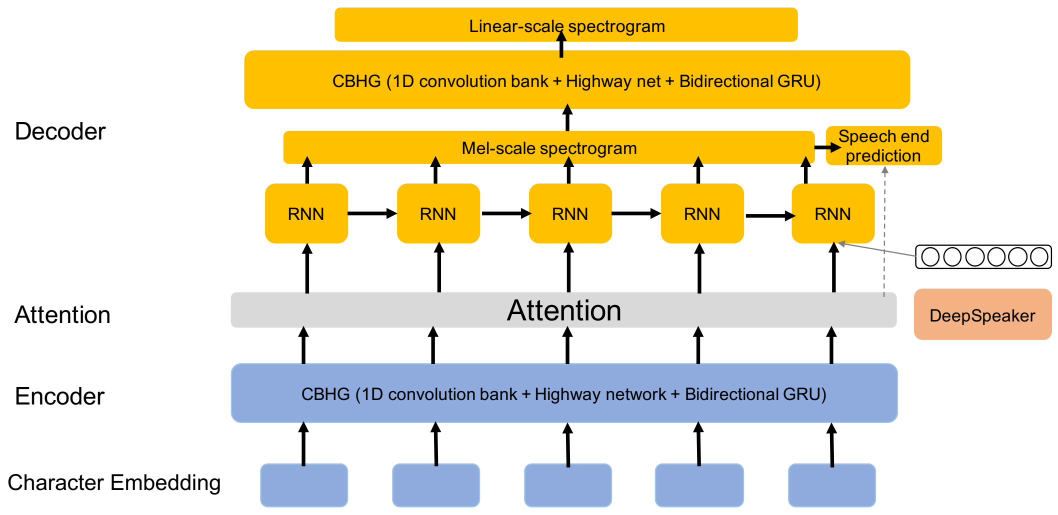


Figure 2.7: Tacotron-based TTS for multi-speakers.

is embedded into a vector representation. The embedding vectors are normalized to the unit norm and by cosine similarity between two embedding vectors:

$$cos(x_i, x_j) = x_i x_j, \tag{2.15}$$

where $\boldsymbol{x}_i$, and $\boldsymbol{x}_j$ are the embedding vectors.

Finally, the model is trained using the following loss function, which maximizes the cosine similarities of the embedding vectors from the same speaker while minimizing those from different speakers for N triplets:

$$L_{triplet} = \sum_{i=0}^{N} max((s_i^{an} - s_i^{ap} + \alpha), 0), \tag{2.16}$$

where s_i^{ap} is the cosine similarity between an utterance a of a speaker and another utterance p of the same speaker in triplet i. s_i^{an} is the cosine similarity between an utterance a of a speaker and an utterance n of another speaker in triplet i.

After the DeepSpeaker models are trained, we generate speaker embedding vector s. The generated speaker vector is used in the speaker embedding of the multi-speakers Tacotron (Fig. 2.7). The speaker vector is concatenated with the encoder output and goes through the decoder. In the loss function, we use the extension of Eq. (2.13) by adding the formula of speaker loss for handling multiple speakers as follows:

$$\begin{aligned} L_{TTSspeaker} = & \frac{1}{T}\sum_{t=1}^{T}\{\gamma_1((m_t - \hat{m_t})^2 + (r_t - \hat{r_t})^2) \\ & - \gamma_2((b_t \log(\hat{b_t}) + (1 - b_t)\log(1 - \hat{b_t})))\} \\ & + \gamma_3(1 - \frac{s \cdot \hat{s}}{\|s\|_2 \cdot \|\hat{s}\|_2}), \end{aligned} \tag{2.17}$$

where the first term is an MSE that compares target Mel-spectrogram m_t with predicted Mel-spectrogram $\hat{m_t}$ at time t, the second term is an MSE that compares target log magnitude spectrogram r_t with predicted log magnitude spectrogram $\hat{r_t}$ at time t, the third term is the binary cross-entropy comparing target speech's end probability b_t with predicted speech's end probability $\hat{b_t}$ at time t, and the last term is the cosine distance comparing target speaker vector s with predicted speaker vector $\hat{s}$. $\gamma_1, \gamma_2, \gamma_3$ are hyperparameters that adjust the balance among the three losses.

Chapter 3

Datasets

This chapter explains all of our datasets to use in our experiments. We also show how to construct synthetic code-switching.

Code-switching speech datasets for training ASR and TTS exist in some language pairs. For example, there are code-switching speech corpora of Hindi-English [20], Chinese-English [21,22], Cantonese-English [23], Frisian-Dutch [24], isiZulu-English [25]. However, this thesis focuses mainly on the language pairs consisting of Japanese, English, and Chinese. Most of their language pairs are unavailable or low-resource, so we utilize monolingual corpora to support them. We also augment the data by constructing the synthetic CS corpora from monolingual datasets.

3.1 Existing Corpora

In this section, we show the existing corpora that are available for our experiments. We first introduce the monolingual corpora for supporting low-resourced CS data and then the CS corpora.

3.1.1 Monolingual Corpora

BTEC

The ATR Basic Travel Expression Corpus (BTEC) [26,27] covers basic conversations in travel domains, such as sightseeing, restaurants, hotels, etc. Its sentences

Table 3.1: Basic statistics of BTEC text data [26].

	BTEC 1	**BTEC 2**	**BTEC 3**	**BTEC 4**
Number of sentences	172k	46k	198k	74k
Number of Japanese word tokens	1,174k	341k	1,434k	548k
Number of Japanese word types	28k	20k	43k	22k

were collected by bilingual travel experts from Japanese/English sentence pairs in travel domain phrasebooks. The BTEC has been translated into French, German, Italian, Chinese, and Korean. Table 3.1 lists the basic statistics of BTEC text data called BTEC 1, 2, 3, and 4. We synthesized the text using Google TTS [28].

LibriSpeech

LibriSpeech [29] is the open-source English read speech corpus. It is derived from free public domain LibriVox's audiobooks* read by volunteers, and most of the text is based on the Project Gutenberg†. This corpus contains approximately 1000 hours of speech, but we choose an officially prepared 100-hour subset with 251 speakers as our training data and use it for supporting SEAME data.

AISHELL-1

AISHELL-1 [30] is a read Mandarin speech corpus. Most of its speakers are from Northern China, and some are from Southern China, Guangdong-Guangxi-Fujian, and others. It contains 150 hours of speech recorded by 340 speakers for the training set. We use 100 hours of speech consisting of the same 340 speakers, and use it for supporting the SEAME data.

*https://librivox.org/
†http://www.gutenberg.org

Table 3.2: Statistics of SEAME corpus [31].

Subset	Speakers	Hours	Duration Ratio		
			Mandarin	English	CS
train	134	101.1	16%	16%	68%
dev_{man}	10	7.5	14%	7%	79%
dev_{sge}	10	3.9	6%	41%	53%

3.1.2 Code-switching Corpora

SEAME

SEAME [22] is a conversational Mandarin-English code-switching corpus, collected from Singaporean and Malaysian speakers. Previous works [31] prepared the subsets of *train*, dev_{man}, and dev_{sge}. The dev_{man} is a test set dominated by Mandarin words, and the dev_{sge} is a test set dominated by English words. Each subset contains monolingual Mandarin and monolingual English utterances not only code-switching utterances. Table 3.2 lists the statistics.

3.2 Code-switching Data Augmentation

Next, we show how to construct the augmented CS data.

3.2.1 Related Works

Since obtaining a large amount of data takes time and money, some researchers have utilized synthetic data to improve the quality of their systems. Jia et al. [32] used synthetic data and machine translation for improving end-to-end speech-to-text translation models. Hasegawa-Johnson et al. [33] trained image-to-speech models with SPEECH-COCO [34], a synthetic speech corpus generated by TTS. Synthetic data were also used for training ASR and TTS [10]. They conducted experiments with synthetic data as well as natural data, and both sets of results showed the same tendency of their proposed model to improve the ASR and TTS performances. Therefore, synthetic data can be utilized for covering low-resourced data.

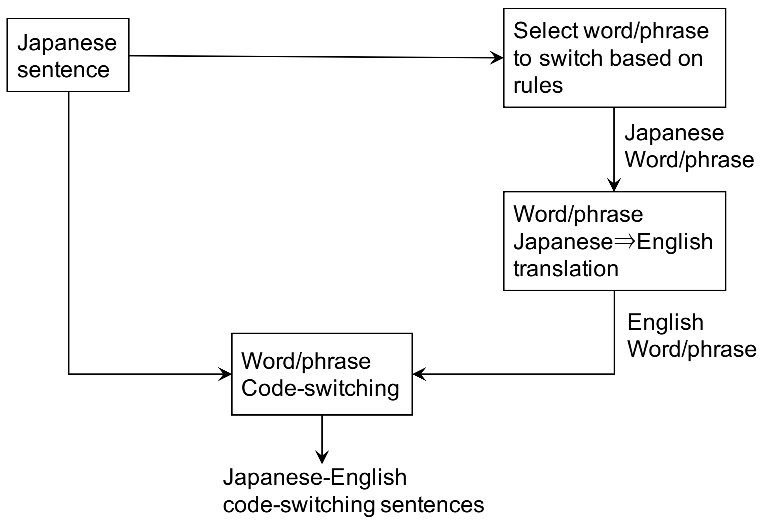

Japanese sentence
Select word/phrase to switch based on rules
Japanese Word/phrase
Word/phrase Japanese⇒English translation
English Word/phrase
Word/phrase Code-switching
Japanese-English code-switching sentences

Figure 3.1: Japanese-English CS text data construction [40].

CS is one of the low-resourced data. The existing corpora are limited to some language pairs and accents, and difficult to collect a new corpus of natural CS. Although we may find either CS speech or CS text in social media, the annotation for CS data requires high language skills. Therefore, some researchers actually utilized synthetic CS data to improve their CS system's quality [35, 36]. In the same way, we utilize synthetic data to cover low-resourced CS data.

For generating synthetic code-switching, there are some previous approaches. Gupta et al. [37] investigated the semi-supervised approach for generating code-switching data from the parallel sentences. Another approach is mBERT based method [38]. Based on the fine-tuning mBERT [39], it predicts words to switch in a monolingual sentence and generates the switched words from the parallel data. In this way, several approaches exist, but we adopt simpler methods utilizing machine translation and TTS.

From the transcriptions of BTEC corpus [26, 27], we created the synthetic speech CS utilizing machine translation and TTS. We also collected natural speech CS, which bilingual speakers created manually from the BTEC corpus. All of them are intra-sentential CS because intra-sentential CS is more challenging for CS ASR and the main target in our research. We constructed word-level and phrase-level intra-sentential CS.

3.2.2 Synthetic Speech Code-switching

Fig. 3.1 shows an overview of the construction of the CS text data [40], which shows the Japanese-English code-switching cases created from Japanese sentences. We first selected words or phrases from the BTEC text data based on the rules described later. Then we translated the chosen words or phrases using Google translation API. Finally, code-switching sentences were created by inserting the translated words or phrases into the original sentences. All the constructed CS text are synthesized using Google TTS [28]. The followings describe the switching rules of intra-sentential word-level CS, intra-sentential phrase-level CS I, and intra-sentential phrase-level CS II:

Intra-sentential Word-level CS

The switching positions for word-level CS is a noun. We chose just one noun word, translated it by machine translation, and inserted it into the original sentence. It produced intra-sentential word-level code-switching sentences. Table 3.3 shows the resulting examples of "JaEnCS" created from a Japanese sentence and "EnJaCS" made from an English sentence. After constructing the data, we investigated the percentage of English and Japanese words included in the code-switching. The investigation results on the datasets used in this thesis are shown in Table 3.4.

Table 3.3: Examples of the constructed intra-sentential word-level CS sentences.

JaEnCS	観光バスの pamphlet はありますか? (*Do you have any brochures for the sightseeing bus?*)
EnJaCS	The size of 人形 is about two-thirds the size of a real person. (*The size of a puppet is about two-thirds the size of a real person.*)

Table 3.4: Statistics of intra-sentential word-level code-switching.

	Utterances	Japanese words	English words
JaEnCS	51k	87%	13%
EnJaCS	21k	22%	78%

Table 3.5: Examples of the constructed intra-sentential phrase-level CS I.

JaEnCS	どこでガソリンを do you want to buy? (*Where do you want to buy gasoline?*)
JaZhCS	気象状況が思わしくありませんので请系好安全带. (*The weather condition is uncertain, so please fasten your seat belt.*)
EnJaCS	Did i fill out このカードでいいですか? (*Did i fill out this card, okay?*)
EnZhCS	Western-style beds are becoming 比以前普及多了. (*Western-style beds are becoming more common than before.*)

Table 3.6: Statistics of intra-sentential phrase-level CS I.

	Utterances	Japanese words	English words	Chinese words
JaEnCS	51k	48%	52%	0%
JaZhCS	51k	48%	0%	52%
EnJaCS	30k	30%	70%	0%
EnZhCS	51k	0%	46%	54%

Intra-sentential Phrase-level CS I

The switching position of intra-sentential phrase-level CS I for JaEnCS is a postpositional particle. In contrast with the word-level code-switching case, we selected phrases beyond the length of the loanword units after the postpositional particles appeared. Here, to produce natural conversations, we referred to the actual examples that were reported in existing studies [6], especially a switching pattern from Japanese-to-English phrases. The JaZhCS case is also chosen the postpositional particles as the switching position. The switching position for EnJaCS chose verb mainly, referring to the top switching positions of our collected natural speech CS (see Table 3.10). The phrases between the chosen position and period were translated by machine translation and inserted into the original sentence. The JaEnCS case is also chosen mainly verb as the switching position. They produced intra-sentential phrase-level code-switching. Table 3.5 shows examples of resulting sentences. The statistics of the intra-sentential phrase-level CS I used in this thesis are shown in Table 3.6.

Intra-sentential Phrase-level CS II

We created another intra-sentential phrase-level CS. The switching position is a comma. We selected phrases following a comma, translated them, and inserted them into the original sentences. The code-switching examples are shown in Table 3.7. From BTEC Japanese, English, and Chinese, we constructed an English-Japanese CS "EnJaCS," a Japanese-Chinese CS "JaZhCS," a Chinese-English CS "ZhEnCS." We also constructed an English-French CS "EnFrCS" from BTEC English and French. All of them are intra-sentential phrase-level CS. The statistics of the constructed intra-sentential phrase-level code-switching are shown in Table 3.8.

Table 3.7: Examples of intra-sentential phrase-level CS II.

EnJaCS	If this shirt doesn't fit, 取り替えてもらえますか.
JaZhCS	このシャツが体に合わなかったら, 可以换吗?
ZhEnCS	如果这件衬衫不合身, will you exchange it?
EnFrCS	If this shirt doesn't fit, allez-vous me l'echanger?

Table 3.8: Statistics of intra-sentential phrase-level CS II.

	Utterances	Words			
		Japanese	English	Chinese	French
EnJaCS	11k	70%	30%	0%	0%
JaZhCS	11k	38%	0%	62%	0%
ZhEnCS	11k	0%	57%	43%	0%
EnFrCS	10k	0%	40%	0%	60%

3.2.3 Natural Speech Code-switching

The natural speech CS is the intra-sentential phrase-level CS created by Japanese-English bilingual speakers. First, a bilingual speaker made the CS text from parallel Japanese-English sentences. Although he lives in an English-speaking country, he speaks with his Japanese family in Japanese and has studied in Japan for one year. Therefore, he often uses code-switching in his daily life. We gave him 1000 pairs of Japanese-English sentences from the BTEC, from which he made phrase-level CS sentences. After that, we asked another bilingual speaker to read and record the constructed natural CS text. He recorded in a quiet room.

Table 3.9 shows one example of the created CS text. In the whole words of 1000 created CS text, Japanese words have 3251 (24%), and English words have 10214 (76%). Of the total CS text, the CS starting from Japanese words (JaEnCS) is 57% and the CS starting from English words (EnJaCS) is 43%. Table 3.10 shows the statistics of the part-of-speech (POS) tags in the switching positions. In the JaEnCS, the postpositional particle is the majority of switching positions. The EnJaCS often switches at the noun and verb. Of the 1000 utterances created, 900 utterances were used for the training set, and 100 utterances were used for the test set.

Table 3.9: A natural CS sentence created from a pair of Japanese-English BTEC.

Japanese sentence	このごろ披露宴では花嫁さんが二度もお色直しをして，派手らしいですね．
English sentence	I hear that nowadays the bride changes her clothes as often as twice during the reception and that the reception is luxurious.
Result CS	このごろ披露宴では花嫁さん changes her clothes as often as twice during the reception and that the reception is 派手らしいですね．

Table 3.10: The statistics of the created CS switching positions.

JaEnCS		EnJaCS	
POS	Ratio(%)	POS	Ratio(%)
Postpositional particle	77	Noun	33
Noun	12	Verb	15
Auxiliary verb	5	Conjunction	15
Verb	2	End punctuation	12
Adverb	2	Preposition	6
Conjunction	1	Adverb	6
End punctuation	1	Determiner	6
		Adjective	4
		Pronoun	3

3.3 Feature Extraction

We sampled all the speech signals at a sampling rate of 16kHz. Then we applied pre-emphasis and normalized the speech signals between -1 and 1. We extracted the spectrogram features using a Short-Time Fourier Transform (STFT) with the Librosa library [41]. The frame had a 50-ms length and a 12.5-ms shift, and the FFT points are 2048. From the spectrogram, we computed the magnitude spectrogram and mapped it to the Mel-scale spectrogram. Those features were transformed to log-scale and normalized into 0 mean and unit variances. Finally, we got 80 dimensions of log Mel-spectrogram features and 1025 dimensions of log magnitude spectrograms.

Chapter 4

Proposed Language-aware Code-switching ASR

This chapter is about the language-aware code-switching ASR, where we compare the proposed LID approach with the other LID approaches.

4.1 Introduction

Several studies have addressed ASR for the CS of specific language pairs, such as Mandarin-English [42–44], English-Malay [45], and Frisian-Dutch [46]. However, most previous works only focused on single-pair CS, and multi-pair CS is still difficult. Compared to developing a system per single-pair CS, the unified system for multi-pair CS can simplify the process of training, deploying, maintaining, and the recognition task. Moreover, the same language composing CS can be expected to improve the performance of multi-pair CS each other. In this work, to solve the multi-pair CS problem, we incorporate language identification (LID), which predicts the language ID. It can be expected to enable to handle multi-pair CS better by providing language information and preventing the predicting error, which is caused by failing to predict the switching time and recognizing speech chunk as the inappropriate language. In this chapter, we investigate the best approach for the LID system.

4.2 Related Works

The LID has two types. One predicts the language per utterance for identifying the language of multilingual or inter-sentential CS utterances. Another identifies the language per token within an utterance for intra-sentential CS.

The LID system per utterance mainly uses the cascade approach. It first predicts what language is per utterance and subsequently sends the input speech into the corresponding monolingual's ASR. There are several methods for identifying the language. For example, phone recognition followed by language modeling (PRLM) and parallel phone recognition followed by language modeling (PPRLM) are proposed [47]. The PRLM first recognizes the phones from the speech features using a given language's phoneme recognizer. Then based on the phoneme sequences, the likelihood score is calculated using the n-gram language models to hypothesize the language. In the case of PPRLM, it uses multiple languages' phoneme recognizers in parallel. Another method is the prosody-based LID. It identifies the language by the prosodic speech features such as using i-vector [48]. Recently, DNN-based LID is shown that it has better performance than the i-vector-based LID system [49]. In any case, those methods are used only for the cascade approach.

For the LID within an utterance, Lyu et al. compared the cascade approach with the direct approach [50]. The direct approach only uses one model trained with CS data. The cascade approach for the CS task identifies the language boundary at first, and then the identified monolingual fragments are recognized by the corresponding monolingual ASRs. The comparison result is that the direct approach is superior to the cascade approach since the un-recover LID error damages the cascade approach. Therefore, the current primary approach for CS ASR is the direct approach. However, the language information is useful additional information. It can prevent the predicting error caused by failing to predict the switching time and recognizing speech chunks as inappropriate language. Therefore, the LID is used as the cascade or the direct approach. As the cascade approach, Li et al. multiply the ASR posteriors with the loss of LID model during decoding [51]. As the direct approach, the approach using the tag of language ID [52] and the approach using multi-task learning are proposed [31, 44].

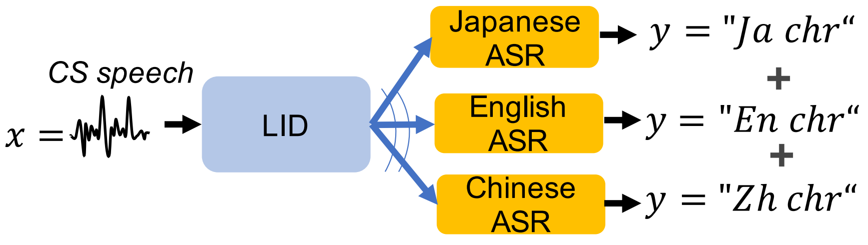
Our proposed LID system adopts the multi-task approach using two softmax layers. Moreover, we investigate the best approach for language identification by comparing it with other approaches.

4.3 Proposed Approaches

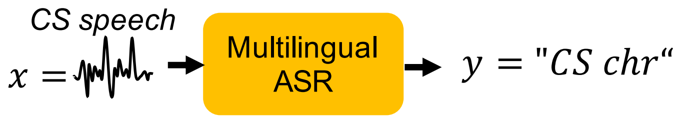
This section explains the details of several LID approaches used in our experiments, including the proposed LID system. Those approaches are mainly classified into two types: Cascade and Direct approaches. Direct approach and Cascade approach are already appeared in Sect. 4.2 and explained shortly, but we clearly distinguish the Cascade approach and the Direct approach from multi-task and single-task here. The Cascade approach uses multiple monolingual ASRs, and there are multiple passes connected to ASRs from LID, as Fig. 4.1 (a) shows. On the other hand, the Direct approach uses one multilingual ASR model, and the number of passes connected to ASR is one, as Fig. 4.1 (b) shows. Therefore, the number of ASRs is the main difference between the Direct approach and the Cascade approach. On the other hand, single-task and multi-task learning refer to the number of tasks within one model. Single-task learning has a single task, such as predicting character sequences. The multi-task has multiple tasks, such as predicting character sequences and language sequences using two decoders. The proposed LID system adopts the multi-task approach using two softmax layers with a Direct approach. Then we describe the detail of the LID approaches one by one.

4.3.1 Cascade Approaches

The Cascade approach is the traditional approach, which identifies the language and then selects the corresponding language's monolingual ASR. It uses multiple models trained separately with monolingual data. Here, the LID output is text, but ASR input is the speech based on the LID results. So, the LID needs to estimate speech length for the input speech of subsequent monolingual ASR. For that, we adopt the frame-level LID estimating language ID per speech frame since the speech consists of several frames. The input is each speech frame, but it is difficult to predict language ID with only one frame, so frames within the index range of plus-minus i are also inputted together, as Fig. 4.3 shows. For example,



(a)



(b)

Figure 4.1: Overview of the (a) Cascade approach using multiple monolingual ASRs and (b) Direct approach using one multilingual ASR.

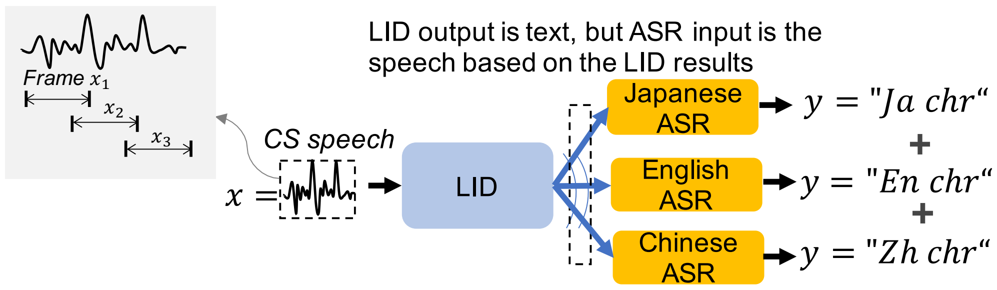


Figure 4.2: A Cascade approach using the frame-level LID. Given CS speech, the LID predicts language ID per frame, and the monolingual speech fragments are estimated based on the LID results. Given the speech fragments, the monolingual ASRs generate the transcription. Finally, those monolingual transcriptions form the CS transcription.

when it predicts the language ID y_{11} from x_{11} speech, if we select plus-minus ten as the frame index range, the input is 21 frames from x_1 to x_{21}. Fig. 4.2 shows the overview of the Cascade approach.

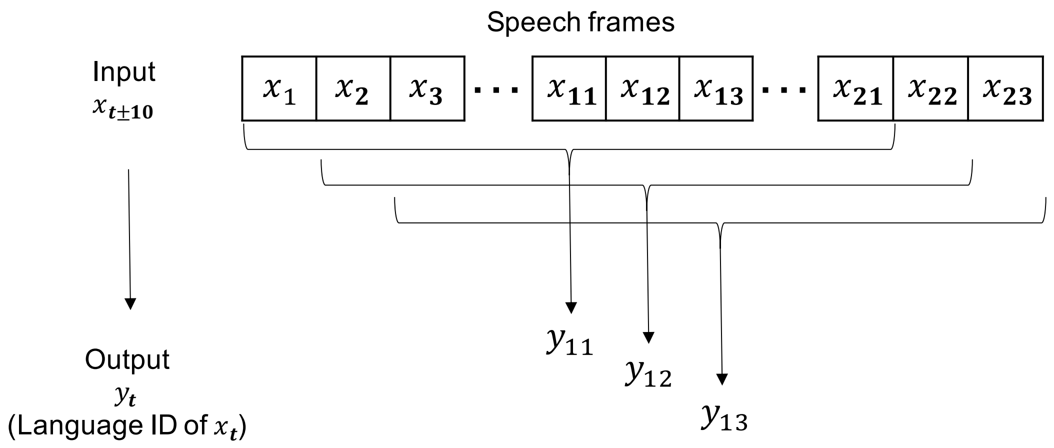


Figure 4.3: Example of input and output in the LID of the Cascade approach. When it predicts the language ID y_{11} from x_{11} speech, if we select plus-minus ten as the frame index range, the input is 21 frames from x_1 to x_{21}.

Cascade MLP LID+ASRs

This approach uses a multilayer perceptron (MLP) [53] for the LID model in the Cascade system. The MLP is one of the most common neural network models. The basic structure comprises an input layer to receive the input features, an output layer that predicts the target from the input, and a hidden layer between those two.

The loss function is cross-entropy as follows:

$$L_{MLP} = -\sum_{n=1}^{N} 1_{\hat{y}_n \in y_n} \log{(p_{\hat{y}_n, y_n})}, \tag{4.1}$$

where N is the number of class languages (JA, EN, ZH); $p_{\hat{y}_n, y_n}$ is the probability of the predicted output $\hat{y}_n$ belonging to the target y_n.
The monolingual ASRs use the basic single-task ASR we already described in Sect. 2.2.2, so the loss function is Eq. (2.12).

Cascade LSTM LID+ASRs

This approach uses Long-short term memory (LSTM) [54] for the LID system of Cascade approach. LSTM is a kind of Recurrent Neural Network (RNN), which can consider the previous outputs with iterating process recursively, and LSTM can find and exploit long-range context better.

The loss function for LSTM LID is also cross-entropy as follows:

$$L_{LSTM} = -\sum_{n=1}^{N} 1_{\hat{y}_n \in y_n} \log{(p_{\hat{y}_n, y_n})}, \tag{4.2}$$

where N is the number of class languages (JA, EN, ZH); and $p_{\hat{y}_n, y_n}$ is the probability of the predicted output $\hat{y}_n$ belonging to the target y_n.
The monolingual ASRs use the basic single-task ASR we already described in Sect. 2.2.2, so the loss function is Eq. (2.12).

4.3.2 Direct Approaches

Direct ASR (No LID)

The Direct ASR (No LID) approach uses only one multilingual model trained with multiple monolingual data. It also does not incorporate any language identification. The input is speech sequence per utterance. The output style is a character (a, b, c, . . . , z). The overview and the output example are shown in Fig. 4.4.

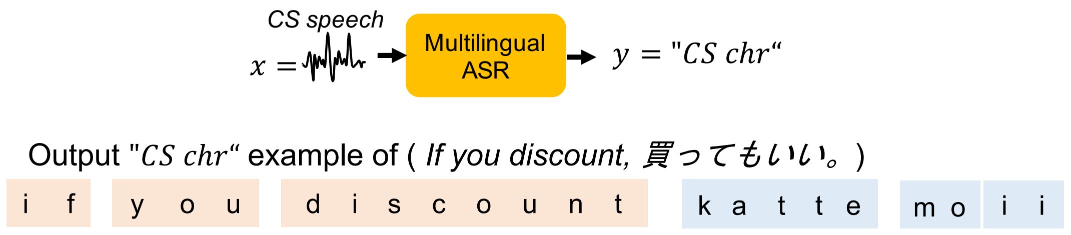


Figure 4.4: Overview of the Direct ASR (no LID) approach. Given CS speech, it hypothesizes CS character sequences without identifying the language.

The ASR use the basic single-task ASR we already described in Sect. 2.2.2, so the loss function is Eq. (2.12).

Direct ASR (LngChr)

The Direct ASR (LngChr) approach also uses only one multilingual model, and the input is speech sequence per utterance, but it outputs the language information with a character together like (JA-a, JA-b, JA-c, ..., JA-z, EN-a, EN-b, EN-c, ..., EN-z, ZH-a, ZH-b, ZH-c, ..., ZH-z). The overview and the output example are shown in Fig. 4.5.

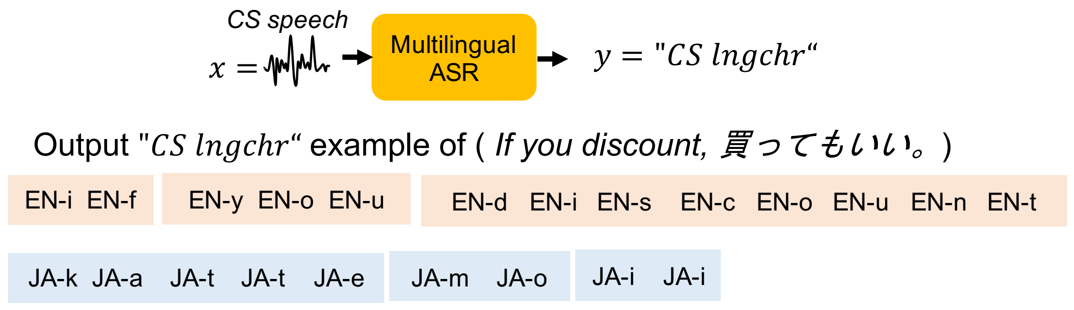


Figure 4.5: Overview of the Direct ASR (LngChr) approach. Given CS speech, it outputs the language information with a CS character together like “JA-a, JA-b, ..., ZH-z.”

The ASR use the basic single-task ASR we already described in Sect. 2.2.2, so the loss function is Eq. (2.12).

Direct ASR (TagLID)

The Direct ASR (TagLID) approach also uses only one multilingual model, and the input is speech sequence per utterance, but it predicts language ID followed by the corresponding character sequence. For example, it predicts language ID tag “[EN]” followed by English character sequence. The overview and the output example are shown in Fig. 4.6.

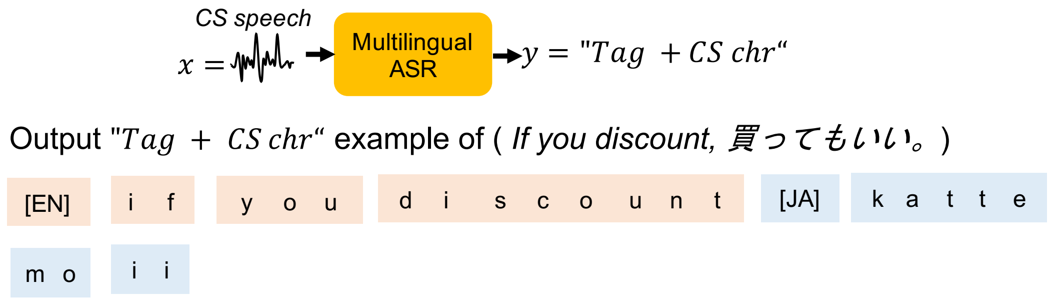

CS speech
$x =$
Multilingual ASR
$y =$ "Tag + $CS\ chr$"
Output "Tag + $CS\ chr$" example of (If you discount, 買ってもいい。)
[EN]
i f
y o u
d i s c o u n t
[JA]
k a t t e
m o
i i

Figure 4.6: Overview of the Direct ASR (TagLID) approach. Given CS speech, it predicts language ID tag like “[EN]” every switching position, followed by the corresponding monolingual character sequences.

The ASR use the basic single-task ASR we already described in Sect. 2.2.2, so the loss function is Eq. (2.12).

Direct ASR (Proposed LID)

The Direct ASR (Proposed LID) approach also uses only one multilingual model, and the input is speech sequence per utterance, but it outputs the language information (JA, EN, ZH) and character sequences (a, b, c, . . . , z), respectively. This architecture has the advantage of dictionary size, which is smaller than LngChr and TagLID approaches. The overview and the output example are shown in Fig. 4.7.

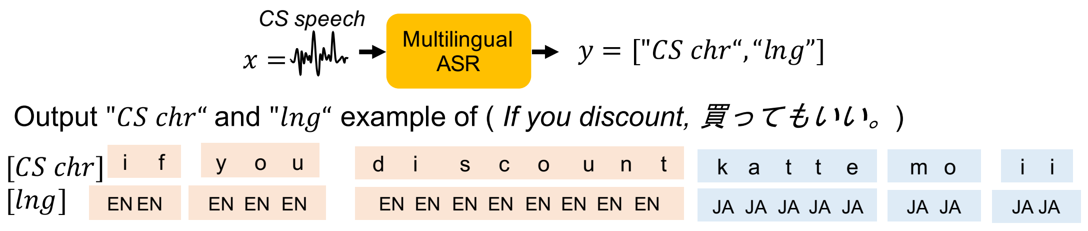


Figure 4.7: Overview of the Direct ASR (Proposed LID) approach. Given CS speech, it outputs CS character sequences and language ID sequences using two softmax layers.

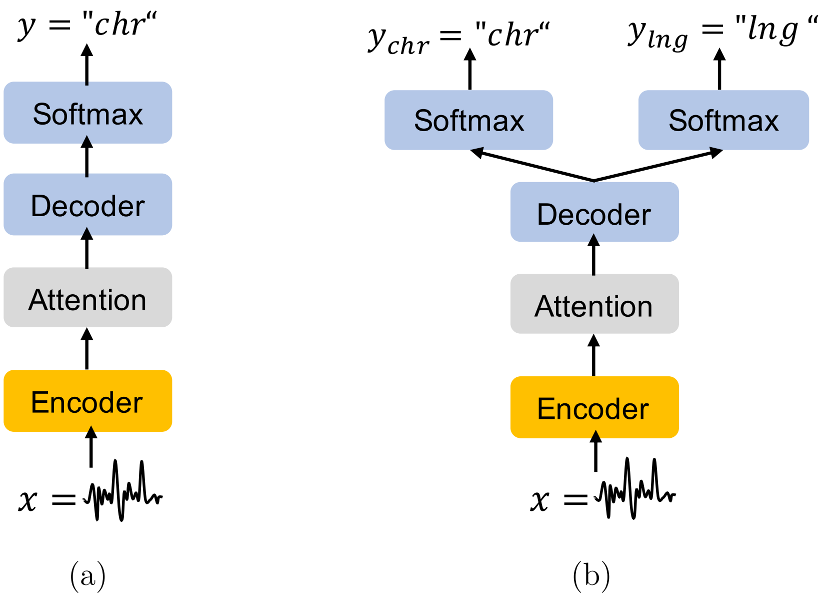


Figure 4.8: Overview of the (a) single-task ASR predicting only character sequences and (b) multi-task ASR predicting both the language ID sequences and character sequences with two softmax layers.

This architecture uses softmax-level multi-task learning. We show the difference with basic single-task learning. The basic single-task learning is the ASR we already described in Sect. 2.2.2. The simplified figure is shown in Fig. 4.8 (a). The ASRs of the basic single-task learning are used in the Cascade MLP LID+ASRs, Cascade LSTM LID+ASRs, Direct ASR (No LID), Direct ASR (LngChr), Direct ASR (Tag LID). On the other hand, the Direct ASR (Proposed LID) adopts the multi-task architecture using two softmax layers shown in Fig. 4.8 (b). Using softmax-level multi-task learning, it outputs the language information (JA, EN, ZH) and character sequences (a, b, c, . . . , z) separately.

The loss function for optimizing ASR (Eq. (2.12)) changes to the following function by the incorporating LID loss:

$$L_{ASR}^{LngAwr} = \lambda_{Chr} L_{ASR}^{Chr} + \lambda_{Lng} L_{ASR}^{Lng}, \tag{4.3}$$

where it's a summation of two cross entropy, tuning those weights by the hyperparameters λ_{Chr} and λ_{Lng}. It can adjust the balance between LID training and

Table 4.1: Statistics of datasets for a single-pair CS (synthetic speech; single speaker).

Subset			Hours	Utterances
Train	Mono	Ja25k (JaTTS)	22.2	25000
		En25k (EnTTS)	17.5	25000
	CS	JaEnCS20k (Ja+MixTTS)	18.4	20000
Test	Mono	TstJa(JaTTS)	0.7	500
		TstEn(EnTTS)	0.6	500
	CS	TstJaEnCS(MixTTS)	0.7	500

character training using loss hyperparameters.

4.4 Experiments

4.4.1 Experimental Settings

We conducted the experiments on the following three scenarios: single-pair CS (synthetic speech; single speaker), single-pair CS (natural speech; multi-speaker), and multi-pair CS.

Dataset Composition

For the experiment on single-pair CS (synthetic speech; single speaker), we used the intra-sentential word-level CS and phrase-level CS I of synthetic speech CS (Sect. 3.2.2). From the synthetic speech CS, we chose 20-k sentences for training sets and 500 sentences for test sets. They are denoted as JaTTS if synthesized using Japanese TTS, as EnTTS synthesized using English TTS, and as MixTTS if synthesized both using Japanese TTS and English TTS. Table 4.1 shows the statistics of these datasets for a single-pair CS (synthetic speech; single speaker). We used the JaEnCS20k (Ja+MixTTS) dataset for training the LID models of Cascade approaches and Direct ASR models and used the Ja25k (JaTTS) and En25k (EnTTS) for training ASRs of Cascade approaches.

For the experiments on single-pair CS (natural speech; multi-speaker), we

Table 4.2: Statistics of datasets for a single-pair CS (natural speech; multi-speaker).

Subset			Speakers	Hours	Utterances
Train	Mono	SEAME EN	134	15.8	21435
		SEAME ZH	134	15.8	21476
	CS	SEAME CS	134	69.6	51027
Test	dev_{man}	*mono*	10	1.6	2228
		CS	10	5.9	4303
		all	10	7.5	6531
	dev_{sge}	*mono*	10	1.8	3156
		CS	10	2.1	2165
		all	10	3.9	5321

used SEAME CS corpus (Sect.-3.1.2). Table 4.2 shows the statistics for a single-pair CS (natural speech; multi-speaker). We used the SEAME CS to train the LID models of Cascade approaches and Direct ASR models and used the SEAME EN and SEAME ZH to train ASRs of Cascade approaches. We evaluated them on dev_{man} and dev_{sge} test sets.

For the experiments on multi-pair CS, we used the BTEC (Sect. 3.1.1) and intra-sentential phrase-level CS II of synthetic speech CS (Sect. 3.2.2) and the natural speech CS (Sect. 3.2.3). From the BTEC corpus, we randomly selected 25-k Japanese utterances, 25-k English utterances, and 25-k Chinese utterances, and 500 sentences for each test set. From the synthetic speech CS, we used 10-k sentences for each training set. We also selected 500 sentences for each test set. They are synthesized using the corresponding language's TTS, where Japanese words are synthesized with Japanese TTS, English words are synthesized with English TTS, and Chinese words are synthesized with Chinese TTS. From the natural speech CS, we divided into 0.2k labeled data, 0.7k unlabeled data denoted as "NatEnJaCS," and 0.1k test data denoted as "TstNatEnJaCS." The labeled data were also divided between Japanese and English, which were used for monolingual data as "NatJa" and "NatEn." Table 4.3 shows the statistics of these datasets. We investigated two scenarios using Ja25k+En25k+Zh25k and EnJaCS10k+JaZhCS10k+NatEnJaCS0.7k. On the Ja25k+En25k+Zh25k scenario,

Table 4.3: Statistics of datasets for multi-pair CS.

Subset			Hours	Utterances
Train	Mono	Ja25k	29.2	25000
		En25k	22.7	25000
		Zh25k	26.9	25000
		NatJa0.2k	0.2	200
		NatEn0.2k	0.2	200
	CS	EnJaCS10k	11.8	10000
		JaZhCS10k	10.4	10000
		ZhEnCS10k	10.9	10000
		NatEnJaCS0.7k	1.1	700
Test	Mono	Ja	0.9	500
		En	0.7	500
		Zh	0.8	500
	CS	EnJaCS	0.8	500
		JaZhCS	0.7	500
		ZhEnCS	0.7	500
		TstNatEnJaCS	0.2	100

we used Ja25k+En25k+Zh25k for training Direct ASRs and Ja25k, En25k, Zh25k for training Cascade ASRs, and EnJaCS10k+JaZhCS10k+ZhEnCS10k for training Cascade LID models. On the EnJaCS10k+JaZhCS10k+NatEnJaCS0.7k scenario, we used EnJaCS10k+JaZhCS10k+ZhEnCS10k+NatEnJaCS0.7k for training Direct ASRs and Cascade LID models, and used Ja25k+NatJa9.2k, En25k+NatEn0.2k, Zh25k for training Cascade ASRs.

All the text characters were converted to lowercase letters, and punctuation marks [, : ? .] were removed. We converted all characters into the lowercase alphabet. For Japanese words, we applied a morphological analyzer Mecab [55] to convert into katakana. Then we converted the katakana into English letters with pykakasi [56]. We also used pypinyin to the Chinese characters [57] and converted them into pinyin. The text not including any language identifier consists of 26 letters (a-z), one mark (-) for stretching Japanese sounds, and three tags that denote the start of sentences (<s>), the end of sentences (</s>), and the spaces between words (<spc>). In the case of LibriSpeech, AISHELL-1, and SEAME, we utilized PASM sub-word units, which is a sub-word unit optimized for accents by taking alignments between phonemes and characters [58].

Model Details

Through all of our experiments in this chapter, our ASR system is an attention-based encoder-decoder model [16] described in Sect. 2.2.2. We used the model, which consists of three stacked BiLSTM encoders, a single layer LSTM, and multilayer perceptron (MLP)-based attention [15] components. The activation function is a LeakyReLU ($l = 1e - 2$) [59]. We did not use the language model.

The MLP of MLP LID+ASRs was set the hyperparameters with 100 hidden layer sizes, Relu activation function, Adam solver, and 200 max iterations. The LSTM of LSTM LID+ASRs used one hidden layer with 128 hidden units. We used frames within the plus-minus ten index range for the LID input of the Cascade approaches. In this chapter, we used 0.1 as λ_{Lng} value and 0.9 as λ_{Chr} value of the loss hyperparameters for the Direct ASR (proposed LID). We implemented all the models with the PyTorch python library [60]. For the MLP, we used the MLPClassifier from the scikit-learn library [61].

4.4.2 Results

Results on Single-pair Code-switching

(Synthetic Speech; Single Speaker)

We investigated the performances of systems on single-pair CS (synthetic speech; single speaker). First, we checked the LID performances on diarization error rate (DER) [62]. The DER is the metric that is used for speaker diarization experiments. It can measure the time ratio allocated to the wrong speaker and the time ratio that is incorrectly detected or not detected speech. We calculate the following formula:

$$DER = \frac{\text{false alarm} + \text{missed detection} + \text{confusion}}{\text{total}}, \tag{4.4}$$

Table 4.4: LID performance comparison on a single-pair CS (synthetic speech; single speaker) between Cascade approaches and Direct approaches in DER%. Direct ASR (proposed LID) was trained with $\lambda_{Lng} = 0.1$ and $\lambda_{Chr} = 1 - \lambda_{Lng}$.

	Monolingual		Code-switching
Train: JaEnCS20k(Ja+MixTTS)	**TstJa (JaTTS)**	**TstEn (EnTTS)**	**TstJaEnCS (MixTTS)**
Cascade MLP LID+ASRs	8.5	6.4	3.4
Cascade LSTM LID+ASRs	9.7	6.4	2.9
Direct ASR (No LID)	-	-	-
Direct ASR (LngChr)	13.8	26.4	6.1
Direct ASR (Tag LID)	16.6	34.0	14.2
Direct ASR (Proposed LID)	11.5	23.0	4.3

where the false alarm is the duration of non-speech incorrectly detected speech, the missed detection is the duration of speech incorrectly undetected speech, the confusion is the length assigned to the wrong speaker, and the total is the total length of the reference. In the case of our LID evaluation, we calculated as follows: the false alarm is the language ID token length detected beyond the reference length, the missed detection is the length detected nothing despite being within the reference length, the confusion is the length assigned to the wrong language ID, and the total is the total length of the reference. The higher DER, the better.

Table 4.4 shows the DER results. The LID performances of Cascade models were good since it was an easier task with one-to-one classification. The Direct ASR models were not better than the Cascade models on LID performances, but the Direct ASR (Proposed LID) was still better among the Direct ASR models. The Direct ASR models' performance on TstEn was not good since we only used CS data for training. However, the performance on TstJa was good as half of the CS data used for training was synthesized using Japanese TTS for the whole sentences, including English words. We were also surprised that the TstEn of the Cascade models were not influenced by not including English training data. It seems to be because the Cascade LID models train to predict language ID from partial speech feature, where it did not matter whether the whole sequence is CS or monolingual English utterance.

Table 4.5: Comparison ASR performances on a single-pair CS (synthetic speech; single speaker) between Cascade approaches and Direct approaches in CER%. Direct ASR (proposed LID) was trained with $\lambda_{Lng} = 0.1$ and $\lambda_{Chr} = 1 - \lambda_{Lng}$. The p-values compared to the Cascade MLP LID+ASRs for statistical significance are presented using ***, **, * and no-star ($^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$).

	Monolingual		Code-switching
Train: JaEnCS20k(Ja+MixTTS)	**TstJa (JaTTS)**	**TstEn (EnTTS)**	**TstJaEnCS (MixTTS)**
Cascade MLP LID+ASRs	41.2	39.0	30.2
Cascade LSTM LID+ASRs	38.6***	33.4***	28.0**
Direct ASR (No LID)	6.3***	47.6	7.8***
Direct ASR (LngChr)	11.4***	48.1	6.9***
Direct ASR (Tag LID)	11.9***	61.5	13.3***
Direct ASR (Proposed LID)	5.8***	40.8	6.3***

Table 4.5 shows the character error rate of the ASR performances. The character error rate is calculated by how many characters were mistaken. The smaller, the better. The p-value shows the result of the statistical significance test compared with one of the traditional approaches, Cascade MLP LID+ASRs. We assessed the statistical significance by a matched-pair sentence-segment word error test [63]. The Cascade models' performances were not good, although the LID performance was good. Identifying the language for Cascade approaches seems to be more difficult than for the Direct approach. It has to estimate language per speech frame, which has much longer sequences than per character. The LID error of the Cascade approach can cause more damage to the ASR than the Direct approach since the wrong ASR cannot recognize at all, and even the right ASR cannot recognize well if the input speech mistakenly starts from the middle of words. The performance on TstEn was better than the Direct approaches since the LID performances were good.

Compared between Cascade models, the Cascade LSTM LID+ASRs was better than the Cascade MLP LID+ASRs. The mistakes of Cascade MLP LID+ASRs occurred at the frame unit, so that it increased the number of switching times and made the subsequent ASR difficult to predict. On the other hand, the Cascade LSTM LID+ASRs could avoid the frame unit error since it included the previous

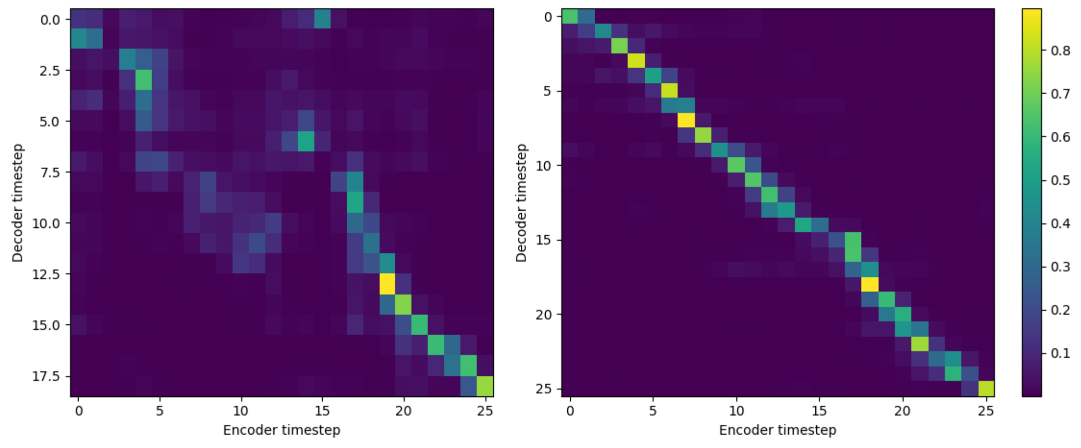


Figure 4.9: Attention alignment matrix between encoder and decoder by (a) Direct ASR (Tag LID) and (b) Direct ASR (Proposed LID), where the source is "最初にbeerをください (*I'll start with a beer*)."

class output in deciding the current class output.

Compared among Direct approaches, with additional language information, Direct ASR (Proposed LID) was better than Direct ASR (No LID). The Direct ASR (Proposed LID) was better than the Direct ASR (LngChr) as well. Direct ASR (LngChr) needs more dictionary sizes. The dictionary size of Direct ASR (No LID) is 30, where the text consists of 26 letters (a-z), one mark (-) for stretching Japanese sounds, and three tags that denote the start of sentences (<s>), the end of sentences (</s>), and the spaces between words (<spc>). The dictionary size of Direct ASR (Tag LID) is 33 by adding the language tags ([JA], [EN], [ZH]). The dictionary size of Direct ASR (Proposed LID) is 30, the same dictionary size as the Direct ASR (No LID), although it has another dictionary for LID with six sizes. On the other hand, the Direct ASR (LngChr) has bigger dictionary sizes with 82 sizes, including three times of 26 letters (a-z) by adding language identifier. Therefore, the Direct ASR (LngChr) got more difficult to correctly predict the character among more candidates, which degraded the performance. The Direct ASR (Tag LID) got more difficulty with attention alignments since it predicted the language ID before the target sequences. Fig. 4.9 compares the

Table 4.6: LID performance comparison on a single-pair CS (natural speech; multi-speaker) between Cascade approaches and Direct approaches in DER%. Direct ASR (proposed LID) was trained with $\lambda_{Lng} = 0.1$ and $\lambda_{Chr} = 1 - \lambda_{Lng}$.

	dev_{man}			dev_{sge}		
Train: SEAME CS	*mono*	*CS*	*all*	*mono*	*CS*	*all*
Cascade MLP LID+ASRs	28.5	34.3	32.3	30.8	34.5	32.3
Cascade LSTM LID+ASRs	27.5	28.3	28.0	29.5	29.3	29.4
Direct ASR (No LID)	-	-	-	-	-	-
Direct ASR (LngChr)	32.1	23.3	26.3	21.7	25.2	23.1
Direct ASR (Tag LID)	38.0	21.3	27.0	24.2	23.1	23.8
Direct ASR (Proposed LID)	33.6	21.0	25.3	24.6	22.4	23.7

attention alignments between by Direct ASR (Tag LID) and by Direct ASR (Proposed LID). The Direct ASR (Tag LID) could not take alignment well due to the insertion of the language ID tag. On the other hand, the Direct ASR (Proposed LID) could take alignment better due to omitting the unnecessary tokens within the sequence. Therefore, the Direct ASR (Proposed LID) tended to have the best performance.

(Natural Speech; Multi-speaker)
Table 4.6 shows the DER of CS ASR on single-pair CS (natural speech). Unlike the case using synthetic speech, the DER of Cascade approaches was higher than the Direct approaches. It seemed to be difficult to predict the language ID per speech frame on the natural CS data. It may be because the sound between words affected each other even beyond the language switching points on the natural CS data. Among Direct approaches, Direct ASR (Proposed LID) is better than Direct ASR (LngChr) and Direct ASR (Tag LID).

Table 4.7 shows the ASR performance results of the single-pair CS (natural speech) language-aware CS ASR. For the evaluation matrix, we used token error rate (TER). It is calculated by the Word Error Rate (WER) for English and the Character Error Rate (CER) for Mandarin. It is frequently adopted for evaluating the ASR of Mandarin-English CS because it is not affected by segmentation algorithms. As Table 4.7 shows, the performances of Cascade approaches are not good. It got more difficult by the increased LID error. Then in Direct approaches,

Table 4.7: Comparison ASR performances on a single-pair CS (natural speech; multi-speaker) between Cascade approaches and Direct approaches in TER%. Direct ASR (proposed LID) was trained with $\lambda_{Lng} = 0.1$ and $\lambda_{Chr} = 1 - \lambda_{Lng}$. The p-values compared to the Cascade MLP LID+ASRs for statistical significance are presented using ***, **, * and no-star ($^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$).

	dev_{man}			dev_{sge}		
Train: SEAME CS	*mono*	*CS*	*all*	*mono*	*CS*	*all*
Cascade MLP LID+ASRs	81.6	82.1	82.0	90.7	87.6	88.9
Cascade LSTM LID+ASRs	74.5*	74.6*	74.6*	87.8	80.6***	83.6**
Direct ASR (No LID)	37.2***	27.8***	29.6***	50.8***	36.9***	42.7***
Direct ASR (LngChr)	37.4***	28.5***	30.2***	51.3***	38.2***	43.7***
Direct ASR (Tag LID)	39.6***	27.6***	29.9***	51.6***	36.8***	43.0***
Direct ASR (Proposed LID)	36.4***	27.3***	29.1***	51.0***	36.2***	42.3***

the Direct ASR (LngChr) is comparatively good on the *mono* test set but not good on the *CS* test set, so in total not better than other Direct approaches. The Direct ASR (Tag LID) seems to suffer less from attention alignments than on single-pair CS (synthetic speech; single speaker). However, it is still not better than the Direct ASR (Proposed LID). The Direct ASR (Proposed LID) is better than the Direct ASR (No LID) and tends to have the best performance.

Results on Multi-pair Code-switching

Next, we confirmed the experimental results on multi-pair CS. At first, we compared the DERs among approaches. As Table 4.8 shows, the Cascade approach could predict better than the Direct approach (but even 0.4% DER made mistakes 901 times). However, Cascade MLP LID+ASRs mistakes occurred at the frame unit to increase the number of switching times. For example, as Fig. 4.10 shows, the reference has three straight English language IDs and three straight Japanese language IDs, so the number of switching times is one time, but the hypothesis inserted the wrong language ID. The number of switching times increased to 5 times. On the other hand, the Cascade LSTM LID+ASRs seemed to avoid the frame unit error since it included the previous class output in deciding the current class output.

Then we investigated the character error rate in Table 4.9. The result shows

Table 4.8: LID performance comparison on a multi-pair CS between Cascade approaches and Direct approaches trained with monolingual data in DER%. Direct ASR (proposed LID) was trained with $\lambda_{Lng} = 0.1$ and $\lambda_{Chr} = 1 - \lambda_{Lng}$.

Train:	Monolingual			Code-switching		
Ja25k+En25k+Zh25k	**Ja**	**En**	**Zh**	**EnJaCS**	**JaZhCS**	**ZhEnCS**
Cascade MLP LID+ASRs	0.4	5.8	8.2	2.9	5.8	4.9
Cascade LSTM LID+ASRs	0.5	5.9	8.7	2.8	5.6	4.7
Direct ASR (No LID)	-	-	-	-	-	-
Direct ASR (LngChr)	1.9	2.0	0.7	10.9	20.2	14.2
Direct ASR (Tag LID)	2.1	1.6	0.6	25.5	29.3	27.3
Direct ASR (Proposed LID)	1.4	1.6	0.7	10.1	13.0	9.0

that the Cascade LSTM LID+ASRs was better than Cascade MLP LID+ASRs. Although the language prediction of Cascade MLP LID+ASRs has similar performance with the Cascade LSTM LID+ASRs, the mistakes of Cascade MLP LID+ASRs occurred at the frame unit, so that it increased the number of switching times and made the subsequent ASR difficult to predict. On the other hand, the Cascade LSTM LID+ASRs could consider the previous class output to decide the current class output to avoid the frame unit error.

Next, we compared the results between the Cascade and Direct approaches. The Cascade approach seems to be more influenced by language error than the Direct approach. It has to estimate language per speech frame, which has much longer sequences than per character. And the LID error of the Cascade approach can cause more damage to the ASR than the Direct approach since the wrong ASR cannot recognize at all, and even the right ASR cannot recognize well if the input speech mistakenly starts from the middle of words.

Compared among Direct approaches, with additional language information, Direct ASR (Proposed LID) was better than Direct ASR (No LID). Direct ASR (LngChr) needs more dictionary sizes, which gets more difficult to predict characters among more candidates correctly. The Direct ASR (Tag LID) has more difficulty with attention alignments since it predicts the language ID before the target sequences. Therefore, the Direct ASR (Proposed LID) tends to have the best performance. Table 4.10 shows that the case using CS data for training also has the same tendency.

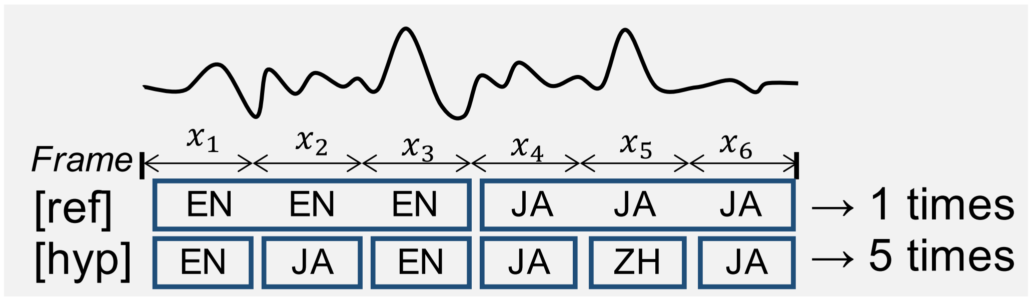


Figure 4.10: Example of frame unit error. The reference has three straight English language IDs and three straight Japanese language IDs, so the number of switching times is one time. On the other hand, the hypothesis inserted the wrong language ID "JA" between English language IDs or "ZH" between Japanese language IDs, so the number of switching times increased to 5 times.

To wrap up, the Direct ASR (Proposed LID) had the best performance among several LID approaches. The Direct approaches are better than the Cascade approaches since the Cascade approaches are more influenced by language error than the Direct approach. Comparison among the Direct approaches, the Direct ASR (Proposed LID) tends to perform the best performance. The Direct ASR (Proposed LID) has the advantage in terms of dictionary size, which is smaller than Direct ASR (LngChr) and Direct ASR (Tag LID) and more manageable with attention alignments than Direct ASR (Tag LID). Moreover, The Direct ASR (Proposed LID) can adjust the balance between LID training and character training using loss hyperparameters.

4.5 Summary

In this chapter, we investigated the best LID approaches for the language-aware ASR among Cascade MLP LID+ASRs, Cascade LSTM LID+ASRs, Direct ASR (No LID), Direct ASR (LngChr), Direct ASR (Tag LID), and Direct ASR (Proposed LID). The Cascade approaches were good at the LID prediction on the experiments with synthetic speech since their LID is an easier task with one-to-one classification. However, in the experiments with natural speech, the LID of Cascade approaches got more complicated. It seems to be challenging to predict

Table 4.9: Comparison ASR performances on a multi-pair CS between Cascade approaches and Direct approaches trained with monolingual data in CER%. Direct ASR (proposed LID) was trained with $\lambda_{Lng} = 0.1$ and $\lambda_{Chr} = 1 - \lambda_{Lng}$. The p-values compared to the Cascade MLP LID+ASRs for statistical significance are presented using ***, **, * and no-star ($^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$).

Train:	Monolingual			Code-switching		
Ja25k+En25k+Zh25k	**Ja**	**En**	**Zh**	**EnJaCS**	**JaZhCS**	**ZhEnCS**
Cascade MLP LID+ASRs	20.7	31.2	48.9	25.6	23.2	34.9
Cascade LSTM LID+ASRs	16.9***	19.2***	46.5**	23.3***	17.7***	30.4***
Direct ASR (No LID)	8.8***	9.1***	5.8***	11.5***	12.3***	13.3***
Direct ASR (LngChr)	8.3***	9.1***	5.4***	14.9***	22.9	19.2***
Direct ASR (Tag LID)	8.5***	7.3***	4.9***	13.1***	14.9***	14.6***
Direct ASR (Proposed LID)	7.3***	7.3***	5.1***	10.1***	11.2***	11.4***

the language ID per speech frame on the natural CS data. It may be because the sound between words is affected by each other even beyond the language switching points. Although the language prediction of Cascade MLP LID+ASRs has similar performance with the Cascade LSTM LID+ASRs, the mistakes of Cascade MLP LID+ASRs occur at the frame unit, so that it increases the number of switching times and makes the subsequent ASR difficult to predict. On the other hand, the Cascade LSTM LID+ASRs can consider the previous class output to decide the current class output to avoid the frame unit error.

The Cascade approaches were sometimes good at the LID prediction, but their character error rate was not good. They seem to be more influenced by language error than the Direct approach. It has to estimate language per speech frame, which has much longer sequences than per character. And the LID error of the Cascade approach can cause more damage to the ASR than the Direct approach since the wrong ASR cannot recognize at all, and even the right ASR cannot recognize well if the input speech mistakenly starts from the middle of words.

Among Direct approaches, with additional language information, Direct ASR (Proposed LID) was better than Direct ASR (No LID). Compared to other Direct approaches using LID, Direct ASR (Proposed LID) was better. Direct ASR (LngChr) needs more dictionary sizes, which gets more difficult to predict characters among more candidates correctly. The Direct ASR (Tag LID) has more

Table 4.10: Comparison ASR performances on a multi-pair CS among Direct approaches trained with CS data in CER%. Direct ASR (proposed LID) was trained with $\lambda_{Lng} = 0.1$ and $\lambda_{Chr} = 1 - \lambda_{Lng}$. The p-values compared to the Cascade MLP LID+ASRs for statistical significance are presented using ***, **, * and no-star ($^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$).

Train: EnJaCS10k+JaZhCS10k +NatEnJaCS0.7k	Monolingual			Code-switching			
	Ja	**En**	**Zh**	**EnJaCS**	**JaZhCS**	**EnZhCS**	**TstNat EnJaCS**
Cascade MLP LID+ASRs	24.5	32.3	43.0	20.8	26.3	29.6	69.8
Cascade LSTM LID+ASRs	21.3^{**}	30.7^{*}	41.5^{**}	19.0^{*}	24.2^{*}	27.2^{**}	65.7^{***}
Direct ASR (No LID)	9.5^{***}	30.6^{*}	16.2^{***}	13.1^{***}	7.6^{***}	13.2^{***}	20.6^{***}
Direct ASR (LngChr)	20.7^{**}	36.4	18.1^{***}	20.3	8.9^{***}	17.2^{***}	29.4^{***}
Direct ASR (Tag LID)	22.2	49.5	44.8	22.6	12.4^{***}	24.2^{***}	30.8^{***}
Direct ASR (Proposed LID)	9.5^{***}	29.9^{*}	16.8^{***}	12.9^{***}	7.1^{***}	11.9^{***}	18.9^{***}

difficulty with attention alignments since it predicts the language ID before the target sequences. Therefore, we confirmed that the Direct ASR (Proposed LID) tended to have the best performance. The Direct ASR (Proposed LID) has the advantage in dictionary size and is more manageable with attention alignments. Therefore, the Direct ASR (Proposed LID) tended to show the best performance.

Chapter 5

Proposed Machine Speech Chain for Semi-supervised Code-switching ASR and TTS

This chapter explains the semi-supervised approach for CS ASR and CS TTS with the machine speech chain.

5.1 Introduction

Common methods of developing CS ASR and TTS rely on supervised learning that requires large amounts of CS data for training models. However, pairs of CS speech and corresponding CS transcriptions are scarce and difficult to obtain, although either CS text or CS speech may be found on social media. Such a data problem hinders the development of CS ASR.

On the other hand, recently, a framework called a machine speech chain [10,11] was proposed to achieve semi-supervised learning for ASR and TTS, trainable with labeled and unlabeled data. The machine speech chain mechanism has a feedback loop between ASR and TTS, allowing them to support each other given the available unpaired speech or text data (unlabeled data). However, the existing works on machine speech chains [10, 11] have only addressed the monolingual issue.

Therefore, in this study, we propose utilizing the machine speech chain for CS tasks to handle not only monolingual but also bilingual. First, we train ASR and TTS with the labeled monolingual data in supervised learning. Next, we perform a machine speech chain with the semi-supervised learning with only CS text or CS speech without requiring any labeled CS data. We also extend the machine speech chain to handle CS better by integrating language embedding and language identification (LID) and investigate our proposed model's performance both on a single-pair CS and multi-pair CS. The multi-pair CS includes the unknown CS excluded from the training data. The task of predicting the unknown CS without training is called a zero-shot CS. It is difficult to predict the switching points and the language in that situation since the target CS is not used as training data. We expect that language embedding and LID can solve these problems by delivering language information in training.

Finally, we analyze it both on native CS and non-native CS. For non-native CS, we use the natural Mandarin-English CS data, SEAME corpus [22]. We control the accented problem better by utilizing efficient pronunciation-assisted sub-word modeling (PASM) [58].

5.2 Related Works

Most previous researches suffer from one or more of the following disadvantages: (a) developed on either only ASR or only TTS; (b) focused only on a single-pair CS; (c) trained in supervised learning that requires a large amount of labeled CS data in which the CS speech and corresponding CS transcriptions are hard to obtain. In contrast, our study builds end-to-end encoder-decoder models for both CS ASR and TTS and connects them so that they train each other. The machine speech chain framework can train CS ASR and CS TTS together in semi-supervised learning, even without labeled CS data.

The common semi-supervised approaches, such as label propagation [64], decode the unlabeled speech with supervised seed ASR and retrains the model using the output text as pseudo-label. Semi-supervised CS ASR using pseudo-label is proposed [65]. However, this method can be unstable performance because of the unreliable labels. On the other hand, the machine speech chain utilizes TTS after getting output text and compares the original speech with the output speech to train with reliable information. Therefore, we adopt the machine speech chain approach.

We also handle multi-pair CS, not only a single-pair CS. We integrate language embedding and LID into the machine speech chain and explore how well the model performs on both a single-pair CS and multi-pair CS, including the unknown CS excluded from the training data. We call the task predicting the unknown CS excluded from the training data as zero-shot CS.

Zero-shot learning, which was initially proposed in the field of computer vision, refers to the problem of recognizing objects that may not have appeared in the training data in multiclass classification [66]. In machine translation, zero-shot tasks faced the challenge of translating the language combinations that were excluded in training sets [67]. Unfortunately, few studies have addressed CS ASR and TTS, so this study has also contributed to the zero-shot CS ASR and TTS.

5.3 Proposed Approaches

5.3.1 Human Speech Chain

The human speech chain [68] is an essential mechanism for communication. We communicate by expressing our thoughts and listening to others. This speaking and listening cycle also occurs when we talk to ourselves. When we utter a word, we aurally check whether we spoke it as we intended. We simultaneously improve speaking and listening while alternately repeating sounds and words. The human speech chain is defined by such a communication cycle.

5.3.2 Machine Speech Chain

Tjandra et al. developed a deep learning-based monolingual machine speech chain [10, 11, 69], inspired by the human speech chain as Fig. 5.1 shows. Its framework is illustrated in Fig. 5.2. It is composed of an end-to-end ASR [16, 17] and an end-to-end TTS [18], and they are connected. The architecture can train ASR and TTS with each other with their feedback. The monolingual machine speech chain [10] improves the performance of monolingual ASR and TTS. The multi-speaker machine speech chain [10, 11] is expanded to deal with multi-speakers

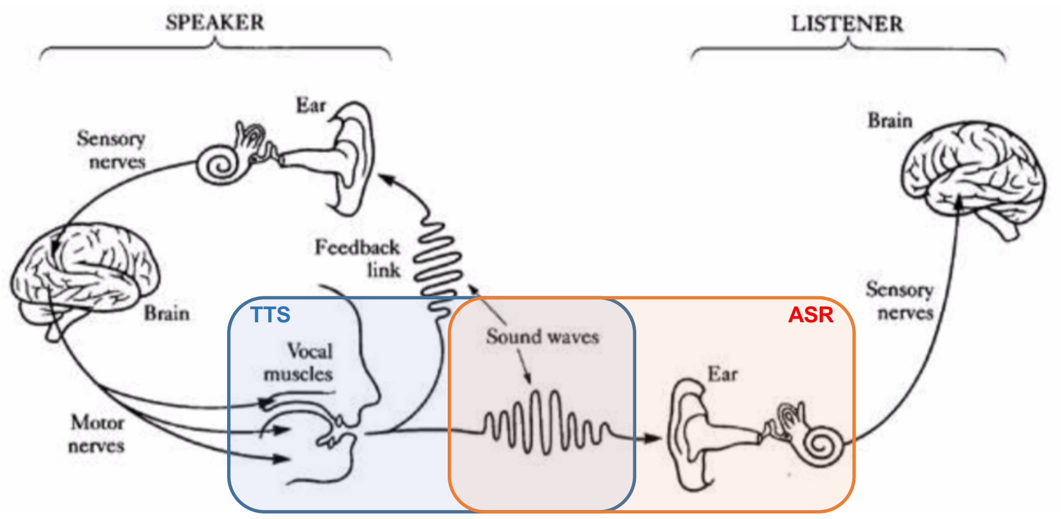


Figure 5.1: Overview of human speech chain [68]. The human speech chain refers to a communication mechanism where a spoken message transmits from the speaker's brain to the listener's brain. A speaker generates a speech sound wave that travels through the air to the listener, and then the listener recognizes the speaker's message. The speakers can also be their own listeners with auditory feedback from their mouths to their ears. The TTS corresponds to the speaker (blue framed box), and the ASR corresponds to the listener (red framed box). The mechanism realized the speech chain using ASR and TTS is the machine speech chain [10].

by integrating speaker recognition (SPKREC) based on DeepSpeaker [19]. Still, they are only for monolingual; they cannot handle CS. Therefore, in this study, we expand it to handle CS.

5.3.3 Basic Code-switching Machine Speech Chain

The basic CS machine speech chain (Fig. 5.3) seeks to improve the ASR and TTS performance on CS without any labeled CS data. The learning process is as follows (In the case of handling multiple speakers, the speaker vector $z = \text{SPKREC}(\mathbf{x})$ is added to the input of the TTS decoder both during supervised and semi-supervised processes):

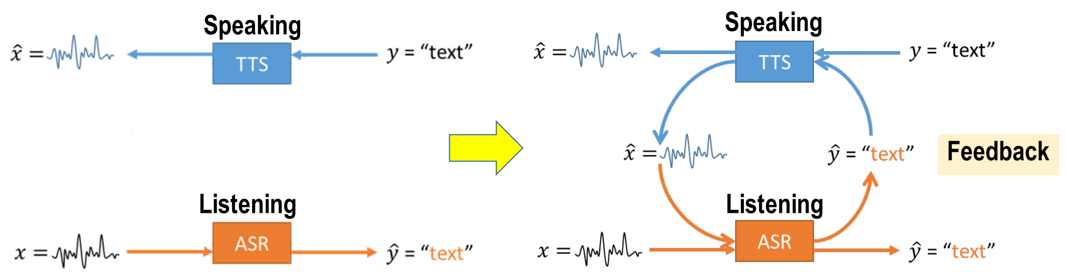


Figure 5.2: The overview of machine speech chain framework [10]. The left figure shows the TTS relevant to speaking and the ASR relevant to listening. The right figure shows the closed-loop architecture of the ASR and TTS with feedback interaction, that is, the machine speech chain framework.

1. **Supervised learning of ASR and TTS with paired speech+text monolingual data**
   First, ASR and TTS are trained in supervised learning with the paired speech+text Japanese and English data or the paired speech+text Mandarin and English data (mixed data of monolingual sets constituting a CS language pair) as shown in Fig. 5.3(a). Once ASR receives the speech and the corresponding text $(\boldsymbol{x}^{Mono}, \boldsymbol{y}^{Mono})$, ASR recognizes speech $\hat{\boldsymbol{y}}^{Mono}$ using teacher-forcing, which is an algorithm that trains efficiently and converges faster by direct training with the target label. Then the loss between output text $\hat{\boldsymbol{y}}^{Mono}$ and reference text $\boldsymbol{y}^{Mono}$ is calculated $L^{Mono}_{ASR}(\hat{\boldsymbol{y}}^{Mono}, \boldsymbol{y}^{Mono})$ using Eq. (2.12). TTS also generates speech $\hat{\boldsymbol{x}}^{Mono}$ from the input text $\boldsymbol{y}^{Mono}$, and the loss between generated speech $\hat{\boldsymbol{x}}^{Mono}$ and reference speech $\boldsymbol{x}^{Mono}$ is calculated $L^{Mono}_{TTS}(\hat{\boldsymbol{x}}^{Mono}, \boldsymbol{x}^{Mono})$, where the loss function in case of single speaker is Eq. (2.13) and the loss function in case of multi-speaker is Eq. (2.17). The parameters are tuned to decrease the loss with gradient descent optimization.

2. **Semi-supervised learning of ASR and TTS together in a machine speech chain**
   We performed a machine speech chain, where we trained ASR and TTS together with an unpaired CS text or an unpaired CS speech data (Fig. 5.3(b)).

The learning process during the semi-supervised learning of the machine speech chain consists of the following two processes:

(a) **Loop connection from TTS to ASR with only unpaired CS text data**
This process (Fig. 5.3(c)) only uses unpaired CS text data $\boldsymbol{y}^{CS}$. TTS outputs speech $\hat{\boldsymbol{x}}^{CS}$ from the input CS text $\boldsymbol{y}^{CS}$, and ASR also predicts text transcription $\hat{\boldsymbol{y}}^{CS}$ from the synthesized speech. Then loss $L^{CS}_{ASR}(\hat{\boldsymbol{y}}^{CS}, \boldsymbol{y}^{CS})$ can be computed between output text $\hat{\boldsymbol{y}}^{CS}$ and input text $\boldsymbol{y}^{CS}$ to tune the ASR parameters.

(b) **Loop connection from ASR to TTS with only unpaired CS speech data**
This process (Fig. 5.3(d)) only uses unpaired CS speech data $\boldsymbol{x}^{CS}$. Once ASR receives speech x^{CS}, ASR outputs predicted transcription $\hat{\boldsymbol{y}}^{CS}$, and TTS generates speech $\hat{\boldsymbol{x}}^{CS}$ from the text of the ASR output. The loss between output speech $\hat{\boldsymbol{x}}^{CS}$ and original speech $\boldsymbol{x}^{CS}$ can be computed $L^{CS}_{TTS}(\hat{\boldsymbol{x}}^{CS}, \boldsymbol{x}^{CS})$ for tuning the TTS parameters.

During the semi-supervised learning process, we also continue supervised learning with monolingual data. The supervised learning loss and unsupervised learning loss are integrated into a single loss:

$$L_{Chain} = \alpha(L^{Mono}_{ASR} + L^{Mono}_{TTS}) + \beta(L^{CS}_{ASR} + L^{CS}_{TTS}), \tag{5.1}$$

$$\theta_{ASR} = Optim(\theta_{ASR}, \nabla_{\theta_{ASR}} L_{Chain}), \tag{5.2}$$

$$\theta_{TTS} = Optim(\theta_{TTS}, \nabla_{\theta_{TTS}} L_{Chain}), \tag{5.3}$$

where the hyperparameters α and β tune the balance of the losses. They balance the influence between the supervised and unsupervised, and between the monolingual and CS data.

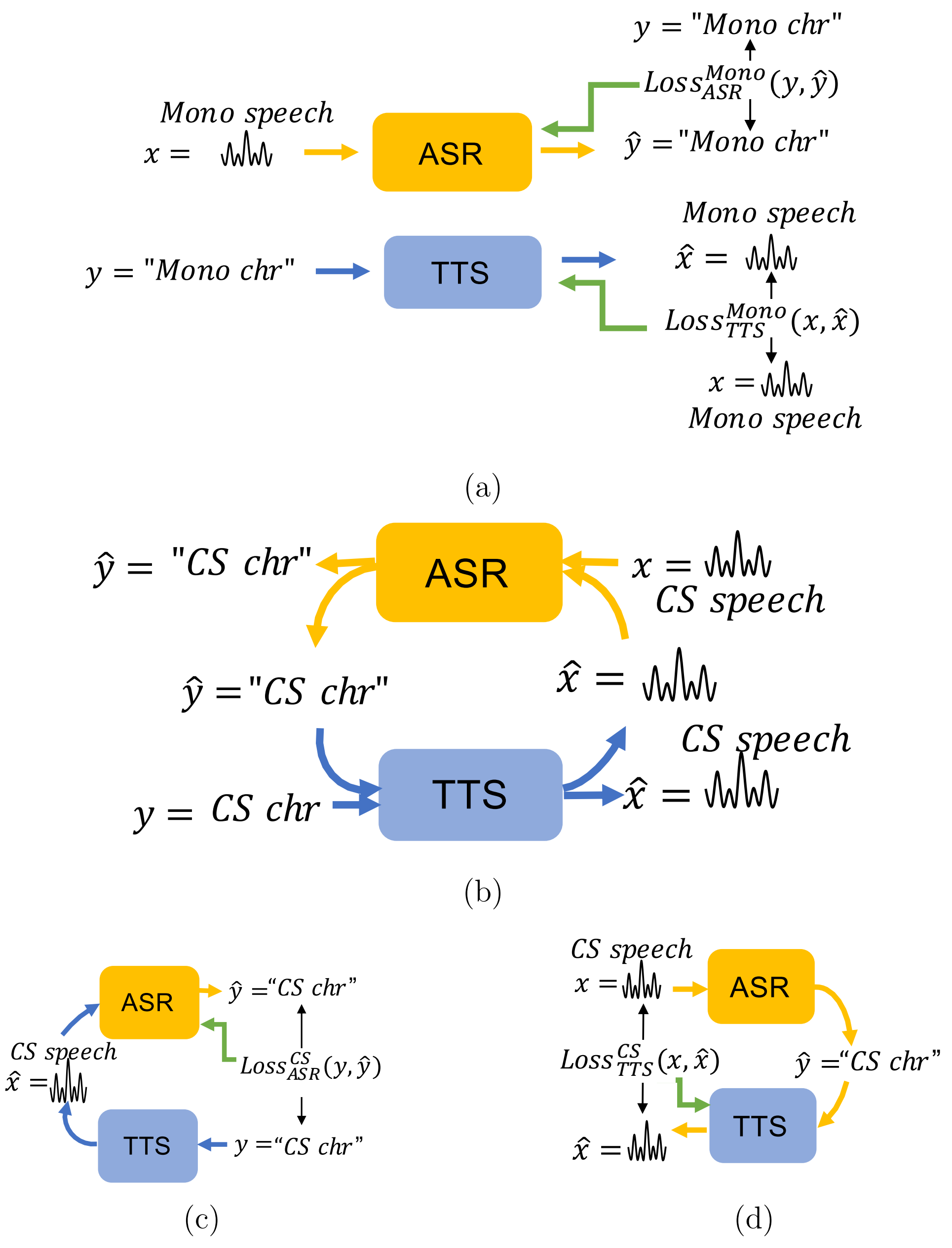


Figure 5.3: Overview of the proposed framework based on [10, 70]: (a) Supervised learning of ASR and TTS with paired speech+text monolingual data of two languages; (b) Semi-supervised learning of ASR and TTS together through machine speech chain with unpaired CS text data or unpaired CS speech data; (c) Loop connection from TTS to ASR with only unpaired CS text data; (d) Loop connection from ASR to TTS with only unpaired CS speech data.

5.3.4 Language-aware Code-switching Machine Speech Chain

Fig. 5.4 shows the differences among the following: (a) a basic CS machine speech chain [70], (b) a multi-speaker CS machine speech chain that incorporates SPKREC for handling multiple speakers, and (c) a language-aware CS machine speech chain [71].

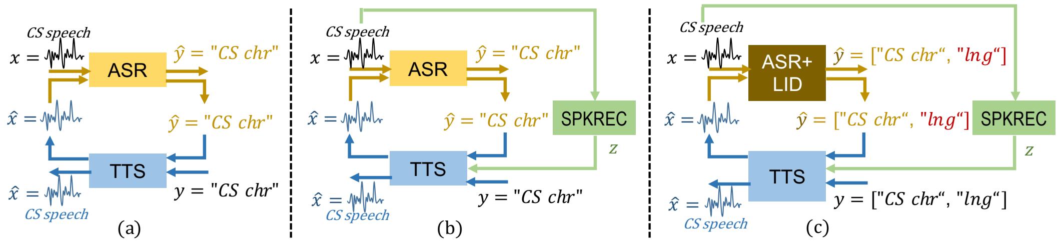


Figure 5.4: Comparison among CS machine speech chain models: (a) Basic CS machine speech chain [70]; (b) Multi-speaker CS machine speech chain incorporating SPKREC; (c) Language-aware CS machine speech chain [71].

In a language-aware CS machine speech chain, we handle CS more efficiently with language information. To achieve this, we put additional functions, LID for ASR and language embedding for TTS. As Fig. 5.5 shows, the LID architecture performs multi-task learning in the ASR softmax layers. The architecture trains the projection between the speech input and the two outputs of the text transcription and the language information with two softmax layers (Fig. 5.5). The language information is given to each character by the language ID. For language IDs, Japanese is denoted as "JA," English is denoted as "EN," Chinese is denoted as "ZH," and an unknown language is denoted as"<unk>."

The language embedding of TTS maps a one-hot vector representing a language ID into continuous vectors. Then it concatenates with the character embedding and goes through the encoder LSTM, attention, decoder, and generates speech. In the case of handling multiple speakers, the speaker vector $z = \text{SPKREC}(\mathbf{x})$ is added to the input of the TTS decoder both during supervised and semi-supervised training.

The training process is almost the same as the basic CS machine speech chain, but the language-aware CS machine speech chain trains language information.

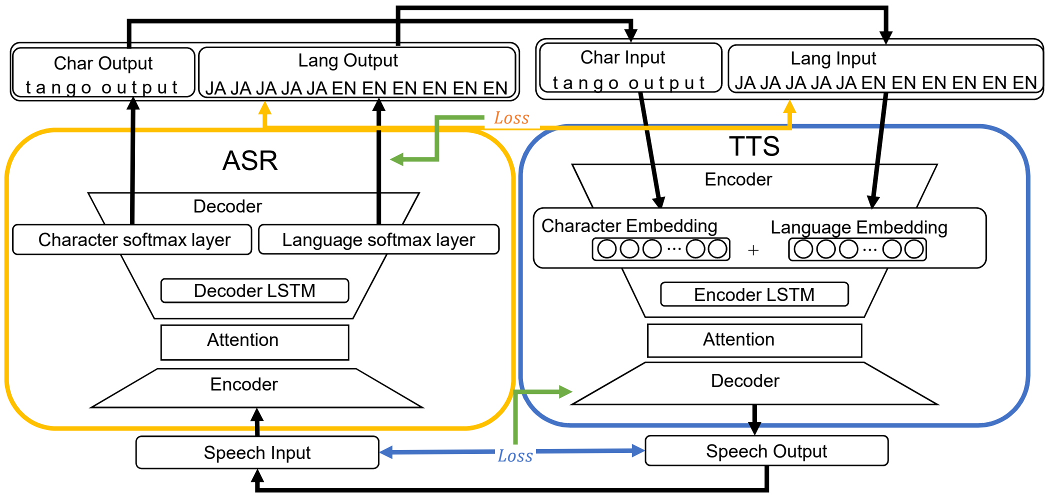


Figure 5.5: Language-aware code-switching machine speech chain. Given input speech, the ASR outputs the character sequences and language ID sequences using two softmax layers. Then the TTS embeds the character sequences language ID sequences and concatenates them to go through processes.

The following is the training process:

1. **Supervised learning of ASR and TTS with paired speech+text monolingual data**
   As shown in Fig. 5.6(a), we first train the ASR and TTS systems with the paired speech+text monolingual corpora from several languages using English (En), Japanese (Ja), and Chinese (Zh). With paired speech+text monolingual data ($\boldsymbol{x}^{Mono}$, $\boldsymbol{y}^{MonoChr}$, and $\boldsymbol{y}^{MonoLng}$), ASR generates sequences of characters $\hat{\boldsymbol{y}}^{MonoChr}$ and language information $\hat{\boldsymbol{y}}^{MonoLng}$ with teacher-forcing and calculates the sum of losses $L_{ASR}^{MonoChr}(\hat{\boldsymbol{y}}^{MonoChr}, \boldsymbol{y}^{MonoChr})$ and $L_{ASR}^{MonoLng}(\hat{\boldsymbol{y}}^{MonoLng}, \boldsymbol{y}^{MonoLng})$. The loss function for optimizing ASR changes from Eq. (2.12) to the following function in accordance with the incorporating LID loss:

$$L_{ASR}^{LngAwr} = \lambda_{Chr} L_{ASR}^{Chr} + \lambda_{Lng} L_{ASR}^{Lng}, \tag{5.4}$$

   where it's a summation of two negative log-likelihood, tuning those weights by the hyperparameters λ_{Chr} and λ_{Lng}. TTS generates speech features

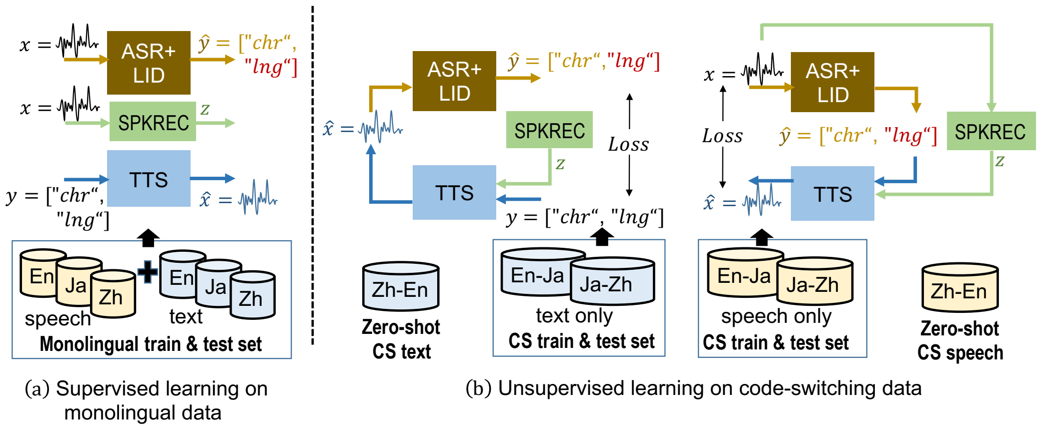


Figure 5.6: Training overview of language-aware CS machine speech chain [71]: (a) supervised training of ASR or TTS with paired speech+text monolingual data; (b) semi-supervised training of a machine speech chain with unpaired CS text or CS speech data.

$\hat{\boldsymbol{x}}^{Mono}$ with teacher-forcing, and we calculate the loss $L_{TTS}^{Mono}(\hat{\boldsymbol{x}}^{Mono}, \boldsymbol{x}^{Mono})$. The TTS loss function does not change from Eq. (2.17) since the TTS output does not change. The parameters are tuned to reduce the loss with gradient descent optimization.

2. **Semi-supervised training of ASR and TTS together in a machine speech chain**

   (a) **Loop connection from TTS to ASR with only unpaired CS text data of characters and language information**
   This process (Fig. 5.6(b), left side) uses only unpaired CS text data of characters and language information [y^{CSChr}, and y^{CSLng}]. TTS outputs speech $\hat{\boldsymbol{x}}^{CS}$ from the unpaired CS text data of the characters and language information [y^{CSChr}, y^{CSLng}]. The generated speech is transcribed by ASR to the CS text [$\hat{\boldsymbol{y}}^{CSChr}$,$\hat{\boldsymbol{y}}^{CSLng}$]. Then the sum of losses $L_{ASR}^{CSChr}(\hat{\boldsymbol{y}}^{CSChr}, \boldsymbol{y}^{CSChr})$ and $L_{ASR}^{CSLng}(\hat{\boldsymbol{y}}^{CSLng}, \boldsymbol{y}^{CSLng})$ can be computed to update the ASR parameters.

   (b) **Loop connection from ASR to TTS with only unpaired CS speech data**

This process (Fig. 5.6(b), right side) only uses CS speech $\boldsymbol{x}^{CS}$ as input. With unlabeled CS speech x^{CS}, ASR generates sequences of characters $\hat{\boldsymbol{y}}^{CSChr}$ and language information $\hat{\boldsymbol{y}}^{CSLng}$. TTS generates CS speech $\hat{\boldsymbol{x}}^{CS}$ with output CS characters and language information from ASR. Then TTS parameters are tuned to decrease loss $L^{CS}_{TTS}(\hat{\boldsymbol{x}}^{CS}, \boldsymbol{x}^{CS})$.

In the end, the losses of the supervised monolingual and unsupervised CS losses are combined into a single loss:

$$\begin{aligned} L^{LngAwr}_{Chain} = \; & \alpha((\lambda_{Chr} L^{MonoChr}_{ASR} + \lambda_{Lng} L^{MonoLng}_{ASR}) \\ & + L^{Mono}_{TTS}) \\ & + \beta((\lambda_{Chr} L^{CSChr}_{ASR} + \lambda_{Lng} L^{CSLng}_{ASR}) \\ & + L^{CS}_{TTS}), \end{aligned} \tag{5.5}$$

$$\theta_{ASR} = Optim(\theta_{ASR}, \nabla_{\theta_{ASR}} L^{LngAwr}_{Chain}), \tag{5.6}$$

$$\theta_{TTS} = Optim(\theta_{TTS}, \nabla_{\theta_{TTS}} L^{LngAwr}_{Chain}), \tag{5.7}$$

where the hyperparameters α and β tune the balance of the losses. They balance the influence between the supervised and unsupervised, and between the monolingual and CS data. After training, we can perform the ASR and TTS on zero-shot CS.

5.4 Experiments

5.4.1 Experimental Settings

We conducted the experiments on the following three scenarios: single-pair CS (synthetic speech; single speaker), single-pair CS (natural speech; multi-speaker), and multi-pair CS.

Table 5.1: Statistics of datasets for a single-pair CS (synthetic speech; single speaker).

Subset			Hours	Utterances
Train	Mono	Ja25k+En25k (JaTTS)	50.7	50000
		Ja25k+En25k (MixTTS)	39.6	50000
	CS	JaEnCS10k (JaTTS)	9.5	10000
		JaEnCS10k (MixTTS)	8.9	10000
		JaEnCS20k (JaTTS)	19.0	20000
		JaEnCS20k (MixTTS)	17.8	20000
		JaEnCS20k (Ja+MixTTS)	18.4	20000
Test	Mono	TstJa(JaTTS)	0.7	500
		TstEn(EnTTS)	0.6	500
	CS	TstJaEnCS(JaTTS)	1.1	500
		TstJaEnCS(MixTTS)	0.7	500

Dataset Composition

For the experiment on single-pair CS (synthetic speech; single speaker), we used the BTEC corpus (Sect. 3.1.1) and intra-sentential word-level CS and phrase-level CS I of synthetic speech CS (Sect. 3.2.2). We randomly chose 50-k Japanese and English sentences for training sets and 500 for test sets from BTEC1-4. From the synthetic speech CS, we chose 10-k or 20-k sentences for training sets and 500 sentences for test sets. They are denoted as JaTTS if synthesized using Japanese TTS, as EnTTS synthesized using English TTS, and as MixTTS if synthesized both using Japanese TTS and English TTS. Table 5.1 shows the statistics of these datasets for a single-pair CS (synthetic speech; single speaker).

For the experiments on single-pair CS (natural speech; multi-speaker), we used LibriSpeech, AISHELL-1, and SEAME corpora (Sect. 3.1.1-3.1.2). Table 5.2 shows the statistics of these datasets for a single-pair CS (natural speech; multi-speaker).

For the experiments on multi-pair CS, we used the BTEC corpus (Sect. 3.1.1) and intra-sentential phrase-level CS II of synthetic speech CS (Sect. 3.2.2) and the natural speech CS (Sect. 3.2.3). From the BTEC corpus, we randomly selected

Table 5.2: Statistics of LibriSpeech, AISHELL-1, and SEAME for a single-pair CS (natural speech; multi-speaker).

Subset			Speakers	Hours	Utterances
Train	Mono	LibriSpeech	251	100.6	28539
		AISHELL-1	340	100.0	80066
		SEAME	134	31.5	42911
		Total	725	232.1	151516
	CS	SEAME	134	69.6	51027
Test	dev_{man}	*mono*	10	1.6	2228
		CS	10	5.9	4303
		all	10	7.5	6531
	dev_{sge}	*mono*	10	1.8	3156
		CS	10	2.1	2165
		all	10	3.9	5321

25-k Japanese utterances, 25-k English utterances, and 25-k Chinese utterances: "Ja25k+En25k+Zh25k," and 500 sentences for each test set. From the synthetic speech CS, we used 10-k sentences for each training set. We also selected 500 sentences for each test set. They are synthesized using the corresponding language's TTS, where Japanese words are synthesized with Japanese TTS, English words are synthesized with English TTS, and Chinese words are synthesized with Chinese TTS. From the natural speech CS, we divided into 0.2k labeled data, 0.7k unlabeled data denoted as "NatEnJaCS," and 0.1k test data denoted as "TstNatEnJaCS." The labeled data were also divided between Japanese and English, which can be used for monolingual data. Those data will be later called "NatJa" and "NatEn." Table 5.3 shows the statistics of these datasets for multi-pair CS.

All the text characters were converted to lowercase letters and punctuation marks [, : ? .] were removed. We converted all BTEC characters into the lowercase alphabet. For Japanese words, we applied a morphological analyzer Mecab [55] to convert into katakana. Then we converted the katakana into English letters with pykakasi [56]. We also used pypinyin to the Chinese characters [57] and converted them into pinyin. The text not including any language identifier consists of 26 letters (a-z), one mark (-) for stretching Japanese sounds,

Table 5.3: Statistics of datasets for multi-pair CS.

Subset			Hours	Utterances
Train	Mono	Ja25k+En25k+Zh25k	78.8	75000
		NatJa0.2k+NatEn0.2k	0.3	400
	CS	EnJaCS10k	11.8	10000
		JaZhCS10k	10.4	10000
		ZhEnCS10k	10.9	10000
		EnFrCS10k	9.6	10000
		NatEnJaCS0.7k	1.1	700
Test	Mono	Ja	0.9	500
		En	0.7	500
		Zh	0.8	500
	CS	EnJaCS	0.8	500
		JaZhCS	0.7	500
		ZhEnCS	0.7	500
		TstNatEnJaCS	0.2	100

and three tags that denote the start of sentences (<s>), the end of sentences (</s>), and the spaces between words (<spc>). In the case of LibriSpeech, AISHELL-1, and SEAME, we utilized PASM sub-word units, which is a sub-word unit optimized for accents by taking alignments between phonemes and characters [58].

Model Details

Our ASR system is an attention-based encoder-decoder model [16] described in Sect. 2.2.2. We used the model, which consists of three stacked BiLSTM layers on the encoder, a single LSTM layer on the decoder, and multilayer perceptron (MLP)-based attention [15] components. The decoder has two softmax layers for predicting the language ID and character sequences in the language-aware CS machine speech chain. The activation function is a LeakyReLU ($l = 1e - 2$) [59]. We did not use the language model.

For the TTS system, our model is based on a sequence-to-sequence TTS (Tacotron) [18] described in Sect. 2.2.3. Although its hyperparameters are almost the same as the original Tacotron, we used LeakyReLU instead of ReLU. Also, on the encoder, although the original Tacotron uses 16 sets of convolutional filters in the CBHG module, we used eight sets of different filter banks to reduce the GPU memory consumption. In the language-aware CS machine speech chain, the encoder has a language-embedding layer as well as a character-embedding layer. The decoder changed the GRU into two stacked LSTMs with 256 hidden units. For the multi-speaker scenario on single-pair CS (natural speech; multi-speaker) and multi-pair CS, we used a DNN-based speaker recognition model called Deep Speaker [19]. We generated a speaker vector from the trained Deep Speaker model and incorporated a speaker-embedding layer into the Tacotron decoder. The hyperparameters of weighting the losses in multi-speaker Tacotron (Eq. (2.17)) are set $\gamma_1 = 1, \gamma_2 = 1, \gamma_3 = 0.25$.

For the α and β hyperparameters of the machine speech chain loss in Eq. (5.1) and Eq. (5.5), we used the same $\alpha = 0.5$, $\beta = 1$ for most of our experiments. We implemented both the ASR and TTS models with the PyTorch library [60].

5.4.2 Results

Results on Single-pair Code-switching

(Synthetic Speech; Single Speaker)

The experimental results on single-pair CS (synthetic speech; single speaker) are described separately in the baseline and proposed systems.

Baseline Systems

We had four types of test sets for our evaluation: (1) **TstJa (JaTTS)**: a Monolingual Japanese test set generated using a Japanese TTS; (2) **TstJaEnCS (JaTTS)**: a Japanese-English intra-sentential CS test set, where both the Japanese part and the English part of the TstJaEnCS are generated using a Japanese TTS; (3) **TstJaEnCS (MixTTS)**: a Japanese-English intra-sentential CS test set generated using a mixed Japanese-English TTS, where we concatenated the speech generated by English TTS for the English part of CS and the speech generated by Japanese TTS for the Japanese part of CS; (4) **TstEn (EnTTS)**: a Monolingual English test set generated using an English TTS. Although we do not have an inter-sentential CS test set, the TstJa (JaTTS) and TstEn (EnTTS) combination are identical to the inter-sentential CS. We evaluated the generated transcription by the character error rate (CER). CER is the edit distance between the reference and the predicted transcription. We also assessed the statistical significance compared to the baseline systems by a matched-pair sentence-segment word error test [63]. For the TTS evaluation, we used the L2-norm squared of the log-Mel spectrogram between the reference and the predicted speech features.

The baseline system performances of ASR and TTS are individually shown in Fig. 5.7 and 5.8. The baseline systems were trained in supervised learning using an attention-based encoder-decoder model framework without a machine speech chain framework. Four types of baselines were evaluated: (1) **Ja50k (JaTTS)**: ASR or TTS trained with 50-k Japanese speech generated using a Japanese TTS; (2) **Ja25k+En25k (JaTTS)**: ASR or TTS trained with 25-k Japanese speech plus 25-k English speech generated using a Japanese TTS (inter-sentential CS); (3) **Ja25k+En25k (MixTTS)**: ASR or TTS trained with 25-k Japanese speech generated using a Japanese TTS and 25-k English speech generated using an English TTS (inter-sentential CS); (4) **En50k (EnTTS)**: ASR or TTS trained with 50-k English speech generated using an English TTS.
As Fig. 5.7 shows, the CER of the Ja50k (JaTTS) ASR was low in the Japanese test, but very high in the English test. In the same way, the CER of the En50k (EnTTS) ASR was very low in the English test but increased in the Japanese test. The Ja25k+En25k (JaTTS) learned English sentences and Japanese sentences, but the English test performance remained unsatisfactory when the speech was generated using Japanese TTS. A similar tendency was slightly shown in the TTS results. Ja25k+En25k (MixTTS), which was trained using speech generated by a Japanese TTS and an English TTS, controlled the balance well among the Japanese, English, and Japanese-English CS test sets. Therefore, we use this Ja25k+En25k (MixTTS) as our baseline model.

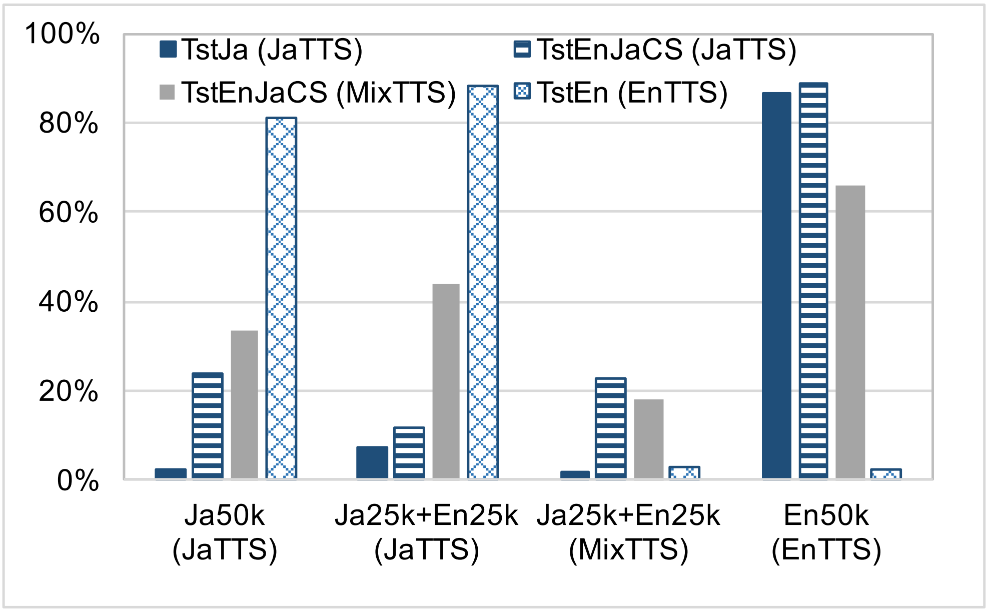


Figure 5.7: ASR baseline performances of single-pair CS (synthetic speech; single speaker) in CER.

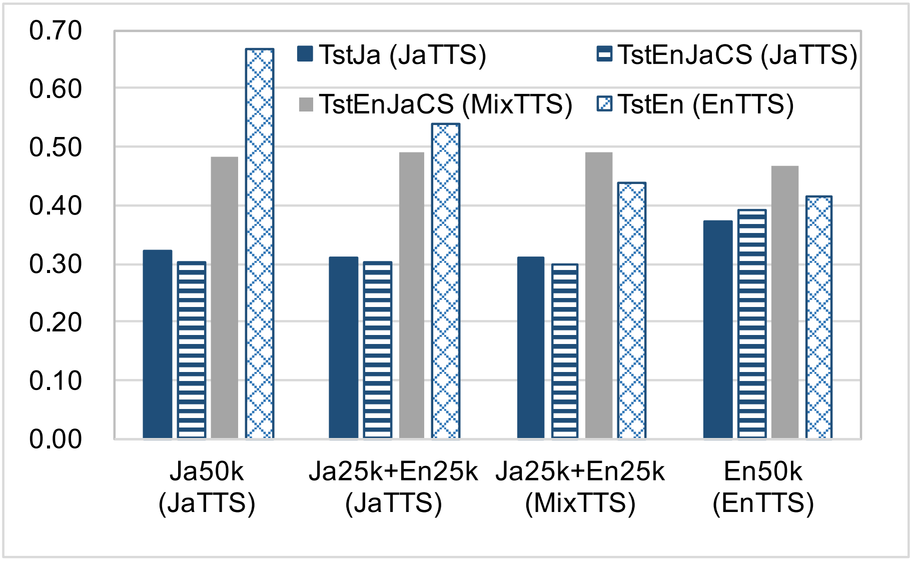


Figure 5.8: TTS baseline performances of single-pair CS (synthetic speech; single speaker) in L2-norm squared of the log-Mel spectrogram.

Table 5.4: ASR performances of a single-pair CS (synthetic speech; single speaker) machine speech chain in CER% . The left side of arrows in the LID is λ_{Lng} value during the supervised learning process, and the right side is λ_{Lng} value during the semi-supervised learning process, where $\lambda_{Chr} = 1 - \lambda_{Lng}$. The p-values compared to the baseline system for statistical significance are presented using ***, **, * and no-star ($^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$).

		Monolingual		Code-switching
Model	LID type	**TstJa (JaTTS)**	**TstEn (EnTTS)**	**TstJaEnCS (MixTTS)**
[Baseline] Supervised learning: labeled mono				
Ja25k+En25k (MixTTS)	No LID	1.7	3.0	18.1
[Proposed Machine Speech Chain] Semi-supervised learning: labeled mono + unlabeled CS				
+JaEnCS10k (JaTTS)	No LID	1.9	4.8	19.7
+JaEnCS20k (JaTTS)	No LID	1.9	4.7	17.2
+JaEnCS10k (MixTTS)	No LID	1.8	3.7	5.4***
+JaEnCS20k (MixTTS)	No LID	1.9	3.6	5.5***
+JaEnCS20k (Ja+MixTTS)	No LID	1.8	4.1	5.1***
+JaEnCS20k (Ja+MixTTS)	LID (0.25→0.0)	1.7	3.2	3.7***
+JaEnCS20k (Ja+MixTTS)	LID (0.25→0.1)	1.7	3.3	3.4***
[Topline] Supervised learning: labeled mono + labeled CS				
+JaEnCS20k (Ja+MixTTS)	No LID	5.1	9.8	3.5***

Proposed Systems

Our proposed models aim to improve ASR and TTS to handle CS input well even without labeled CS for training while keeping the performance of the monolingual test. Table 5.4 shows the ASR and TTS performances of the proposed CS machine speech chain framework. After we individually trained ASR and TTS using labeled monolingual Ja25k and En25k, Ja25k+En25k (MixTTS), we carried out a machine speech chain on the following settings: (1) **JaEnCS (JaTTS)**: semi-supervised learning with unlabeled code-switching JaEnCS (JaTTS); (2) **JaEnCS (MixTTS)**: semi-supervised learning with unlabeled code-switching JaEnCS (Mix TTS); (3) **JaEnCS (Ja+MixTTS)**: a semi-supervised learning with unlabeled code-switching JaEnCS (JaTTS) and unlabeled code-switching JaEnCS

Table 5.5: Performance comparison between ASR systems trained in proposed machine speech chain with different λ_{Lng} (where $\lambda_{Chr} = 1 - \lambda_{Lng}$) in CER %. The left side of arrows is λ_{Lng} value during the supervised learning process with labeled Ja25k+En25k (MixTTS), and the right side is λ_{Lng} value during semi-supervised learning process with unlabeled JaEnCS20k (Ja+MixTTS).

	Monolingual		Code-switching
	TstJa	**TstEn**	**TstJaEnCS**
λ_{Lng}	**(JaTTS)**	**(EnTTS)**	**(MixTTS)**
No LID	1.8	4.1	5.1
0.25→0.0	1.7	**3.2**	3.7
0.25→0.1	**1.7**	3.3	**3.4**
0.25→0.25	1.9	3.5	3.9
0.25→0.5	1.8	3.8	4.1
0.25→0.75	2.1	3.9	4.5
0.5→0.0	1.9	3.5	3.5
0.5→0.1	1.9	3.7	4.2
0.5→0.25	1.7	3.5	3.6
0.5→0.5	2.0	3.8	4.3
0.5→0.75	1.8	4.2	5.7

(MixTTS). We excluded TstJaEnCS (JaTTS) from the test sets because many English words are pronounced as Japanese words generated by the Japanese TTS.

Our results show that our proposed model with JaEnCS20k (Ja+MixTTS) improved the ASR performance on the CS test, TstJaEnCS (MixTTS), from 18.1% CER to 5.1%, which reduced the absolute CER by 13.0%. It has a statistically significant difference. Monolingual performances are often damaged by optimizing the CS performance, but our proposed model maintained its performance on the monolingual test. It only slightly changed from 1.7% CER to 1.8% for the Japanese test and from 3.0% CER to 4.1% for the monolingual English test. It also improved the TTS performance on the CS test TstJaEnCS (MixTTS), where the L2-norm squared decreased from 0.489 to 0.372; the performance on the Japanese and monolingual English tests was maintained. Compared with the topline model that uses full-set data (speech+text), the proposed model reached a similar performance. Moreover, we investigated the performance of ASR trained by the language-aware CS machine speech chain with unlabeled JaEnCS20k (Ja+MixTTS). The performances among systems with some different hyperparameters λ_{Lng} are shown in Table 5.5. The best performance on TstJaEnCS (MixTTS) is 3.4% CER with λ_{Lng} (0.25→0.1), which improved even more than the Basic CS machine speech chain. The model with λ_{Lng} (0.25→0.0) performed the best performance on TstJa(JaTTS), so Table 5.4 shows both LID results of (0.25→0.0) and (0.25→0.1). Here, 0.0 indicates that the language information is used for character prediction of the semi-supervised learning process while maintaining the language information trained during the supervised learning process. Both cases of the LID showed a statistically significant difference with $p < .001$.

(Natural Speech; Multi-speaker)

The experimental results on single-pair CS (natural speech; multi-speaker) are described. We first evaluate our end-to-end ASR against previous research on the single-pair CS (natural speech; multi-speaker) data and then investigate the baseline and proposed system.

Evaluation of Our End-to-end ASR against Previous Researches

We compared our attention-based encoder-decoder models with Hybrid CTC/attention approaches [31] using supervised learning of the SEAME data. Following the counterpart's evaluation criterion, in this experiment, we evaluated a character-based model and a sub-word-based model with the token error rate (TER). The TER, which is calculated by the Word Error Rate (WER) for English and the Character Error Rate (CER) for Mandarin, is frequently adopted for evaluating the ASR of Mandarin-English CS because it is not affected by segmentation algorithms. For sub-words, we utilized PASM [58], whose effectiveness has already been shown for overcoming the byte-pair encoding (BPE) of sub-word units [72]. As shown in Table 5.6, our encoder-decoder-based model can be similar performances as

Table 5.6: ASR comparison between our attention-based encoder-decoder models and Hybrid CTC/attention approaches with SEAME data (in TER %). Our model with LID was trained with $\lambda_{Lng} = 0.1$ and $\lambda_{Chr} = 1 - \lambda_{Lng}$.

Model	dev_{man}	dev_{sge}
Hybrid CTC/attention char [31]	26.5	38.4
+LID [31]	25.6	37.0
Hybrid CTC/attention sub-word(BPE) [31]	26.4	36.1
+LID [31]	26.0	35.8
Att Enc-Dec char (ours)	26.2	37.8
Att Enc-Dec sub-word(PASM) (ours)	25.7	36.6
+LID	25.4	36.2

the CTC-based models.

Baseline and Proposed Systems

Next, we conducted machine speech chain experiments with the SEAME data. We first trained the base model with LibriSpeech, AISHELL-1, and the SEAME monolingual data. Then we performed a speech chain by the unlabeled SEAME CS data while continuing the supervised training of LibriSpeech and AISHELL-1 and the SEAME monolingual data. Table 5.7 shows the ASR results in TER. The baseline is the model trained with the labeled monolingual data of LibriSpeech and AISHELL-1, and SEAME. The proposed machine speech chain model improved the ASR performances on both the CS test sets of dev_{man} and dev_{sge} more than the baseline performances: from 47.7% to 37.4% and from 57.7% to 47.1%. Optimizing the CS performances only slightly degraded the performances on the monolingual evaluation sets. Still, our proposed model also improved the ASR on the overall performances from 44.9% to 37.6% on dev_{man} and from 53.6% to 49.5% on dev_{sge}. The topline is the model retrained with the labeled SEAME CS data from the model trained with the labeled monolingual data of LibriSpeech and AISHELL-1 and SEAME. Compared to the topline model, the proposed model achieves similar performance, although it did

Table 5.7: ASR performances of a single-pair CS (natural speech; multi-speaker) machine speech chain in TER %. The left side of arrows in the LID is λ_{Lng} value during the supervised learning process, and the right side is λ_{Lng} value during the semi-supervised learning process, where $\lambda_{Chr} = 1 - \lambda_{Lng}$. The p-values compared to the baseline system for statistical significance are presented using ***, **, *, and no-star ($^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$).

Model	dev_{man}			dev_{sge}		
	mono	*CS*	*all*	*mono*	*CS*	*all*
Supervised learning: labeled mono						
Baseline	33.3	47.7	44.9	47.8	57.7	53.6
Semi-supervised learning: labeled mono + unlabeled CS						
Label propagation	37.7	48.4	46.4	54.6	59.2	57.3
Proposed speech chain	38.6	37.4***	37.6***	52.9	47.1***	49.5***
+LID (0.25→0.0)	34.0	32.5***	32.8***	48.8	42.3***	45.0***
+LID (0.25→0.1)	35.1	33.7***	33.9***	50.0	42.8***	45.8***
Semi-supervised learning: labeled mono + labeled CS + unlabeled CS						
Label propagation	34.5	45.4***	43.3***	50.7	54.5***	52.9
Supervised learning: labeled mono + labeled CS						
Topline	34.4	28.6***	29.7***	51.4	39.1***	44.2***

not use the labeled CS at all while the topline model used only labeled data. Label propagation is a semi-supervised model retrained by newly labeling with the supervised model's output. Label propagation without any labeled CS data (semi-supervised learning: labeled mono + unlabeled CS) performed the worst on *all* evaluation set of dev_{man} and dev_{sge}. It used the CS label generated from the monolingual model, which does not know the CS speech and text, for the retraining model. As a result, although the performance on monolingual test sets is better than the proposed model since it retrains with the hypothesis generated from the model based on monolingual data, it degraded the performances both on *mono* and CS test sets than the baseline model in TER. Therefore, we removed 300 utterances as unlabeled CS and added 300 utterances (0.2% of the total CS) as labeled

Table 5.8: ASR performances of a single-pair CS (natural speech; multi-speaker) machine speech chain in CER %. The left side of arrows in the LID is λ_{Lng} value during the supervised learning process, and the right side is λ_{Lng} value during the semi-supervised learning process, where $\lambda_{Chr} = 1 - \lambda_{Lng}$. The p-values compared to the baseline system for statistical significance are presented using ***, **, * and no-star ($^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$).

Model	dev_{man}			dev_{sge}		
	mono	*CS*	*all*	*mono*	*CS*	*all*
Supervised learning: labeled mono						
Baseline	32.4	56.9	52.4	34.9	59.3	47.0
Semi-supervised learning: labeled mono + unlabeled CS						
Label propagation	36.4	55.5***	52.0	40.9	59.3	50.1
Proposed speech chain	39.8	39.6***	39.6***	39.1	44.4***	41.7***
+LID (0.25→0.0)	34.0	33.3***	33.5***	35.2	38.3***	36.7***
+LID (0.25→0.1)	35.5	34.6***	34.7***	36.3	39.0***	37.6***
Semi-supervised learning: labeled mono + labeled CS + unlabeled CS						
Label propagation	33.4	53.1***	49.5***	37.0	55.4***	46.1*
Supervised learning: labeled mono + labeled CS						
Topline	35.2	28.7***	29.9***	37.9	36.5***	37.2***

CS. The label propagation result (semi-supervised learning: labeled mono + labeled CS + unlabeled CS) improved slightly from the baseline on the *CS* test, but it required labeled CS and was not better than the proposed model. On the other hand, our proposed speech chain model improved the ASR performance without any labeled CS. It showed statistically significant improvements with $p < .001$ on *CS* and *all* of both evaluation sets. Moreover, the performance with LID is even better. The LID(0.25→0.0) produced 32.5% TER on dev_{man} *CS* and 42.3% TER on dev_{sge} *CS*.
We also checked the CER for the ASR performances (Table 5.8). It showed the same tendency as the TER results. Therefore, the proposed machine speech chain model improved the performance on SEAME data without any labeled CS data.

Table 5.9: Comparison performance (in CER%) between ASR baselines with/without LID. The p-values compared to the baseline system for statistical significance are presented using ***, **, * and no-star ($^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$).

Train: Ja25k+En25k+Zh25k	$(\lambda_{Chr},\lambda_{Lng})$	**Ja**	**En**	**Zh**
ASR without LID [chr]	No LID	8.8	9.1	5.8
ASR with LID [chr,lng]	(1,1)	8.9	8.5	5.1
ASR with LID [chr,lng]	(0.75,0.25)	7.3^{***}	7.3^{***}	5.1^{*}

Results on Multi-pair Code-switching

The experimental results on multi-pair CS are described separately in ASR and TTS evaluations.

ASR Evaluation

First, we checked the influence of the additional LID architecture to confirm whether that additional information hindered the original quality. We used the baseline model, an ASR **Ja25k+En25k+Zh25k (labeled)** trained with labeled monolingual data of 25-k Japanese and 25-k English and 25-k Chinese. Table 5.9 shows the baseline performance of ASR without LID that only generated character transcription and of ASR with LID that generated both character and language information sequences. In the case of $(\lambda_{Chr}, \lambda_{Lng}) = (1, 1)$, we found there was no statistically significant difference from ASR without LID in any of the tests. However, in the case of $(\lambda_{Chr}, \lambda_{Lng}) = (0.75, 0.25)$, which are the λ values during the supervised learning process of the best model on a single CS (Table 5.5), the results raised the possibility that the architecture with LID could assist the ASR performance. Hereafter, we show only the λ_{Lng} value such as (0.25) instead of (0.75, 0.25), and use $\lambda_{Chr} = 1 - \lambda_{Lng}$.

Table 5.10: ASR performance in CER% of multi-pair CS machine speech chain (The bold figures show the unused CS during training). The value inside the LID brackets shows λ_{Lng} value, where $\lambda_{Chr} = 1 - \lambda_{Lng}$. The left side of arrows in the LID is λ_{Lng} value during the supervised learning process, and the right side of arrows is λ_{Lng} value during the semi-supervised learning or fine-tuning process. The p-values compared to the baseline system for statistical significance are presented using ***, **, * and no-star ($^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$).

		Monolingual			Code-switching		
	LID type	**Ja**	**En**	**Zh**	**EnJa**	**JaZh**	**ZhEn**
[Baseline] Supervised learning of only labeled monolingual data							
Ja25k+En25k+Zh25k	No LID	8.8	9.1	5.8	**11.5**	**12.3**	**13.3**
	LID(0.25)	7.3***	7.3***	5.1*	**10.1**	**11.2****	**11.4*****
[Machine Speech Chain] Semi-supervised learning of unlabeled two CS data							
+JaEnCS10k+JaZhCS10k	No LID	8.8	9.7	5.9	8.2***	6.9***	**7.7*****
	LngChr	8.9	9.2	5.4	7.9***	7.2***	**7.2*****
	LID(0.25→0.0)	8.3	7.6	5.2	7.7***	6.7***	**7.1*****
	LID(0.25→0.1)	8.6	8.4	5.1	8.6***	6.9***	**7.4*****
+EnJaCS10k+ZhEnCS10k	No LID	8.9	9.9	5.9	8.5***	**7.0*****	7.5***
	LngChr	9.1	9.3	5.6	8.3***	**7.1*****	7.4***
	LID(0.25→0.0)	8.5	7.4	5.7	7.8***	**6.8*****	7.1***
	LID(0.25→0.1)	8.7	8.8	5.3	8.1***	**7.4*****	7.2***
+ZhEnCS10k+JaZhCS10k	No LID	9.0	10.2	5.9	**8.6*****	7.0***	7.6***
	LngChr	9.0	9.4	5.5	**8.3*****	6.9***	7.4***
	LID(0.25→0.0)	8.5	7.5	5.2	**7.8*****	6.8***	6.9***
	LID(0.25→0.1)	8.8	8.8	5.2	**8.6*****	7.0***	7.7***
[Topline] Supervised learning of labeled two or three CS data							
+EnJaCS10k+JaZhCS10k	LID(0.25→0.0)	8.4	8.5	7.9	7.8***	6.4***	**6.8*****
+EnJaCS10k+ZhEnCS10k	LID(0.25→0.0)	8.3	8.0	7.2	7.7***	**6.5*****	6.6***
+ZhEnCS10k+JaZhCS10k	LID(0.25→0.0)	9.3	9.4	5.2	**7.8*****	6.6***	6.7***
+EnJaCS10k+JaZhCS10k +ZhEnCS10k	LID(0.25→0.0)	8.1	8.1	7.0	7.6***	6.4***	6.6***

Next, we investigated how our proposed approach performed on multi-pair CS, including the unknown CS excluded from the training data. We individually trained ASR and TTS using labeled monolingual Ja25k, En25k, and Zh25k and carried out a machine speech chain on the following three different scenarios: (1) **EnJaCS10k+JaZhCS10k (unlabeled)**: semi-

Table 5.11: ASR performance in CER% of multi-pair CS machine speech chain using natural CS (The bold figures show the unused CS during training). The value inside the LID brackets shows λ_{Lng} value, where $\lambda_{Chr} = 1 - \lambda_{Lng}$. The left side of arrows in the LID is λ_{Lng} value during the supervised learning process, and the right side of arrows is λ_{Lng} value during the semi-supervised learning or fine-tuning process. The p-values compared to the baseline system for statistical significance are presented using ***, **, * and no-star ($^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$).

		Monolingual			Code-switching			
	LID type	**Ja**	**En**	**Zh**	**EnJa**	**JaZh**	**ZhEn**	**NatEnJa**
[Baseline] Supervised learning of labeled monolingual data								
Ja25k+En25k+Zh25k plus	No LID	8.8	9.7	5.4	**11.5**	**13.4**	**14.0**	**33.0**
NatJa0.2k+NatEn0.2k	LID(0.25)	8.6	7.7	5.1	**10.9**	**11.0*****	**11.0*****	**31.8**
[Machine Speech Chain] Semi-supervised learning of unlabeled two CS and one natural CS data								
+EnJaCS10k	No LID	9.2	12.1	5.4	9.3***	7.7***	**8.7*****	14.2***
+JaZhCS10k plus	LngChr	9.2	11.0	6.3	9.0***	8.0***	**9.6*****	14.0***
NatEnJaCS0.7k	LID(0.25→0.0)	$\underline{8.8}$	$\underline{10.1}$	5.6	$\underline{8.7}^{***}$	$\underline{7.2}^{***}$	$\underline{\mathbf{7.9}}^{***}$	$\underline{11.8}^{***}$
	LID(0.25→0.1)	9.3	11.0	$\underline{5.3}$	9.4***	9.1***	**8.5*****	13.7***
[Topline] Supervised learning of labeled two CS and one natural CS data								
+EnJaCS10k +JaZhCS10k plus NatEnJaCS0.7k	LID(0.25→0.0)	8.7	9.0	7.1	7.6***	6.8***	**6.9*****	9.6***

supervised learning with unlabeled EnJaCS 10k and JaZhCS 10k (ZhEnCS for a zero-shot target); (2) **EnJaCS10k+ZhEnCS10k (unlabeled)**: semi-supervised learning with unlabeled EnJaCS 10k and ZhEnCS 10k (JaZhCS for a zero-shot target); (3) **EnZhCS10k+ZhJaCS10k (unlabeled)**: semi-supervised learning with unlabeled EnZhCS 10k and ZhJaCS 10k (EnJaCS for a zero-shot target).

There are four types for LID: (1) the "No LID" systems without using language information; (2) "LngChr" systems, which output the language information with character together like (Jp-a, Jp-b, Jp-c, ..., Jp-z, En-a, En-b, En-c, ..., En-z, Zh-a, Zh-b, Zh-c, ..., Zh-z); (3) LID (0.25→0.0), where

0.25 is λ_{Lng} value during supervised learning process and 0.0 is λ_{Lng} value during semi-supervised learning process; (4) LID (0.25→0.1), where 0.25 is λ_{Lng} value during supervised learning process and 0.1 is λ_{Lng} value during semi-supervised learning process.

As Table 5.10 shows, compared to the baseline model, our proposed model of any language pairs improved the ASR performance on all CS, including zero-shot CS. Compared to the No LID and LngChr, the LID (0.25→0.0) overcame them on all CS test sets. All of the machine speech chain models showed statistical significance with $p < .001$.

We also investigated whether our proposed machine speech chain improved the ASR performance on multi-pair CS, including the natural speech CS. We applied the last model among the trained models of 60 epochs since we could not include the natural speech CS in the development sets. Since natural CS may switch multiple times within a single utterance, it tends to be more complicated than synthetic one. Besides, the natural CS was just only 1k, which is insufficient for training. As shown in Table 5.11, the performances were not as good as only the synthetic data. However, our proposed machine speech chain model improved ASR, showing statistical significance in the multi-pair CS, including the natural speech CS.

We also investigated French and Chinese CS (FrZhCS) performance with French as an unknown language. Since the system has never been trained with French data in supervised learning, it did not have a chance to learn the relation between French speech and the corresponding transcription. Table 5.12 shows the ASR performance. The results reveal that the proposed model still improved the ASR performance on the FrZhCS test data even though no monolingual French labeled data are available, and even though the French language is unknown. In the "+EnJaCS10k+JaZhCS10k (unlabeled)", we performed the zero-shot CS using a completely new or unseen language. We confirmed the improvement from baseline even in that case. However, the model (No LID) was better than the proposed model with LID since the LID could not learn the "unknown" label for the French language. When we added the EnFrCS for semi-supervised training, the proposed model with LID improved from that without LID. The LngChr model could not get a good result, showing it was difficult to handle the unknown language. Moreover, it increased the error by adding the EnFrCS training data. However, our proposed model improved the performance. However, our proposed model improved the performance.

Table 5.12: ASR performance (in CER%) of a multi-pair CS machine speech chain on the zero-shot CS with the unknown language, where labeled monolingual French data are unavailable, and the French language is unknown. The value inside the LID brackets shows λ_{Lng} value, where $\lambda_{Chr} = 1 - \lambda_{Lng}$. The left side of arrows in the LID is λ_{Lng} value during the supervised learning process, and the right side of arrows is λ_{Lng} value during the semi-supervised learning or fine-tuning process. The p-values compared to the baseline system for statistical significance are presented using ***, **, * and no-star ($^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$).

Train data	LID type	**Unknown FrZhCS**
[Baseline] Supervised learning of labeled monolingual data		
Ja25k+En25k+Zh25k (labeled)	No LID	33.7
	LID (0.25)	33.0
[Machine Speech Chain] Semi-supervised learning of unlabeled CS data		
+EnJaCS10k+JaZhCS10k (unlabeled)	No LID	25.4***
	LngChr	35.8
	LID(0.25→0.0)	26.3***
	LID(0.25→0.1)	27.6***
+EnJaCS10k+JaZhCS10k+EnFrCS10k (unlabeled)	No LID	24.0***
	LngChr	44.8
	LID(0.25→0.0)	22.4***
	LID(0.25→0.1)	21.6***
[Topline] Supervised learning of labeled CS data		
+EnJaCS10k+JaZhCS10k+EnFrCS10k (labeled)	LID (0.25→0.0)	16.4***

TTS Evaluation

We evaluated the zero-shot CS speech generated by the TTS of the proposed multilingual and zero-shot CS machine speech chain. We conducted an AB

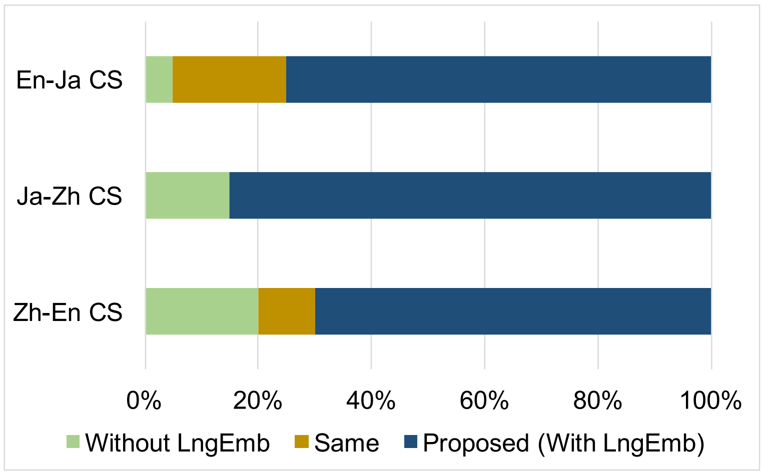


Figure 5.9: Comparison of AB preference subjective evaluation between generated zero-shot CS speech from the model with/without language-embedding.

preference subjective evaluation between the generated zero-shot CS speech from the model with/without language-embedding $[y^{CSChr}, y^{CSLng}]$. All language pairs of the zero-shot CS were evaluated by ten bilingual speakers who compared two speech utterances while looking at the transcription and chose which speech was better in terms of being more native. They were also given the option to admit they could not determine which sounded more native. They compared 20 pairs shown randomly. The results (Fig. 5.9) show our method supports the quality of synthesized speech; particularly on the switching places between two languages.

5.5 Summary

In this chapter, we introduced a machine speech chain for semi-supervised learning of CS ASR and TTS. Unfortunately, common methods of developing CS ASR and TTS are separate training, where just CS ASR or CS TTS is developed. Moreover, it relies on supervised learning that requires large amounts of CS data for training models. Although either CS text or CS speech may be found on social media, pairs of CS speech and corresponding CS transcriptions are scarce and difficult to obtain. Such a data problem hinders the development of CS ASR. Therefore, we proposed utilizing the machine speech chain framework for CS tasks. The machine speech chain is the mechanism trainable with semi-supervised learning using labeled and unlabeled data.

We beforehand confirmed that the Japanese ASR could not perform well on the English test, and the English ASR could not perform well on the Japanese test. The multilingual ASR trained with Japanese and English data controlled the balance well among the Japanese, English, and Japanese-English CS test set. Therefore, we set its multilingual ASR as the baseline model and aimed to improve ASR to handle CS input better without any labeled CS data utilizing semi-supervised learning of the machine speech chain.

For the machine speech chain training, we first individually trained ASR and TTS systems with labeled monolingual data in supervised learning. Then we carried out a machine speech chain with semi-supervised learning of either CS text or CS speech.

We investigated CS ASR and TTS improvements on single-pair CS (synthetic speech; single speaker), single-pair CS (natural speech; multi-speaker), and multi-pair CS. Our results revealed that such a mutually complementary architecture of machine speech chain trains ASR and TTS together and improves performance even without any labeled CS data on all the scenarios. Although monolingual performances are often damaged by optimizing the CS performance, our proposed machine speech chain model improved the CS ASR and CS TTS's performance while maintaining the monolingual input's performance.

We also introduced a language-aware CS machine speech chain. We expanded our model to handle CS better by integrating language embedding and LID into the machine speech chain. Although we had to tune the weight value of the LID loss parameter, we confirmed that the machine speech chain model with language embedding and LID could produce satisfactory performances both on a single-pair CS and multi-pair CS. On the multi-pair CS, we also challenged the zero-shot CS. Zero-shot CS refers to the task predicting the unknown CS not included in the training data. We confirmed the proposed language-aware CS machine speech chain also performed well on the zero-shot CS, where we investigated the zero-shot CS that composed of the language pairs not only from the known language used during supervised learning but also the unknown language. The proposed model was also compared with the LngChr model, which model uses the output character added the language identifier, and we found that the proposed model handled the unknown language better than the LngChr model. In addition to that, we also investigated the speech quality of the zero-shot CS generated from the proposed language-aware CS machine speech chain, and we confirmed that it could maintain the quality of native sounds with the help of the language information.

Chapter 6

Proposed Code-switching Speech Translation

This chapter is about code-switching speech translation. We show the experimental results of several approaches for code-switching speech translations and discuss them.

6.1 Introduction

The common aim of developing a CS ASR is merely for transcribing CS-speech utterances into CS-text sentences, where we assume only dialogues between the

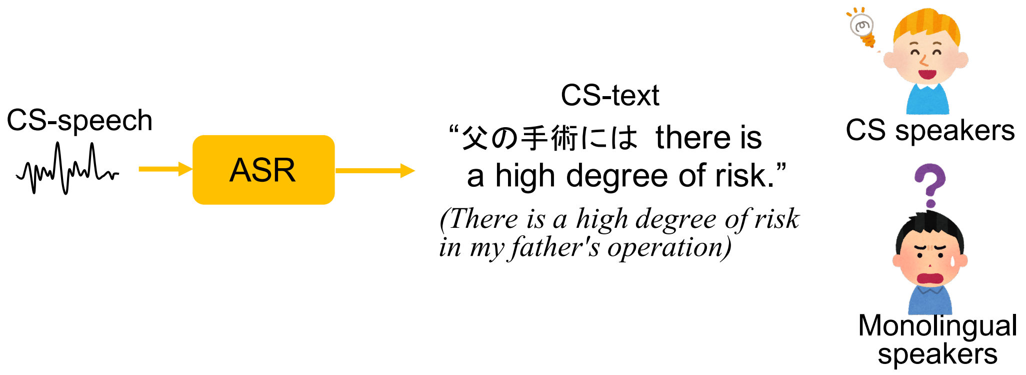

CS-speech
ASR
CS-text
“父の手術には there is a high degree of risk.”
(There is a high degree of risk in my father's operation)
CS speakers
?
Monolingual speakers

Figure 6.1: CS ASR assuming only dialogues between the same CS speakers.

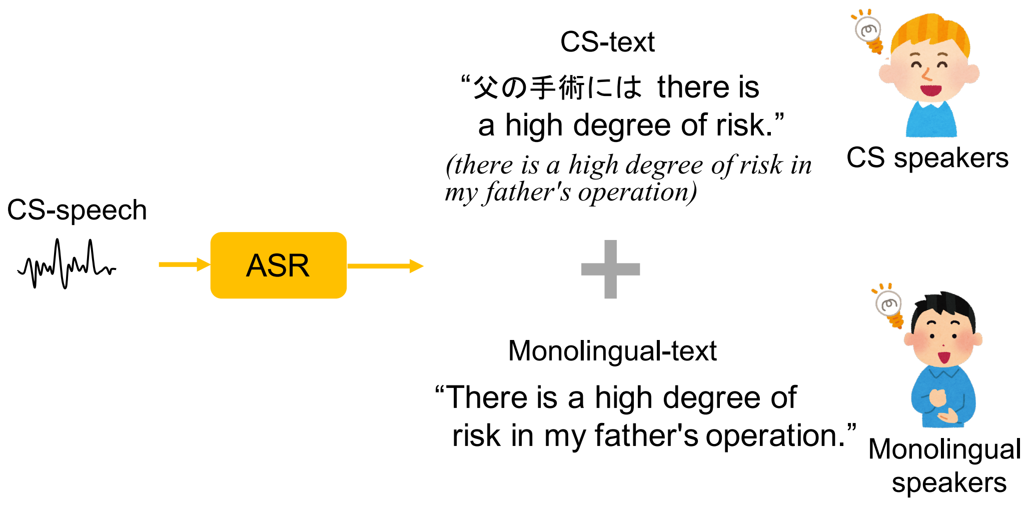

CS-speech
ASR
CS-text
“父の手術には there is a high degree of risk.”
(there is a high degree of risk in my father's operation)
CS speakers
Monolingual-text
“There is a high degree of risk in my father's operation.”
Monolingual speakers

Figure 6.2: CS ASR supporting monolingual speakers.

same CS speakers as Fig. 6.1 shows. In contrast, our study addresses the situational context during dialogues between CS and non-CS (monolingual) speakers to support monolingual speakers trying to understand CS speakers as Fig. 6.2 shows. CS is also used during interactions between CS and non-CS speakers. For example, more than half of the immigrant children in the U.S. have at least one parent who cannot speak English well [73]. The children can become bilingual since they speak English at school and speak their native language at home. In such cases, their parents cannot understand when their children talk to them in English. Therefore, we need to construct a system that recognizes the CS speech and translates it to monolingual text so that the monolingual speakers can understand CS speakers.

6.2 Related Works

If we assume text-to-text translation, there are several studies for CS translation. Sinha et al. developed text-to-text CS translation by separating CS text to monolingual fragments [74]. However, it cannot consider the context beyond languages. Johnson et al. proposed the text-to-text multilingual translation, which translates to the desired language by designating the target language in input, and it implied the possibility for text-to-text CS translation [67]. However, in the case of applying it to speech-to-text translation, we have to concatenate speech with the text designating the target language, and it degrades the performance without applying the normalization. Moreover, if we train the multiple target languages with the same decoder model, the bias between mutual languages degrades the performance. Menacer et al. attempted several approaches for translating Arabic-English CS text [75]. The best system with a high BLEU score was copying the input text to output text for the part which does not have to be translated, and the rest parts were translated with the model trained using multilingual languages. However, in the case of speech-to-text translation, we cannot copy the input to output. Moreover, translating the whole input sentence using the multilingual model without copying had the lowest BLEU score among their compared approaches.

Anyway, this work attempts to realize the speech-to-text CS translation, not text-to-text translation. To address the problems, we investigate several approaches: a cascade of neural machine translation (NMT) from ASR, a cascade of Bidirectional Encoder Representations from Transformers (BERT) from ASR, a direct single-task speech translation, and a direct multi-task speech translation. We evaluate and discuss these four ways on a Japanese-English CS to English monolingual task and on a Japanese-English CS to Japanese monolingual task.

6.3 Proposed Approaches

We propose two cascade approaches and two direct approaches to perform speech translation ST from CS speech to monolingual text. The two cascade approaches are the methods of machine translation using BERT or NMT for CS text transcribed from CS speech by ASR. The two direct approaches output monolingual text from CS speech directly in single-task or multi-task without going through the process of transcribing to CS text by ASR. We introduce these proposed methods one by one.

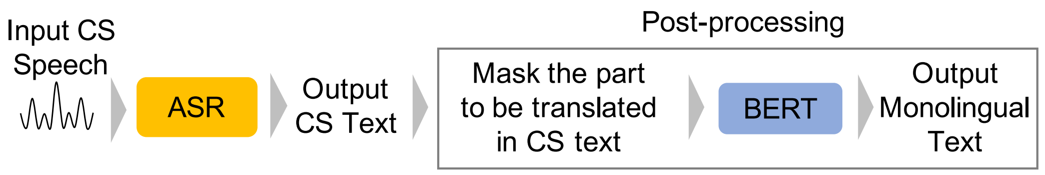


Figure 6.3: Model architecture of Cascade ASR+BERT.

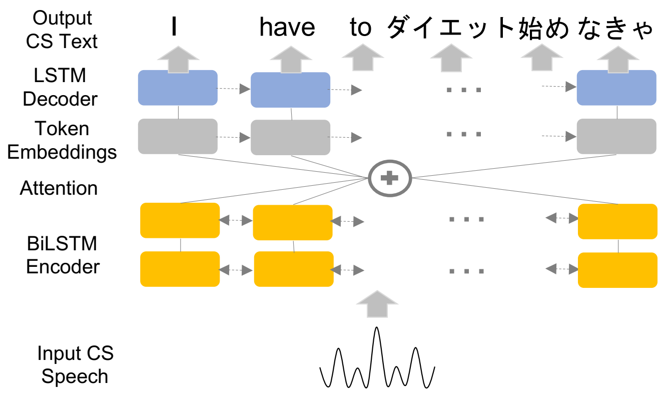


Figure 6.4: Model architecture of ASR.

6.3.1 Cascade Approaches

Cascade ASR+BERT

A first cascade approach is an approach that uses BERT. The cascade structure is depicted in Fig. 6.3. From CS speech, a neural ASR produces CS text. Then we mask the part to be translated in CS text, and the BERT model recovers the monolingual text from it. Fig. 6.4 shows the ASR system, a standard attention-based encoder-decoder ASR [16, 17]. The encoder has BiLSTM layers, and the decoder has an embedding layer and LSTM layer. The attention module maps between encoder and decoder. The ASR system is the same as Chap. 2.2.2, so the loss function is Eq. (2.12).

The BERT model architecture is depicted in Fig. 6.5. It consists of Transformers [76]. BERT [39] is a language understanding model that has a deeper sense of language context than traditional language models (LMs). Traditional LMs are based on a single-directional (left-to-right) approach that predicts the next word given a sequence. Unfortunately, such an approach limits the learning

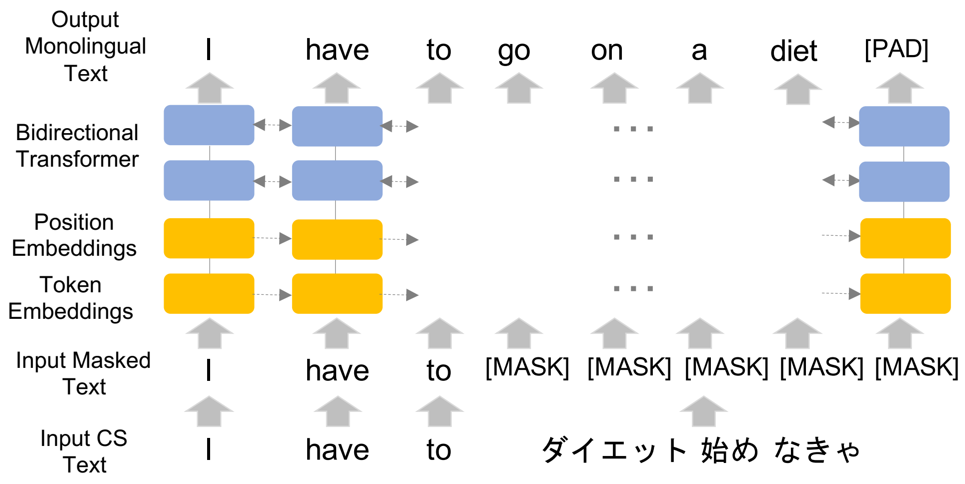


Figure 6.5: Model architecture of BERT for CS translation.

of context. On the other hand, BERT learns context bi-directionally (left-to-right and right-to-left) with Transformer.

BERT has two training phases: (1) pre-training with a dataset for language representation and (2) fine-tuning on a specific task, such as sentiment analysis [77], question answering [78], name entity recognition [79]. In the pre-training phase, BERT randomly replaces some tokens with [MASK] tokens. It predicts the original tokens hidden under the [MASK] by learning the representations using other tokens. Ghazvininejad et al. also utilized conditional masked language models like BERT for translation tasks by introducing a new mask-prediction algorithm [80] that repeatedly selects the new positions of the mask tokens and predicts them at each iteration. We also utilized the pre-trained BERT that leverages a masked language model (Masked LM). In our case, given a CS text

Table 6.1: Example of the monolingual text recovered by BERT.

Source CS	i have to ダイエット 始め なきゃ before my belly explodes
Masked text	i have to [MASK] [MASK] [MASK] [MASK] [MASK] before my belly explodes
Label	i have to go on a diet [PAD] before my belly explodes
Target English	i have to go on a diet before my belly explodes

that mixed words from the 1st and 2nd languages, we masked unwanted words from the 2nd language and used BERT to recover complete sentences in the monolingual text of the 1st language. Since we do not know exactly how many words should be replaced, we put several [MASK] tokens in the positions of unwanted words. Then the model is filled with tokens [PAD] if the original target token size is smaller than the number of [MASK] tokens. Table 6.1 shows an example of monolingual text recovery using a BERT.

Cascade ASR+NMT

Another cascade approach uses NMT. Fig. 6.6 shows the cascade structure. Given CS speech, we first perform a neural ASR and produce the CS text. After that, we utilize NMT to translate from CS text to monolingual text. The ASR system is the same architecture as Fig. 6.4. Our NMT system is a standard attention-based encoder-decoder model [16, 17] as with the ASR system. The encoder has BiLSTM layers, and the decoder has LSTM layers. The model architecture is depicted in Fig. 6.7, and the loss function is the same as Eq. (2.12).

6.3.2 Direct Approaches

Direct Single-task Speech Translation

Direct single-task speech translation (Direct single-task ST) is the speech translation system directly predicting monolingual text from CS speech. We trained the model directly predicting monolingual text from CS speech using the same architecture as the ASR system. Fig. 6.8 shows this architecture, and the loss function is the same as Eq. (2.12).

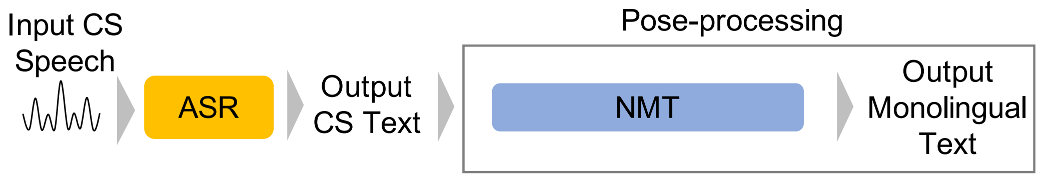


Figure 6.6: Model architecture of Cascade ASR+NMT.

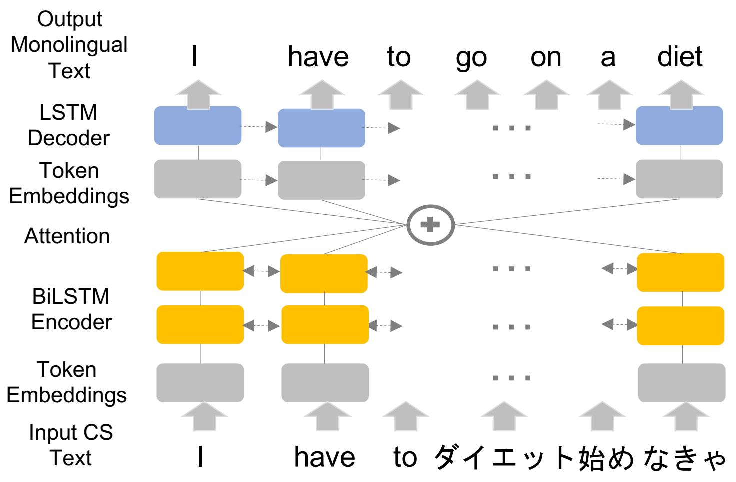


Figure 6.7: Model architecture of NMT.

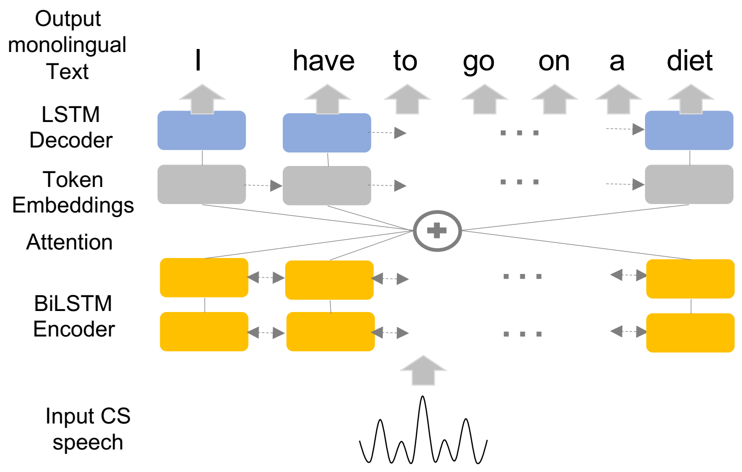


Figure 6.8: Model architecture of Direct single-task speech translation.

Direct Multi-task Speech Translation

Direct multi-task speech translation (Direct multi-task ST) is the speech translation system while training monolingual text and CS text from CS speech with multi-task learning. Multi-task learning has variations, but we adopted the typical multi-task learning [81] having two decoders with shared an encoder. The first decoder outputs CS text, and the second outputs monolingual text. This

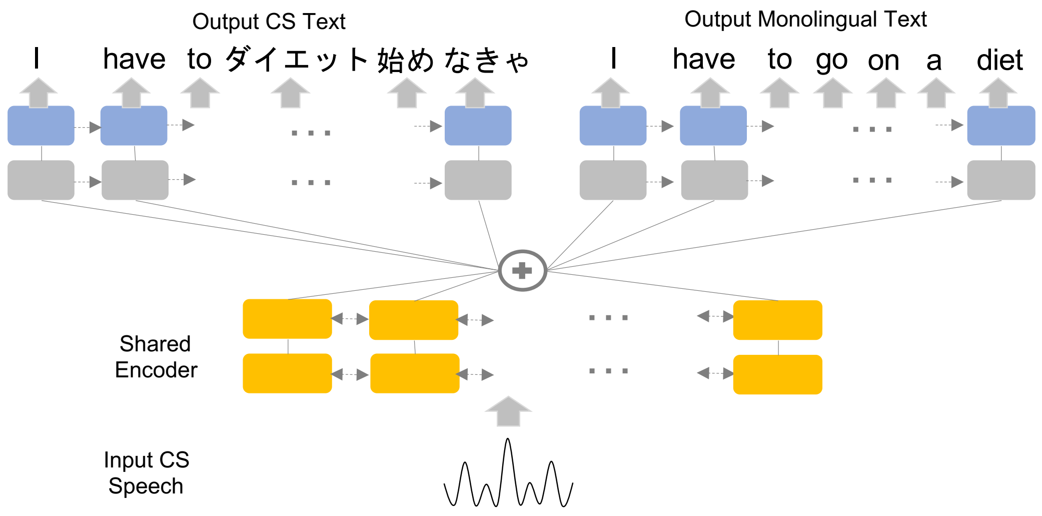


Figure 6.9: Model architecture of Direct multi-task speech translation.

model architecture is depicted in Fig. 6.9. The loss function is the following:

$$L_{MTL} = (1 - \lambda)L_{ASR} + \lambda L_{NMT}, \tag{6.1}$$

where L_{ASR} is the loss of CS text and L_{NMT} is the loss of monolingual text. λ is the hyperparameter to balance the weight between L_{ASR} and L_{NMT}.

Direct Multi-task+LID Speech Translation

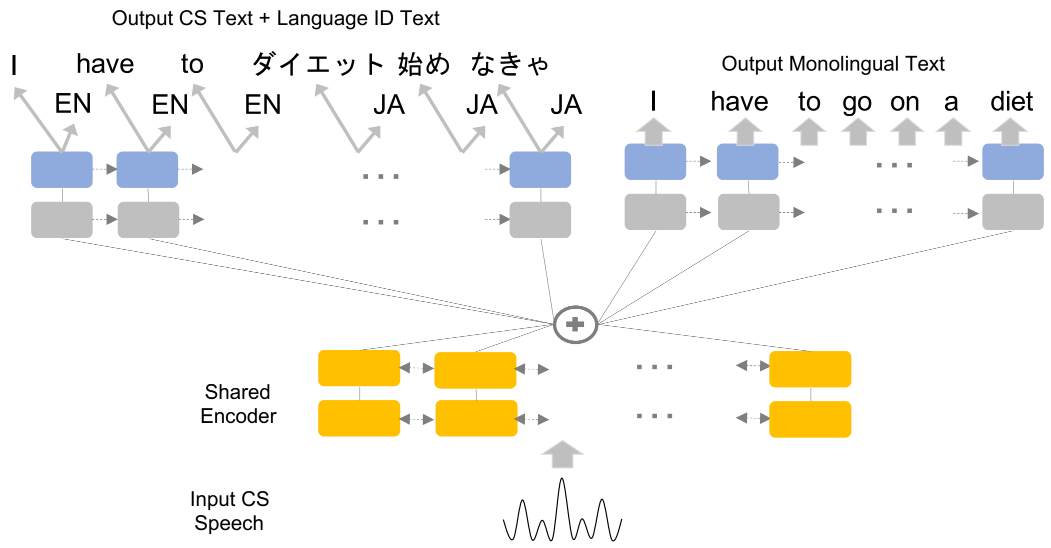


Figure 6.10: Model architecture of Direct multi-task+LID speech translation.

Direct multi-task+LID speech translation (Direct multi-task+LID ST) is the speech translation system incorporating the LID system into the above Direct multi-task ST. As Fig. 6.10 shows, it shares the encoder and has two decoders for multi-task learning. The two decoders predict CS text and monolingual text as the same as the Direct multi-task ST. In addition to that, the first decoder also identifies language (LID) using two softmax layers, which LID architecture is the same as the proposed LID system in Chap. 4 and Chap. 5. The loss function is extended to the following from that of Direct multi-task ST:

$$L_{MTL+LID} = (1-\lambda)((1-\lambda_{LID})L_{ASR} + \lambda_{LID}L_{LID}) + \lambda L_{NMT}, \tag{6.2}$$

where L_{LID} is the loss of output language ID text. λ and λ_{LID} are the hyperparameters to balance the weight between L_{ASR}, L_{LID}, and L_{NMT}.

6.4 Experiments

6.4.1 Experimental Settings

Dataset Composition

We used the intra-sentential word-level CS and intra-sentential phrase-level CS I of synthetic speech CS (Sect. 3.2.2). They are synthesized using the corresponding language's TTS, where Japanese words are synthesized with Japanese TTS, and English words are synthesized with English TTS. We prepared "All artificial CS" and "Mix natural CS," where "All artificial CS" corpus includes only synthetic speech CS, and "Mix natural CS" corpus added natural speech CS to the "All artificial CS" corpus. We applied the data augmentation approach with speed perturbation to natural speech CS since it has only 900 utterances for training set [82, 83]. We applied the speed perturbation on 90%, 100%, 110% so that we got triple-valued 2700 utterances.

Table 6.2 shows the statistics of the training and evaluation corpora for a single-pair CS experiment. We used EnJaCS for the CS-to-English translation tasks and used JaEnCS for the CS-to-Japanese translation tasks.

For the multi-pair CS experiment, we added the Japanese-Chinese CS of intra-sentential phrase-level CS I. They are also synthesized using the corresponding language's TTS, where Japanese words are synthesized with Japanese TTS, and Chinese words are synthesized with Chinese TTS. Table 6.3 shows the statistics of training and evaluation data for multi-pair CS speech to English text, and Table 6.4 shows the statistics of training and evaluation data for multi-pair CS speech to Japanese text.

We tokenized all the text. We applied a morphological analyzer Mecab [55] for Japanese sentences and applied WordPiece [84] for English sentences. The WordPiece is a subword unit for efficiently reducing the unknown words.

Table 6.2: Statistics of the training and evaluation corpora for single-pair CS speech translation.

Subsets			Hours	Utterances		
				Synthetic speech	Natural speech	Total
Train	EnJaCS	All artificial CS	76	50k	-	50k
		Mix natural CS	81	50k	900x3	52.7k
	JaEnCS	All artificial CS	95	100k	-	100k
		Mix natural CS	100	100k	900x3	102.7k
Test	EnJaCS	All artificial CS	0.8	500	-	500
		Mix natural CS	0.8	400	100	500
	JaEnCS	All artificial CS	1.1	500	-	500
		Mix natural CS	1.2	400	100	500

Table 6.3: Statistics of the training and evaluation corpora for translation from multi-pair CS speech to English text.

			Hours	Utterances			
				Synthetic speech		Natural speech	Total
				EnJaCS	EnZhCS		
Train	All artificial CS		152	50k	50k	-	100k
	Mix natural CS		157	50k	50k	900x3	102.7k
Test	All artificial CS	EnJaCS	0.8	500	-	-	500
		EnZhCS	0.8	-	500	-	500
	Mix natural CS		0.8	400	-	100	500

Table 6.4: Statistics of the training and evaluation corpora for translation from multi-pair CS speech to Japanese text.

			Hours	Utterances			
				Synthetic speech		Natural speech	Total
				JaEnCS	JaZhCS		
Train	All artificial CS		183	100k	50k	-	150k
	Mix natural CS		188	100k	50k	900x3	152.7k
Test	All artificial CS	JaEnCS	1.1	500	-	-	500
		JaZhCS	0.9	-	500	-	500
	Mix natural CS		1.2	400	-	100	500

Model Details

Our ASR system is an attention-based encoder-decoder ASR [16,17] in Chap. 2.2.2. The encoder has three stacked BiLSTM layers that have 256 hidden units for each direction. The decoder has a 128-dims embedding layer and one LSTM layer with 512 hidden units. We used a log-scaled Mel-spectrogram as an input features and used a LeakyReLU ($l = 1e - 2$) [59] as an activation function. The alignment score of the attention module mapping between encoder and decoder is calculated with MLP [15].

Our NMT system is a standard attention-based encoder-decoder model [16,17] as with the ASR system. The encoder has two stacked BiLSTM layers with 256 hidden units for each direction (512 hidden units in both directions). The decoder has two LSTM layers with 512 hidden units.

In the Direct multi-task speech translation, the shared encoder and ASR decoder have the same hyperparameters as the ASR model. The translation decoder has the same hyperparameters as the decoder of the NMT model. For the λ value of Eq. (6.1) and Eq. (6.2), we mainly took the value of 0.5. The λ_{LID} of Eq. (6.2) mainly took the value of 0.1.

For the BERT model, we followed the hyperparameters and the weight initialization scheme to the BERT$_{Base}$ model, which is a publicly available BERT English model [39]. It consists of Transformers [76] with 12 layers, 768 hidden sizes, and 110-M parameters.

Table 6.5: WER%↓ of BERT and NMT translation from CS text to English text.

Test	BERT model All artificial CS	BERT model Mix natural CS	Test	NMT model All artificial CS	NMT model Mix natural CS
All artificial CS	11.14	12.14	All artificial CS	7.47	6.56
Mix natural CS	20.61	19.53	Mix natural CS	37.56	17.94

Table 6.6: BLEU↑ of BERT and NMT translation from CS text to English text.

Test	BERT model All artificial CS	BERT model Mix natural CS	Test	NMT model All artificial CS	NMT model Mix natural CS
All artificial CS	78.46	79.42	All artificial CS	86.54	88.11
Mix natural CS	72.03	73.11	Mix natural CS	57.46	77.86

6.4.2 Results

We conduct experiments on single-pair CS and multi-pair CS. Evaluation matrix uses word error rate (WER) and BLEU score [85]. The lower WER, the better. The higher BLEU, the better.

Results on Single-pair Code-switching

On single-pair CS, we first conduct the text-to-text translation from CS text to monolingual English text as preliminary experiments and then perform the speech-to-text translation experiments from CS speech to English and Japanese text.

Code-switching Text to Monolingual Text
Although our final goal is the evaluation for speech translation, we also evaluated the text-to-text translation to investigate the effect of ASR error. Table 6.5 and Table 6.6 show the result of BERT and NMT translation performance from CS text to English text in WER and BLEU, respectively. Compared between BERT and NMT, the NMT model tends to be better. However, only the Mix natural CS evaluation on the All artificial CS, the BERT model has higher performance than the NMT model. In the BERT, the part not to be translated in CS is copied from the input text, so the

performance can be good if the input text does not include an error from ASR. Therefore, from the next, we investigate the scenarios with the ASR error.

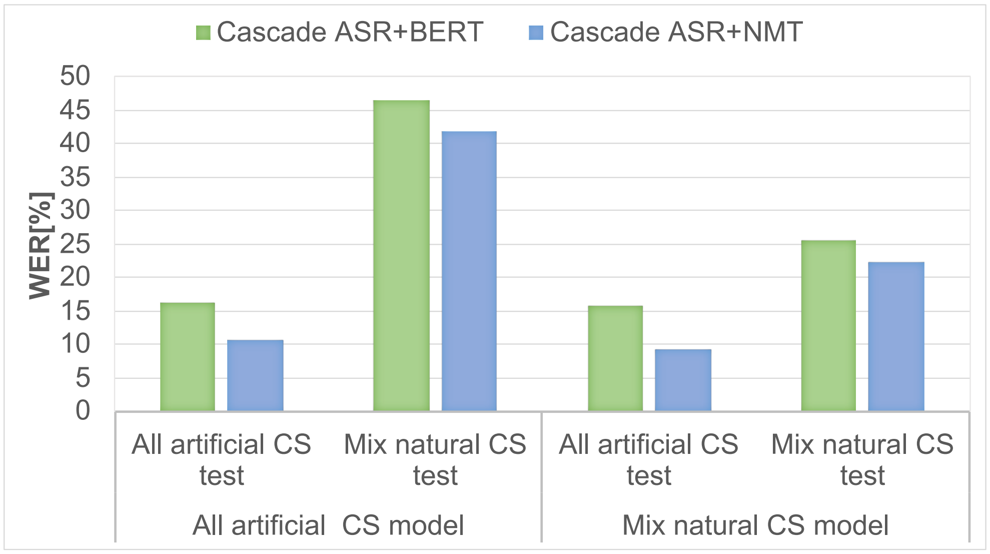


Figure 6.11: WER%↓ of Cascade ASR+BERT and Cascade ASR+NMT in CS speech to English translation.

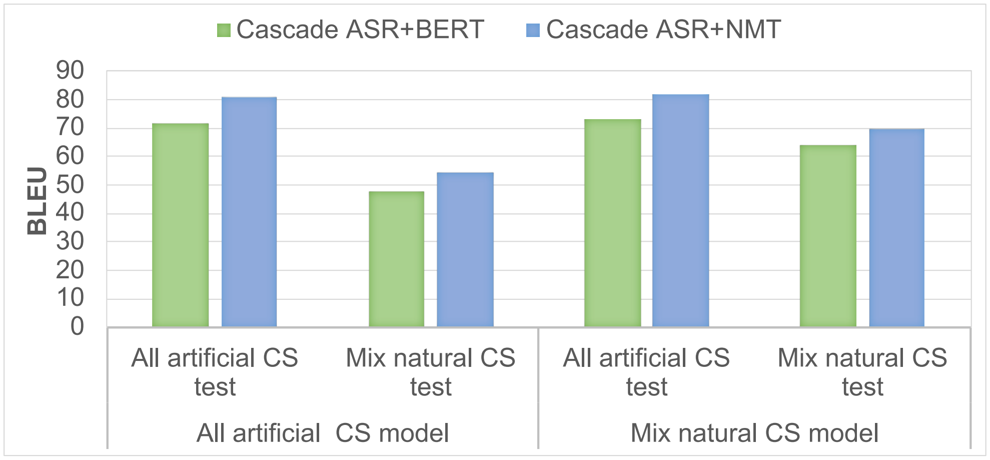


Figure 6.12: BLEU↑ of Cascade ASR+BERT and Cascade ASR+NMT in CS speech to English translation.

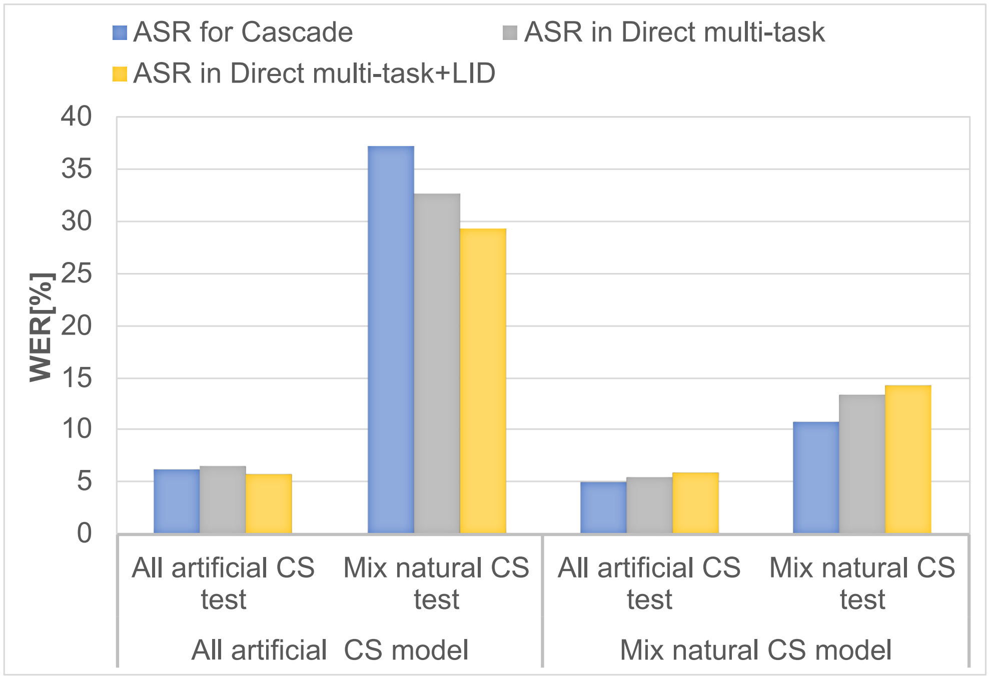


Figure 6.13: WER%↓ of CS output in CS speech to English translation.

Code-switching Speech to Monolingual Text

Fig. 6.11 and Fig. 6.12 show the result of the speech translation with the Cascade ASR+BERT and the Cascade ASR+NMT from CS speech in WER and BLEU, respectively. Compared between the Cascade ASR+BERT and the Cascade ASR+NMT, the Cascade ASR+NMT is better than the Cascade ASR+BERT in all cases. It seems the task became too hard for BERT, as the ASR error increased the number of [MASK] tokens.

Then we compared the ASR performances between the Cascade and Direct multi-task approaches. Fig. 6.13 shows the comparison result. Although the Direct multi-task approaches have the additional task of translating to monolingual, it does not much seem to hinder the ASR quality. Moreover, it even had the possibility of helping the performance. For example, the test case of Mix natural CS test by All artificial CS model is the most challenging case since the All artificial CS model has not seen the natural speech CS yet, so it increased the ASR error. However, the multi-task settings did not increase the error as much as the ASR for Cas-

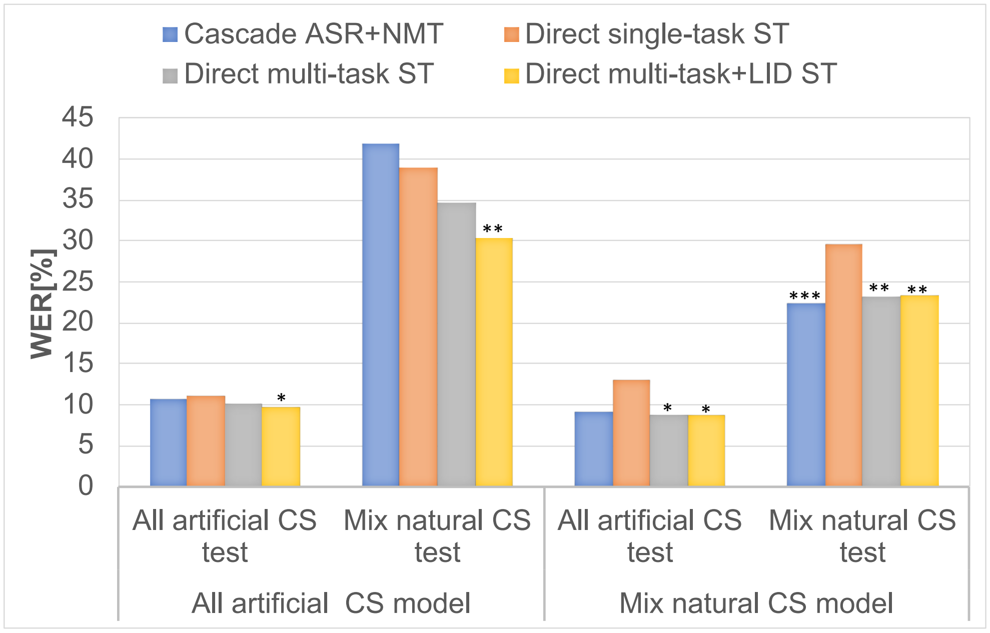


Figure 6.14: WER%↓ of English output in CS speech to English translation in Cascade and Direct approaches. P-values compared to Direct single-task ST are shown with $^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$.

cade. It might be thanks to training together in multi-task learning. If so, the translation task can also be better with the help of CS recognition. Moreover, since the CS recognition result can be regarded as the middle result for the speech translation, the translation task is expected to be even better with the help of the CS recognition task. The CS text is the middle result in the sense that it is the transcription of the input speech and also the parallel translation of the target monolingual. Therefore, we expected that multi-task learning could improve the translation performance with the help of the CS recognition task.

From the following paragraphs, we compare the Cascade ASR+NMT, the Direct single-task ST, Direct multi-task ST, and Direct multi-task+LID ST on the translation task from CS speech to English and Japanese text. Our proposed approach is the Direct multi-task ST, and we further confirm if the LID architecture can help the performance. We conduct the statistical significance test, where we compare the systems with the most straightfor-

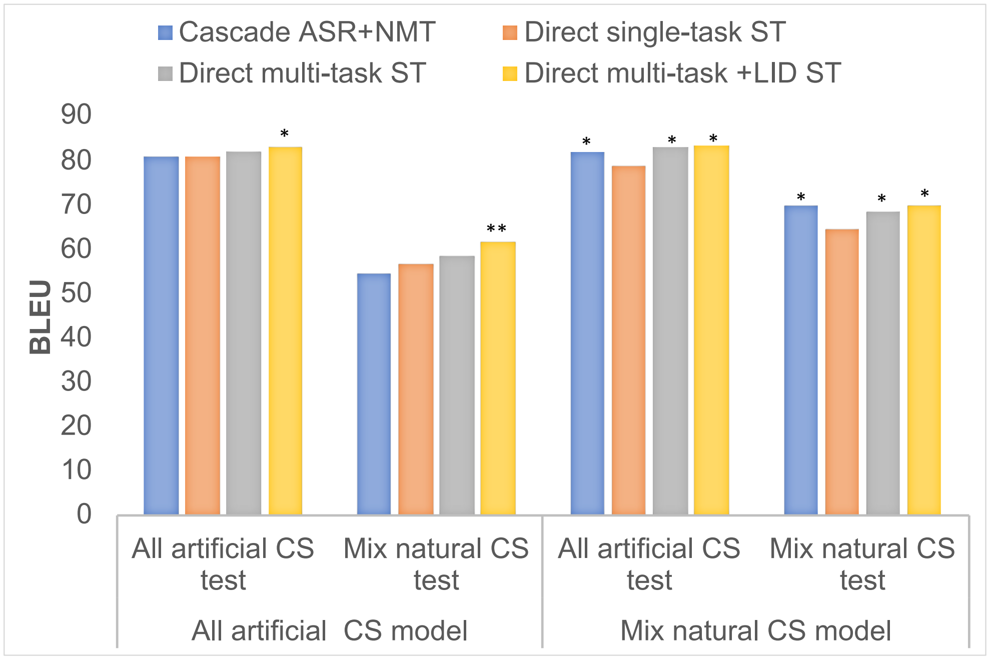


Figure 6.15: BLEU↑ of English output in CS speech to English translation in Cascade and Direct approaches. P-values compared to Direct single-task ST are shown with $^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$.

ward method, Direct single-task ST. We assessed the statistical significance by a matched-pair sentence-segment word error test [63] and the paired bootstrap test [86].

Fig. 6.14 and Fig. 6.15 show the comparisons of translation performance from CS speech to English text between Cascade and Direct approaches in WER and BLEU, respectively. Compared among the Cascade ASR+NMT, the Direct single-task ST, and the Direct multi-task ST, The Direct single-task ST tends to perform the worst, and the Direct multi-task ST tends to perform the best. First, the Direct single-task ST seems to be more difficult than other models since it needs to transcribe directly from CS speech into monolingual English transcriptions. Table 6.7 shows that only the Direct single-task ST could not grasp the keyword “homework.” On the other hand, the Direct multi-task ST could predict “homework” despite being the same Direct approach. In the Direct multi-task ST, we guessed

Table 6.7: Output examples showing that the Direct single-task ST is more difficult in CS to English translation.

(1)	Source text		if you want to watch tv , you should **宿題** は 終え た ほう が いい です よ .
	Reference text		if you want to watch tv , you should finish your **homework** first .
	Output	Cascade ASR+NMT	if you want to watch tv , you should go to you **homework** .
		Direct single-task ST	if you want to watch tv , you should be able to **talk** .
		Direct multi-task ST	if you want to watch tv , you should have any **homework** .
		Direct multi-task+LID ST	if you want to watch tv , you should give up some **homework** .

that transcribing CS text assisted in translating into monolingual English transcriptions. Therefore, Direct multi-task ST could predict better than Direct single-task ST.

The Cascade ASR+NMT was not so different from the Direct multi-task ST. However, the Cascade ASR+NMT degraded the performance, especially on the Mix natural CS test by the All artificial CS model. The Cascade ASR+NMT may degrade the performance because of the ASR error propagation. Table 6.8 shows that the Cascade ASR+NMT translated "手 荷物 事故 報告 書 (*property irregularity report*)" wrongly to "tennis." It seems to be error propagation from ASR since the ASR recognizes mistakenly "テニス (*tennis*)." Other approaches also have difficulty translating the compound words, but they do not have much ASR error propagation thanks to direct prediction. Therefore, the Direct multi-task ST tends to have a better performance. The Direct multi-task+LID ST generally showed even more improvement of the performance. However, when we first tried the experiments, the Direct multi-task+LID ST was not better than the Direct multi-task ST on the Mix natural CS model, although it was better on the All artificial CS model. We guessed that the LID could not train enough since natural speech CS data is few. Therefore, we first trained the model with only synthetic speech CS data with the LID loss λ_{LID} 0.1, and then

Table 6.8: Output examples of the case where Cascade ASR+NMT has ASR error propagation in CS to English translation.

(2)	Source text		i ' ll do my best to find your baggage but first i ' d like you to fill in this 手 荷物 事故 報告 書 .
	Reference text		i ' ll do my best to find your baggage , but first i ' d like you to fill in this property irregularity report .
	Output	ASR	i ' ll do my best to find your baggage but first i ' d like you to fill in this テニス も 時 を 報告 し う .
		Cascade ASR+NMT	i ' ll do my best to find your baggage but first i ' d like you to fill in this **tennis** .
		Direct single-task ST	i ' ll do my best to fly your baggage , but first i ' d like you to fill in this morning , so i ' ll do you ' ll be in the kanto person .
		Direct multi-task ST	i ' ll do my best to find your baggage , but first i like you to fill in this form to yourself in japan .
		Direct multi-task+LID ST	i ' ll do my best to find your baggage , but first i ' d like you to fill in this to my uncle ' s parents .

fine-tuned the model with synthetic speech CS + natural speech CS with the LID loss λ_{LID} 0.0. As a result, the performance improved a little bit. We expect that it can improve more if we could get more amount of natural speech CS. The BLEU score shown in Fig. 6.15 has the same tendency as WER. The tendency is that the Direct multi-task ST performs better than other approaches.

We also investigated which sentence is difficult for CS translation beyond system comparisons. Table 6.9 shows the additional output examples of speech translation from CS speech to English text. The sentences where the translation part can work as one sentence like (3) "どう し たら いい ん でしょ う ? (*what should i do?*)" could be translated well. On the other hand, the sentences which translation part is inserted into another language's phrases such as "you should 宿題 は 終え た ほう が いい です よ (*you should finish your homework*)" of Table 6.7 (1) tend to be difficult. The sentences that consist of the combination of words such as (4)"現金 です か , カード です か ? (*cash or charge*)" can be translated relatively easily, but the compound words that bridge over the multiple tokens such

Table 6.9: Output examples of speech translation from CS speech to English text on Mix natural CS test.

(3)	Source text		oh , no . i don ' t have any change so , どう し たら いいん でしょ う ?
	Reference text		oh , no . i don ' t have any change so , what should i do ?
	Output	Cascade ASR+NMT	oh , no . i don ' t have any change so , what should i do ?
		Direct single-task ST	oh , no . i don ' t have any change so , what should i do ?
		Direct multi-task ST	oh , no . i don ' t have any change so , what should i do ?
		Direct multi-task+LID ST	oh , no . i don ' t have any change . what should i do ?
(4)	Source text		how would you like to pay 現金 です か , カード です か ?
	Reference text		how would you like to pay , cash or charge ?
	Output	Cascade ASR+NMT	how would you like to pay , cash or charge ?
		Direct single-task ST	how would you like to pay , cash or charge ?
		Direct multi-task ST	how would you like to pay , cash or charge ?
		Direct multi-task+LID ST	how would you like to pay , cash or charge ?

as “手 荷物 事故 報告 書 (*property irregularity report*)” of Table 6.8 (2) seem to be difficult to translate.

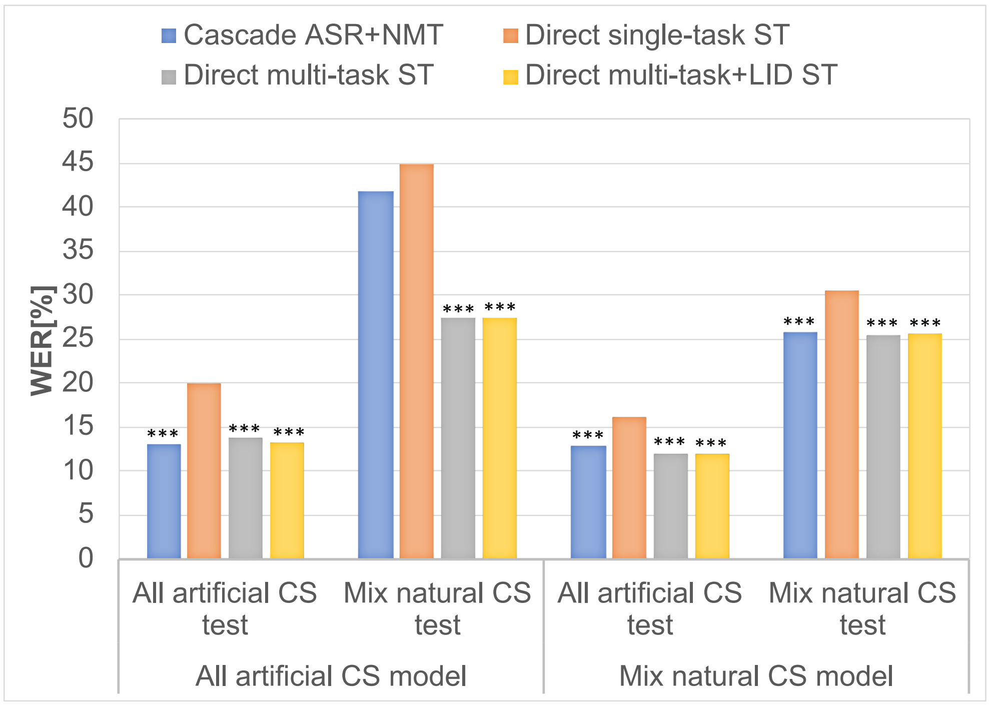


Figure 6.16: WER%↓ of Japanese output in CS speech to Japanese translation in Cascade and Direct approaches. P-values compared to Direct single-task ST are shown with $^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$.

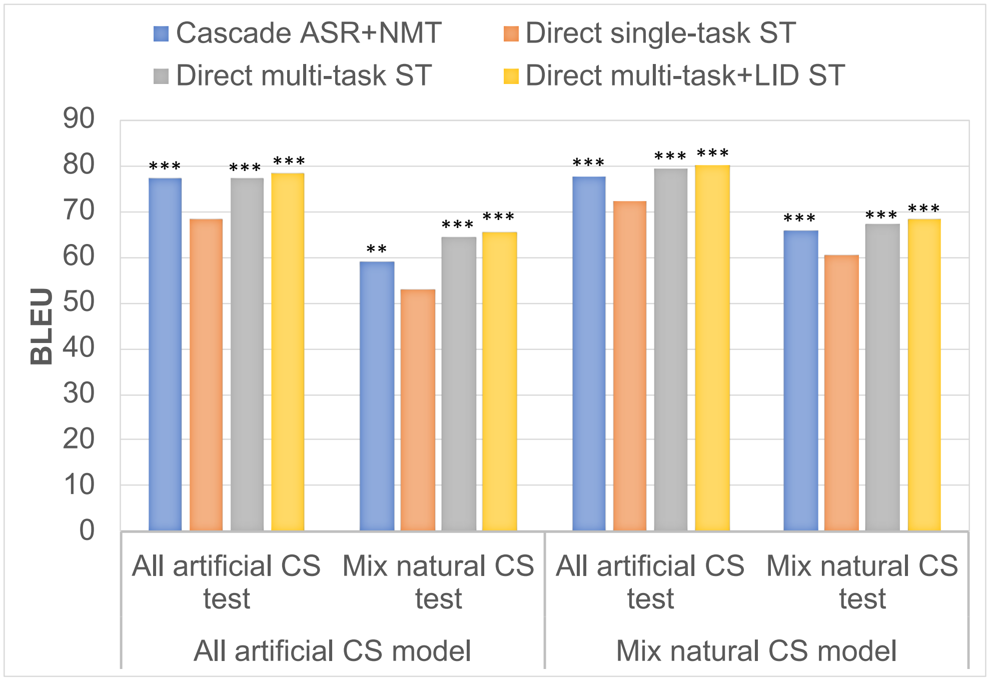


Figure 6.17: BLEU↑ of Japanese output in CS speech to Japanese translation in Cascade and Direct approaches. P-values compared to Direct single-task ST are shown with $^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$.

Fig. 6.16 and Fig. 6.17 show the comparison of translation performance from CS speech to Japanese text between Cascade and Direct approaches in WER and BLEU, respectively. Compared among the Cascade ASR+NMT, Direct single-task ST, Direct multi-task ST, the Direct single-task ST tends to have the worst performance, and the Direct multi-task ST tends to have the best performance. In the same way as the translation toward English, the Direct single-task ST seems to be more difficult than other models since it needs to transcribe directly from CS speech into monolingual English transcriptions. Table 6.10 shows that only single-task ST could not grasp "パン (*bread*)." On the other hand, the Direct multi-task ST could predict well since the CS text prediction of multi-task learning assisted the translation task. Therefore, Direct multi-task ST could predict better than Direct single-task ST.

Table 6.10: Output examples showing that the Direct single-task ST is more difficult in CS to Japanese translation.

(1)	Source text		ゆで 卵 一 個 と orange juice and **bread** please .
	Reference text		ゆで 卵 一 個 と オレンジ ジュース と パン を お願い し ます . (*How would you like the boiled egg?*)
	Output	Cascade ASR+NMT	ゆで 卵 一 個 と オレンジ ジュース と パン を ください .
		Direct single-task ST	ゆで 卵 一 個 と オレンジ ジュース 二つ お願い し ます .
		Direct multi-task ST	ゆで 卵 一 個 と オレンジ ジュース と パン を お願い し ます .
		Direct multi-task+LID ST	ゆで 卵 一 個 と オレンジ ジュース と パン を お願い し ます .

Then the Cascade ASR+NMT degraded the performance, especially on the Mix natural CS test by the All artificial CS model since the Cascade ASR+NMT has ASR error propagation. Table 6.11 shows that the Cascade ASR+NMT outputs wrongly "素材 (*material*)" although it should be "不在 (*be not at home*)." It seems to be error propagation from ASR. On the other hand, the Direct multi-task ST has a fewer ASR error propagation, so it tends to perform better.

Moreover, the Direct multi-task+LID ST generally further improved the performance from the Direct multi-task ST on the BLEU score. However, in the same way as the translation to English, when we first tried the experiments, the Direct multi-task+LID ST was not better than the Direct multi-task ST on the Mix natural CS model, although it was better on the All artificial CS model. Therefore, we first trained the model with only synthetic speech CS data with the LID loss λ_{LID} 0.1, and then fine-tuned the model with synthetic speech CS + natural speech CS with the LID loss λ_{LID} 0.0. As a result, the performance improved a little bit, which result is shown in Fig. 6.16. Although the WER on Mix natural CS test of the Mix natural CS model are not better than the Direct multi-task ST, we expect that it can improve more if we get more natural speech CS. It will be a future task.

Table 6.11: Output examples of the case where Cascade ASR+NMT has ASR error propagation in CS to Japanese translation.

(2)	Source text		あいにく 今 不在 です 六 時 に he said that he would go back .
	Reference text		あいにく 今 不在 です 六 時 に は 戻る と いっ て い まし た . (*He isn't at home. He said that he would go back by six.*)
	Output	ASR	あいにく 今 素材 です 六 時 に he said that he would go back .
		Cascade ASR+NMT	あいにく 今 素材 です 六 時 に 言っ て もらい たい と 思っ た .
		Direct single-task ST	あいにく 今 不在 です 六 時 に 戻っ た そう です
		Direct multi-task ST	あいにく 今 不在 です 六 時 に 戻っ て き て と 言っ て .
		Direct multi-task+LID ST	あいにく 今 不在 です 六 時 に 戻っ て き た と 思い ます .

The additional output examples of the Japanese target are shown in Table 6.12. The sentences where the translation part can work as one sentence such as (3) “please give me two pieces” seem to be easy. The phrases consisting of the combination of words can also be translated well such as Table 6.10 (1) “orange juice and bread please.” On the other hand, the compound words like (4) “24 points five centimeters” seem to be difficult to translate.

Table 6.12: Output examples of speech translation from CS speech to Japanese text on Mix natural CS.

(3)	Source text		七月 二 十 日 二 回 目 公演 の エー 指定 席 券 を please give me two pieces .
	Reference text		七月 二 十 日 二 回 目 公演 の エー 指定 席 券 を 二 枚 ください . (*Can I have two A reserved seats for the second performance on July twentieth?*)
	Output	Cascade ASR+NMT	七月 二 十 日 二 回 目 の エー 指定 席 券 を 二 枚 ください .
		Direct single-task ST	七月 二 十 日 二 回 公園 の エー 席 券 を 二 枚 ください .
		Direct multi-task ST	七月 二 十 日 二 回 目 公演 の エー 指定 席 券 を 二 枚 ください .
		Direct multi-task+LID ST	七月 二 十 日 二 回 目 公演 の エー 指定 席 券 を 二 枚 ください .
(4)	Source text		サイズ の 番号 は わかり ませ ん が the size of the foot is 24 points five centimeters .
	Reference text		サイズ の 番号 は わかり ませ ん が 足 の 大きさ は 二 十 四 点 五 センチ です . (*I don't know the size number, but my foot is 24 points five centimeters.*)
	Output	Cascade ASR+NMT	サイズ の 番号 は わかり ませ ん が 二 十 五 点 六 六 円 です .
		Direct single-task ST	サイズ の 番号 は わかり ませ ん が 五 十 四 センチ の 名前 は .
		Direct multi-task ST	サイズ の 番号 は わかり ませ ん が 残念 ながら 二 十 四 点 五 十 センチ です .
		Direct multi-task+LID ST	サイズ の 番号 は わかり ませ ん が 二 十 センチ の サイズ です .

Results on Multi-pair Code-switching

Fig. 6.18 and Fig. 6.19 show the performance results for translation from multi-pair CS speech to English text in WER and BLEU, respectively. Among a Cascade ASR+NMT, a Direct single-task ST, and a Direct multi-task ST, the Direct multi-task ST tends to have the best performance since it has fewer ASR error propagation and the translation is assisted by multi-task learning of CS text prediction. Moreover, the Direct multi-task ST also has the possibility of more performance improvement with LID. For Mix natural CS model of Direct multi-task+LID ST, we first trained the model with only synthetic speech CS data with the LID loss λ_{LID} 0.1 and then fine-tuned the model with synthetic speech CS + natural speech CS with the LID loss λ_{LID} 0.0. As a result, the performance improved by incorporating LID architecture both on the All artificial CS model and the Mix natural CS model. We expect that it can improve more if we could get more amount of natural CS.

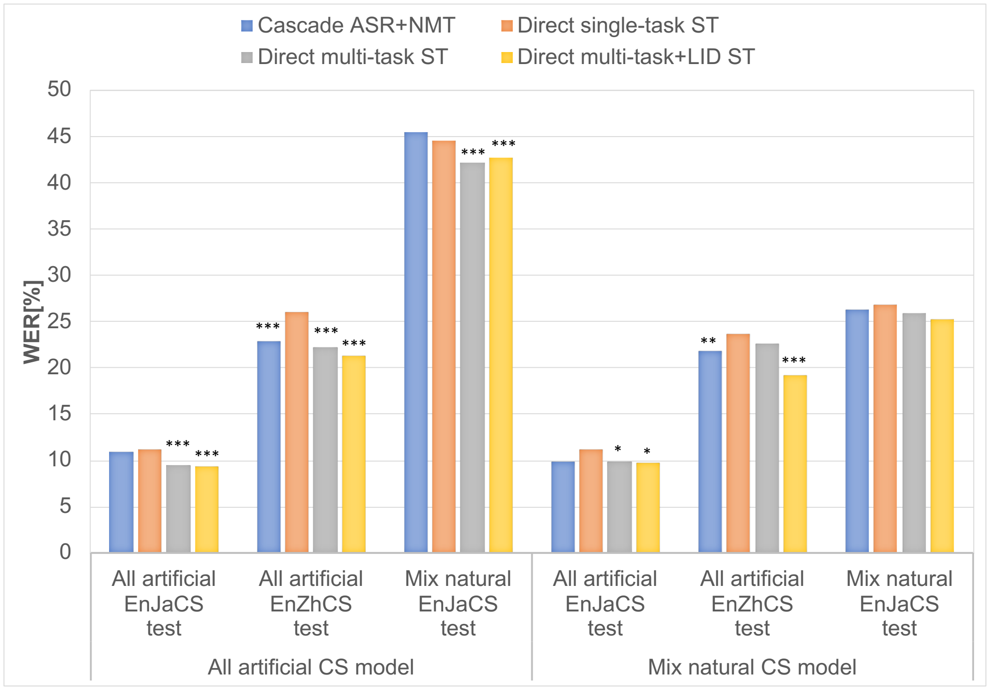


Figure 6.18: WER%↓ of English output in multi-pair CS speech to English translation in Cascade and Direct approaches. P-values compared to Direct single-task ST are shown with $^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$.

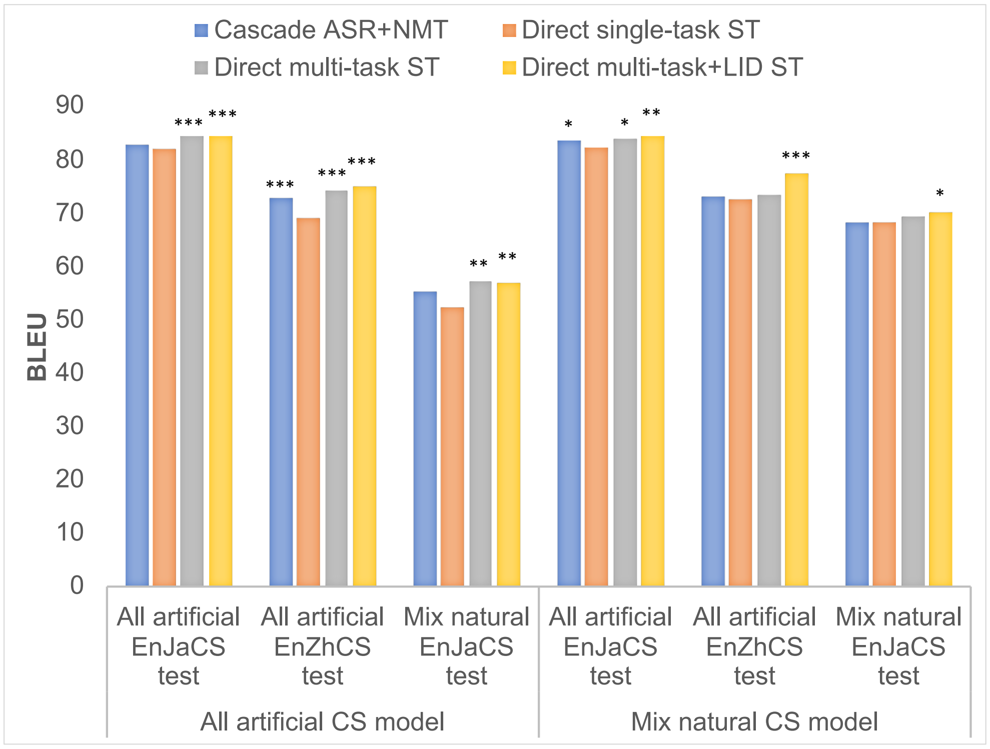


Figure 6.19: BLEU↑ of English output in multi-pair CS speech to English translation in Cascade and Direct approaches. P-values compared to Direct single-task ST are shown with $^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$.

Fig. 6.20 and Fig. 6.21 show the performance results for translation from multi-pair CS speech to Japanese text in WER and BLEU, respectively. As the same with the case of the translation toward English text, among a Cascade ASR+NMT, a Direct single-task ST, and a Direct multi-task ST, the Direct multi-task ST tends to have the best performance. It has fewer ASR error propagation and has translation assistance by multi-task learning of CS text prediction. Moreover, the Direct multi-task ST also has the possibility of more performance improvement with LID. We expect that it can improve more if we could get more amount of natural CS.

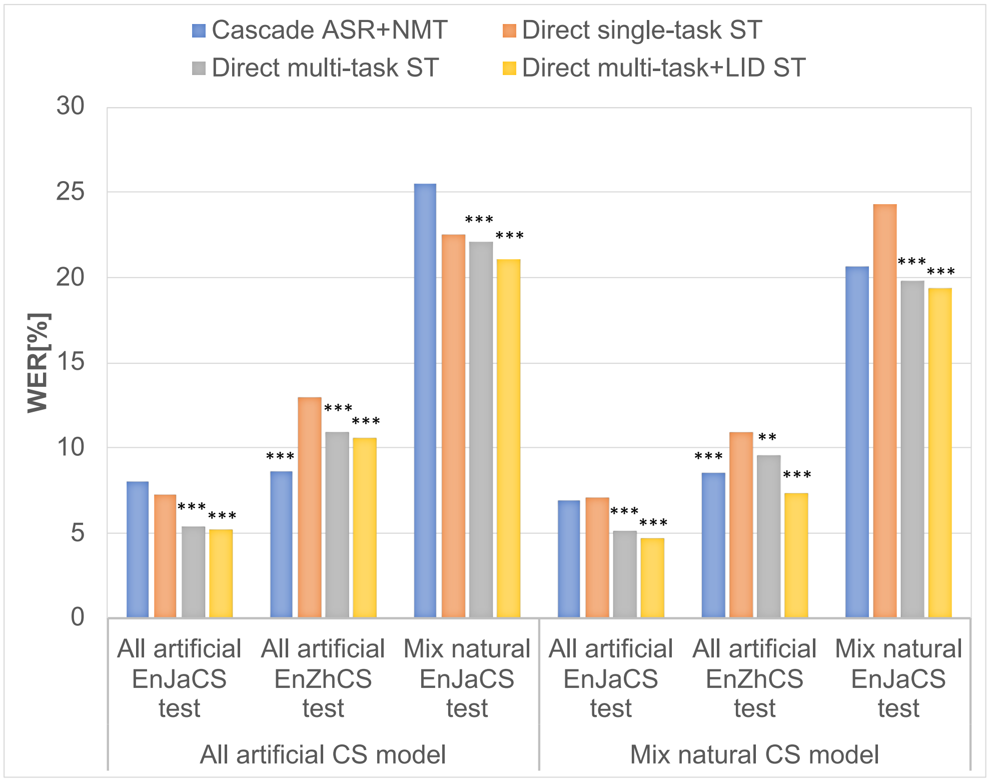


Figure 6.20: WER%↓ of Japanese output in multi-pair CS speech to Japanese translation in Cascade and Direct approaches. P-values compared to Direct single-task ST are shown with $^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$.

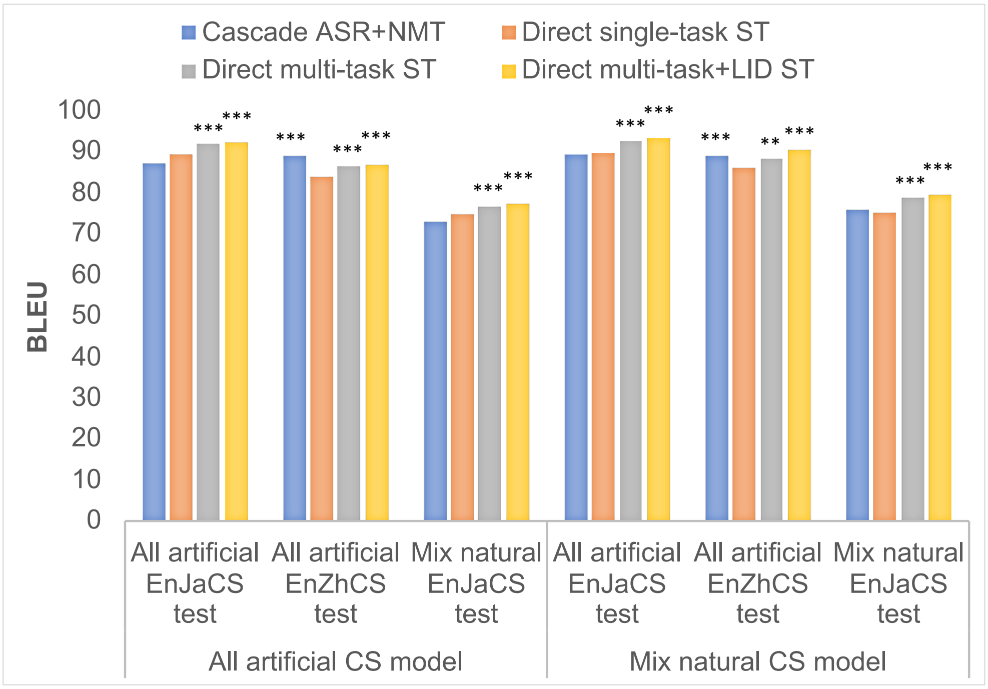


Figure 6.21: BLEU↑ of Japanese output in multi-pair CS speech to Japanese translation in Cascade and Direct approaches. P-values compared to Direct single-task ST are shown with $^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$.

6.5 Summary

This chapter investigated the system that can recognize code-switching speech and translate it into monolingual texts to support monolingual speakers trying to understand CS speakers. The common aim of developing a CS ASR is merely for transcribing CS-speech utterances into CS-text sentences, where we assume only dialogues between the same CS speakers. In contrast, this study addresses the situational context during dialogues between CS and non-CS (monolingual) speakers and supports monolingual speakers who want to understand CS speakers. We investigated several approaches, including a Cascade ASR+NMT, a Cascade ASR+BERT, a Direct single-task ST, and a Direct multi-task ST.

As the result of comparisons between systems, we found that the Direct multi-task ST tended to perform the best. The Cascade ASR+BERT model became too hard as the ASR error increased the number of [MASK] tokens. The Direct single-task ST is a difficult task since it has to transcribe directly from CS speech to monolingual text. The Cascade ASR+NMT has an ASR error propagation. On the other hand, the Direct multi-task ST has a fewer ASR error propagation and CS text prediction helps the translation into monolingual text, so it tends to have the best performance. The Direct multi-task ST improved statistically significant from the most straightforward method single-task ST, about 5% WER improvement on mix natural CS test. Moreover, the Direct multi-task+LID ST showed the possibility of even more improvement by the language information. These findings appeared both on the experiments for translation toward English and Japanese and on single-pair CS and multi-pair CS.

Chapter 7

Conclusion and Future Directions

This last chapter concludes our thesis and discusses future directions for our proposed framework.

7.1 Problem Reiteration

As the bilingual community grows, we have to handle CS on the ASR system to recognize all conversations. However, handling CS is challenging for automatic speech recognition (ASR) because it requires coping with multilingual input. There are many challenges for the CS ASR, but this thesis has focused on the following three problems for the CS ASR's development: language coverage, training mechanism, and usability.

For the problem of language coverage, to support several kinds of bilingual speakers, such as English-Japanese, English-Korean, and Japanese-Chinese, the system for multi-pair CS, which covers multiple pairs of languages, is necessary to recognize all CS conversations. Moreover, the CS proficiency level varies from beginners to near-native speakers. Therefore, developing the system handling non-native CS well is also required.

For the problem of training mechanism, although the primary method for training CS ASR is supervised learning, the datasets of CS speech and the corresponding CS transcriptions are difficult to obtain. Therefore, we need to seek the training mechanism trainable with fewer labeled CS data.

For the problem of usability, the common aim of developing a CS ASR is merely for transcribing CS-speech utterances into CS-text sentences, where we assume only dialogues between the same CS speakers. Therefore, we need to suppose the situational context during conversations between CS and non-CS (monolingual) speakers and support monolingual speakers who want to understand CS speakers.

7.2 Conclusion

This section reviews our work from the perspective of language coverage, training mechanism, and usability.

7.2.1 Language Coverage

For the problem of language coverage, we introduced a language-aware mechanism, enabling handling multi-pair CS better by providing language information. Our proposed language-aware ASR adopts the Direct approach predicting the target text directly, and it predicts language ID sequences together with two softmax layers in a multi-task way. When we compared the Direct ASR (Proposed LID) with other LID approaches, our proposed system significantly improved more than 10% error rate on most of the CS test sets from the traditional Cascade approaches and was better than other Direct approaches. We could confirm the improvements not only on single-pair CS but also on multi-pair CS. For the problem of covering non-native CS, although we handled native CS and non-native CS individually, we have to seek the method for handling non-native CS with various proficiency-level in future work.

7.2.2 Training Mechanism

For the problem of the training mechanism, we utilized the semi-supervised architecture called the machine speech chain. The machine speech chain is the mechanism inspired by the speech chain of the human communication mechanism. It enables ASR and TTS to assist each other when they receive unpaired data (speech or text only) since it allows them to infer the missing pair and optimize the models with reconstruction loss. We applied the machine speech chain framework to the CS task, enabling the CS ASR to develop without any labeled CS data. Our results on single-pair CS (synthetic speech; single speaker) showed that our proposed model improved the ASR performance on the CS test, from 18.1% CER to 5.1%, which reduced the absolute CER by 13.0%. Our proposed model also improved more than 10% error rate from the baseline system on single-pair CS (natural speech; multi-speaker) scenario and about 5% error rate on multi-pair CS scenario. They showed statistically significant improvements with $p < .001$. In addition, we integrated language identification into the ASR and language embedding into the TTS of the CS machine speech chain. We confirmed that the machine speech chain model with language embedding and LID could produce satisfactory performances both on single-pair CS and multi-pair CS.

7.2.3 Usability

For the problem of usability, we investigated the system that recognized and translated CS speech, supporting monolingual speakers trying to understand CS speakers. We explored several approaches, including a cascade of ASR and NMT, a cascade of ASR and BERT, a single-task speech translation, and a multi-task speech translation. As a result, we found that the Direct multi-task ST tends to achieve the best. The Direct single-task ST is difficult since it has to transcribe directly from CS speech to monolingual text. The Cascade ASR+NMT has an ASR error propagation. On the other hand, the Direct multi-task ST has a fewer ASR error propagation, and CS text prediction assists the translation into monolingual text, so it tends to have the best performance. It improved statistically significantly from the most straightforward method single-task ST. The Direct multi-task ST showed the possibility of more performance improvement with LID. We also confirmed the same tendency on single-pair CS and multi-pair CS, where the Direct multi-task ST tended to have the best performance.

7.3 Future Directions

Despite all the contributions listed in Sect. 7.2, there are still several things that our proposed systems cannot do, such as:

- **Performing the CS holding more than two languages within one utterance spoken by a single speaker**
  This thesis assumes that the CS can allow only two languages in one sentence. Therefore, in this research, single-pair CS means one sentence has only two languages and there are only the same language pairs in that data (see Fig. 1.3 (a)). The multi-pair CS means one sentence has only two languages and there are different language pairs in that data (see Fig. 1.3 (b)). Both single-pair CS and multi-pair CS have only two languages in one sentence. However, we can think of a case that includes more than two languages in one sentence. Therefore, tackling the CS with different language pairs of more than two languages (see Fig. 1.3 (c)) will be future work.

  This thesis also allowed the CS switched by multi-speakers within one utterance due to covering low-resource CS data with synthetic CS speech, but the future work will be done only using natural CS speech spoken by a single speaker within one utterance.

- **Performing the completely unknown zero-shot CS**
  Zero-shot learning originally refers to recognizing objects excluded in the training data, where the target is usually unavailable at all. However, in this thesis, our zero-shot CS means the CS language pair is unknown, but one monolingual element of CS may be available as labeled (with speech-text data) or unlabeled data (only speech or only text) like Fig. 7.1 (a). We confirmed that our proposed model could perform the zero-shot CS using **ONE** completely unseen language as Fig. 7.1 (b) shows. However, it has not yet realized the zero-shot CS where **ALL** the comprised languages are unavailable as Fig. 7.1 (c) shows. It will be challenging for the future.

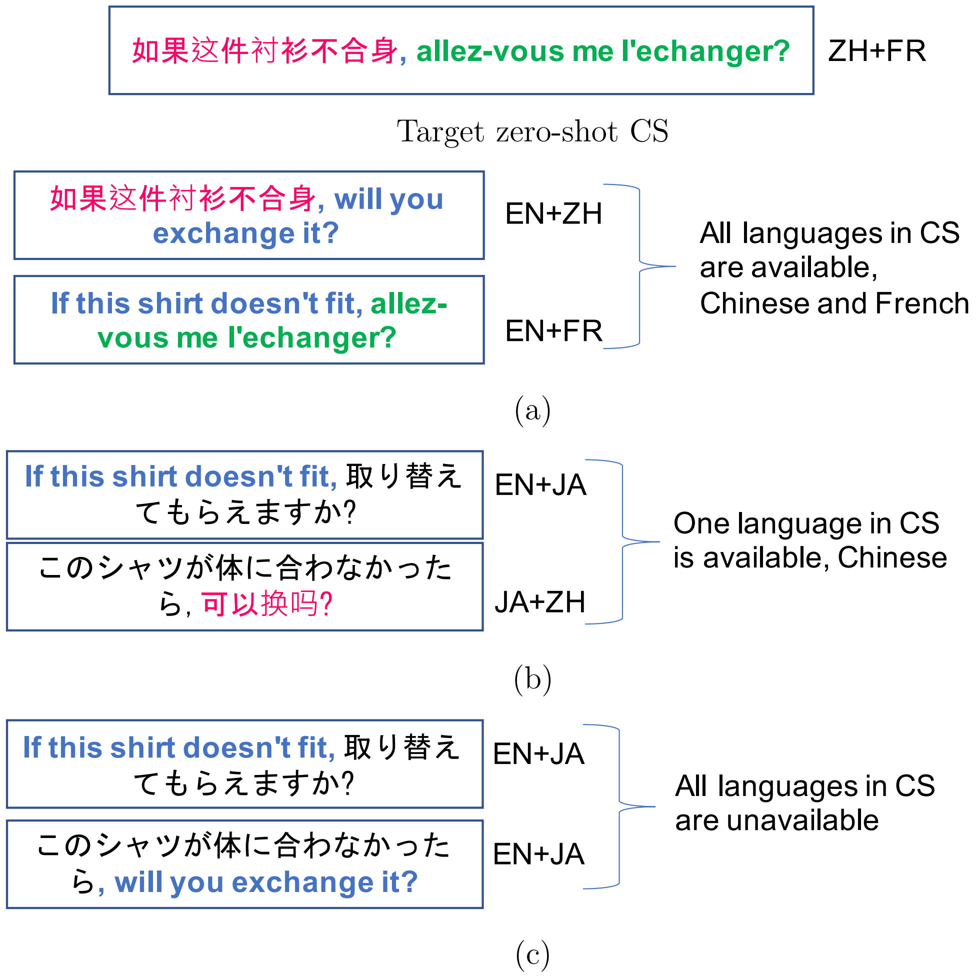

如果这件衬衫不合身, allez-vous me l'echanger?
ZH+FR
Target zero-shot CS
如果这件衬衫不合身, will you exchange it?
EN+ZH
If this shirt doesn't fit, allez-vous me l'echanger?
EN+FR
All languages in CS are available, Chinese and French
(a)
If this shirt doesn't fit, 取り替えてもらえますか?
EN+JA
このシャツが体に合わなかったら, 可以换吗?
JA+ZH
One language in CS is available, Chinese
(b)
If this shirt doesn't fit, 取り替えてもらえますか?
EN+JA
このシャツが体に合わなかったら, will you exchange it?
EN+JA
All languages in CS are unavailable
(c)

Figure 7.1: Zero-shot situations, where (a) all languages in CS are available, (b) one language in CS is available, and (c) all languages in CS are unavailable. We challenged (a) and (b), but (c) is future work.

- **Performing natural CS data as a zero-shot target**
  We challenged the zero-shot CS in the easier situation using intra-sentential phrase-level CS II. The CS's switching points are a comma, and the CS has parallel translation settings between language pairs of CS. For example, "If this shirt doesn't fit, 取り替えてもらえますか" and "If this shirt doesn't fit, allez-vous me l'echanger?" are parallel translation settings in the meaning of "*If this shirt doesn't fit, will you exchange it?*." It can be easier

for zero-shot CS since they can share the monolingual beyond particular language pairs of CS. Although we also conducted the experiments using natural speech CS data, we could not target the natural speech CS data as zero-shot CS. Therefore, challenging the zero-shot CS of natural CS data will be future work.

- **CS translation providing multiple monolingual outputs**
  Normally, the system should provide two or more monolingual outputs using a single system since the CS is composed of multiple monolinguals. However, in this thesis, the system only provides one monolingual output. The target speaker is fixed in English or Japanese, and the system cannot handle multiple monolingual users. We still need to build multiple systems to handle multiple monolingual users. Therefore, future work will handle multiple monolingual users by a single system.

- **Satisfactory quality**
  Although the performance has improved from the baseline through all experiments, the overall quality may not be satisfied. For example, the performance of ASR in single-pair CS (natural speech; multi-speaker) is still around 30-40% TER. With the speech chain for semi-supervised learning, the performance of ASR in single-pair CS (natural speech; multi-speaker) is still around 30-50% TER. Therefore, we have to seek performance improvement in future work.

Our system has the above issues, but there are several solutions that we can consider. They are as follows:

- **To challenge the CS including three or more languages within one utterance spoken by a single speaker**
  To tackle the CS that includes three or more languages within one utterance spoken by a single speaker, we first have to collect the data since we have not found the corpus handling such CS yet. Based on the data collection method of SEAME corpus [22], we have to design the trigger question of CS. Even if participants are multilingual speakers, they may answer only in Japanese if we ask a question only in Japanese. Therefore, we will design the question

to trigger the CS that includes three or more languages. After collecting the CS, we will try to perform it on our proposed model, incorporating the language information.

- **To challenge the completely unknown zero-shot CS**
  Challenging the completely unknown zero-shot CS will be a considerably difficult task, but we can investigate the zero-shot cross-lingual approaches. Based on the zero-shot cross-lingual approach [87], wav2vec-based feature extraction [88], phoneme inventory masking [89], and the language encoder offering the similarity with other languages, can be helpful to recognize the unseen languages. Therefore, we will incorporate those architectures into our system for totally unknown zero-shot CS.

- **To challenge the zero-shot CS of natural CS data**
  The current system may be challenging to perform the zero-shot CS of natural CS data. Therefore, as with the previous paragraph, we incorporate the wav2vec-based feature extraction, the phoneme inventory masking, and the language encoder, based on the zero-shot cross-lingual approach [87]. By combining them, we expect to perform the zero-shot CS of natural CS data well.

- **To realize the CS translation providing multiple monolingual outputs**
  In a previous experiment, the CS starting from Japanese could not translate to monolingual English well, and the CS starting from English could not translate to monolingual Japanese well. For that reason, we could not implement the single system translating multiple monolingual outputs. The performance degradation may be caused by the monotonic switching rule in the synthetic CS used for training. The result may change if I train with many natural CS data holding various switching rules. In this work, I have not investigated it yet since I could not obtain a lot of natural CS data for the translation tasks, but it would be future work. I also have a plan to adopt the pre-ordering models. The pre-ordering models rearrange the word order of the source language to the word order of the target language beforehand. By incorporating the pre-ordering models, we expect the sys-

tem to translate to multiple monolingual outputs without being unaffected by the source language's word order.

- **To improve the performance**
  To improve the performance, we can try some approaches. For example, data augmentation approaches increasing the amount of training data can effectively improve performance. Since we have already applied the speed perturbation [82, 83], we can try another data augmentation approaches such as SpecAugment [90]. We can also consider using more powerful models such as Transformer [76] or Conformer (Convolution-augmented Transformer) [91] instead of an attention-based encoder-decoder model. Using a language model can also be helpful. To improve the performance of the non-native CS, we can also consider incorporating accent embedding and accent classification into the architectures. We will later describe the detail about the accent embedding and accent classification architecture when we describe the future directions for the non-native CS ASR.

We also still have several things to do for the extension researches of CS ASR. Fig. 7.2 shows the research roadmap towards future works. From the viewpoint of the language coverage, we have already investigated the CS ASR on single-pair CS and multi-pair CS. We have also investigated the system to translate to monolingual for supporting monolingual listeners both from the speech of single-pair CS and multi-pair CS speakers. Therefore, in future works, we will challenge to develop the CS ASR for non-native CS (single/multi-pair languages) speakers and listeners.

In terms of data necessity, we have already done the CS ASR with paired CS data (speech+text) using supervised learning and with unpaired CS data (speech/text only) using semi-supervised learning. However, achieving without any CS data (monolingual speech+text) using supervised/semi-supervised learning will be future work.

We have another remaining task. We conducted experiments both with synthetic and natural speech. However, some experiments used the mixed data of the natural speech and the synthetic speech since we could not have enough natural speech data. Therefore, improving the performance enough only on natural speech will also be future work.

Accordingly, the potential future works are the following tasks: (1)To develop the CS ASR for non-native CS; (2)To realize the CS ASR without any CS data using only monolingual data; and (3)To verify the performance using natural CS speech data. We describe the detail of them one by one.

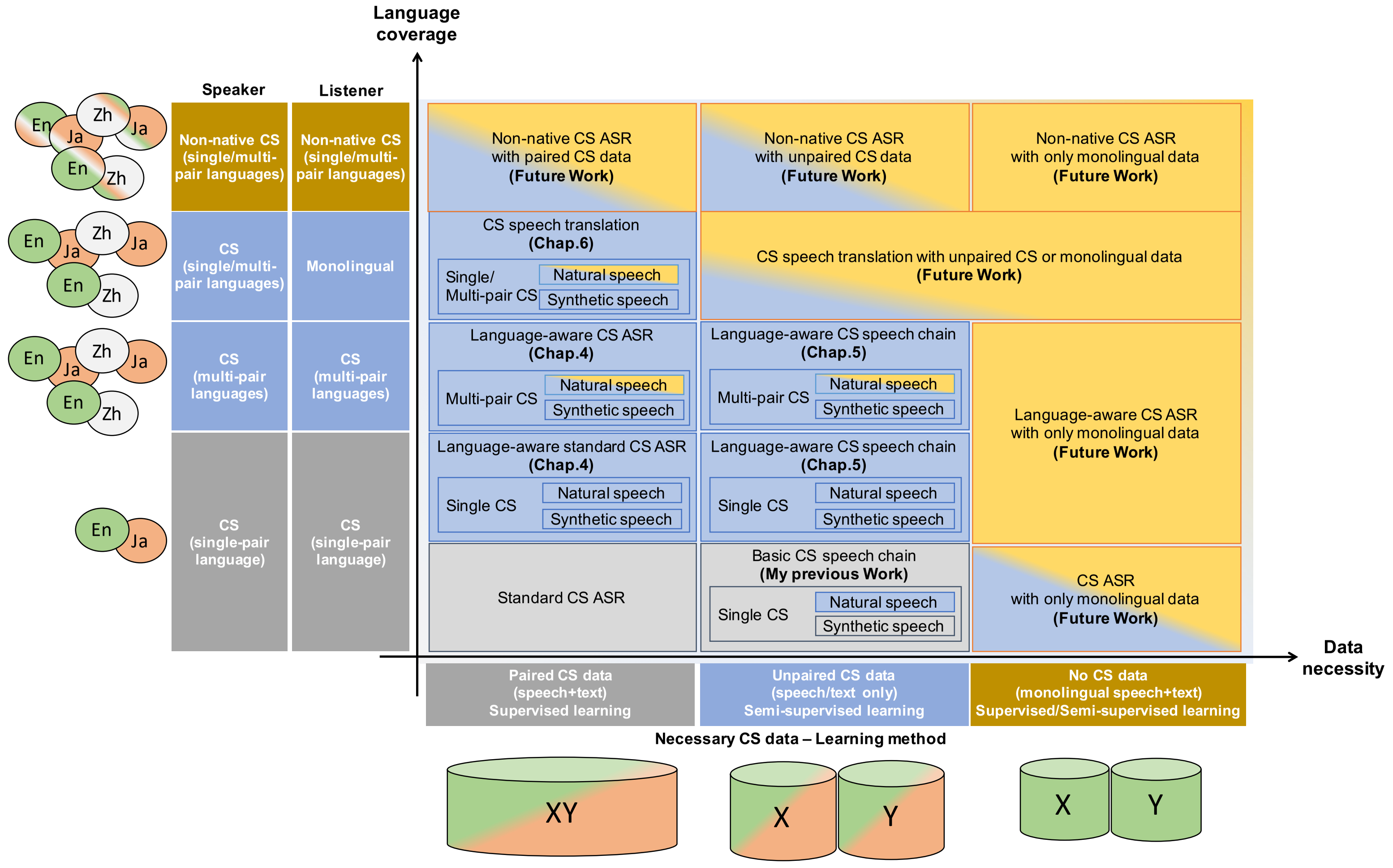

Language coverage
Speaker
Listener
Non-native CS (single/multi-pair languages)
Non-native CS (single/multi-pair languages)
CS (single/multi-pair languages)
Monolingual
CS (multi-pair languages)
CS (multi-pair languages)
CS (single-pair language)
CS (single-pair language)
En
Ja
Zh
Non-native CS ASR with paired CS data (Future Work)
Non-native CS ASR with unpaired CS data (Future Work)
Non-native CS ASR with only monolingual data (Future Work)
CS speech translation (Chap.6)
Single/ Multi-pair CS
Natural speech
Synthetic speech
CS speech translation with unpaired CS or monolingual data (Future Work)
Language-aware CS ASR (Chap.4)
Multi-pair CS
Language-aware CS speech chain (Chap.5)
Language-aware CS ASR with only monolingual data (Future Work)
Language-aware standard CS ASR (Chap.4)
Single CS
Standard CS ASR
Basic CS speech chain (My previous Work)
CS ASR with only monolingual data (Future Work)
Data necessity
Paired CS data (speech+text) Supervised learning
Unpaired CS data (speech/text only) Semi-supervised learning
No CS data (monolingual speech+text) Supervised/Semi-supervised learning
Necessary CS data – Learning method
XY
X
Y

Figure 7.2: Research roadmap toward future works. The grey area shows the previous researches. The blue area shows the topics handled in this thesis. The yellow area shows the remaining tasks to be achieved in future works.

Developing the CS ASR for non-native CS speakers and non-native CS listeners

As we described in Sect. 1.2.2, there are native CS and non-native CS in CS. In this thesis, we handled separately native CS (synthetic speech and our collected CS utterances) and non-native CS (SEAME corpus [22]) but have not yet handled them together. Handling them together is difficult for ASR since it causes a mismatch between the input speech and trained model [9].

As a solution, based on the previous research [92], we can try the approach using accent embedding, and multi-task with accent recognition as Fig. 7.3 shows. We first train the accent recognition model and generate an accent vector in advance. Then in the ASR model, we embed the accent vector with speech features together. In the decoder of the ASR model, we conduct multi-task training of character recognition and accent recognition. As a result, we expect it can handle native CS and non-native CS well.

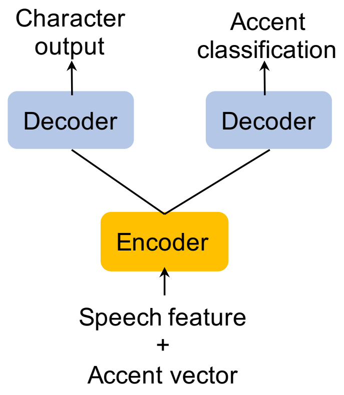


Figure 7.3: A multi-task model with accent classification and accent embedding. It embeds speech feature and accent vector, proceeds those concatenated vectors, then predicts character sequences and accent classification with multi-task learning.

Realizing the CS ASR without any CS data (only monolingual data)

We have investigated the semi-supervised learning approach regarding the CS data problem, but it still uses CS data. We need to investigate the CS ASR with only monolingual data, which is more available than CS data. Although Chap. 4

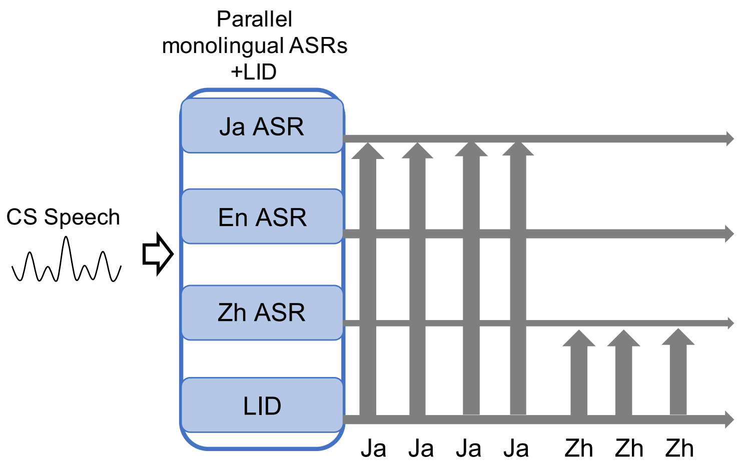


Figure 7.4: Parallel monolingual ASRs and LID. It decodes monolingual ASRs and LID at the same time. After decoding, we choose a sequence based on the LID output.

shows the CS ASR trained with only monolingual data (Ja25k+En25k+Zh25k), we can consider a more efficient approach. The approach uses parallel monolingual ASRs and LID, as Fig. 7.4 shows. The idea comes from the related works [45,93]. It is also similar to the Cascade approach we introduced in Sect. 4.3, but they are different. The Cascade approach has non-recoverable damage from LID error, but it does not have such damage since it conducts ASRs and LID simultaneously. We first train monolingual ASRs, where we also train them to predict the start frame and end frame for each word to get alignment. We also train the LID system with multilingual data. Then we decode the CS speech using the monolingual ASRs and LID. After decoding, we choose a sequence based on the LID output. Finally, we can realize the CS ASR using the model trained with only monolingual data. We will also realize it with semi-supervised learning, utilizing a machine speech chain.

Verification of the performance using natural CS speech data

We validated the performance using natural CS speech data with SEAME corpus [22] for a single CS. However, we have not yet validated the performance using natural CS speech for multi-pair CS and CS speech translation since we have not obtained enough natural speech data for those tasks. So far, for those tasks, we used the natural speech CS created by bilinguals, but it does not have enough, so we used it by mixing it with synthetic data. Therefore, the data collection of natural CS speech for multi-pair (zero-shot) CS and CS speech translation and the performance verification using natural CS speech data will be our future task.

Appendix

A Detailed Results on Code-switching Speech Translation

In Chap. 6, we investigated the code-switching speech translation. In this appendix, we show the detailed results. The column of "Source text against reference text" shows the WER, CER, and BLEU of source text against reference text, which means how close the source text is to the reference text. We compare the score of the translation results with those values since it means to be closer to the reference by the translation if the WER and CER get lower or BLEU get higher. However, these comparisons can fluctuate according to the training data size, so we focused on comparing the systems. We also calculate the score of the "CS part" and "non-CS part." The "CS part" is the different language's part from the target language in the source CS text, and the "non-CS part" is the same language's part from the target language in the source CS text. We confirm how well it can translate the CS part and the non-CS part, respectively.

Results on Single-pair Code-switching

Code-switching Text to Monolingual Text

As we see Table A.1, the "CS part" WER and CER in the "Source text against reference text" were over 100%. This is because the number of insertion, substitution, and deletion to source "CS part" was more than the length of reference "CS part."

The BERT language model has lower WER and CER and higher BLEU than "Source text against reference text," so it became closer to the reference text by the performance. Similarly, the NMT model has lower WER and CER and higher BLEU than "Source text against reference text" excepting the Mix natural CS test on the All artificial CS model, so it became closer to the reference text by the performance. Compared between BERT and NMT, the NMT model tends to be better both on "All" and "CS part." Only the Mix natural CS "All" evaluation on the All artificial CS, the BERT model has higher performance than the NMT model. However, the NMT model is better than the BERT model on “CS part,” which means that the BERT “All” performance is good by “non-CS part” performance. The BERT “non-CS part” is copied from the input text, so the performance can be good if the input text does not include an error from ASR.

Table A.1: BERT and NMT translation performance from CS text to English text.

Test			Source text against reference text	Model: BERT, All artificial CS	Model: BERT, Mix natural CS
All artificial CS	WER%↓	All	27.43	11.14	12.14
		CS part	179.52	60.11	66.40
		non-CS part	0.11	5.17	5.20
	CER%↓	All	19.79	12.01	12.62
		CS part	106.66	61.98	65.29
		non-CS part	0.07	1.03	1.09
	BLEU↑	All	66.36	78.46	79.42
Mix natural CS	WER%↓	All	35.23	20.61	19.53
		CS part	164.96	83.06	72.02
		non-CS part	3.44	8.67	8.77
	CER%↓	All	24.80	19.31	18.56
		CS part	104.94	74.72	68.44
		non-CS part	2.54	3.69	3.92
	BLEU↑	All	61.85	72.03	73.11

Test			Source text against reference text	Model: NMT, All artificial CS	Model: NMT, Mix natural CS
All artificial CS	WER%↓	All	27.43	7.47	6.56
		CS part	179.52	34.31	35.11
		non-CS part	0.11	2.64	1.43
	CER%↓	All	19.79	8.87	7.94
		CS part	106.66	32.92	33.35
		non-CS part	0.07	3.39	2.08
	BLEU↑	All	66.36	86.54	88.11
Mix natural CS	WER%↓	All	35.23	37.56	17.94
		CS part	164.96	60.69	57.88
		non-CS part	3.44	28.18	6.64
	CER%↓	All	24.80	28.87	17.72
		CS part	104.94	50.59	49.96
		non-CS part	2.54	22.37	7.16
	BLEU↑	All	61.85	57.46	77.86

Table A.2: Comparison between cascade ASR+BERT and cascade ASR+NMT from CS speech to English text.

Test			Source text against reference text	Model			
				All artificial CS		Mix natural CS	
				Cascade		Cascade	
				ASR+ BERT	ASR+ NMT	ASR+ BERT	ASR+ NMT
All artificial CS	WER%↓	All	27.43	16.16	10.76	15.80	9.20
		CS part	179.52	66.76	36.08	69.15	34.13
		non-CS part	0.11	8.74	6.16	8.52	5.37
	CER%↓	All	19.79	15.83	9.73	14.97	8.92
		CS part	106.66	65.74	33.39	67.11	31.86
		non-CS part	0.07	4.76	4.22	3.65	3.31
	BLEU↑	All	66.36	71.48	80.79	73.17	82.08
Mix natural CS	WER%↓	All	35.23	46.42	41.94	25.61	22.34
		CS part	164.96	89.67	56.50	74.23	57.15
		non-CS part	3.44	32.61	32.73	14.76	13.17
	CER%↓	All	24.80	36.95	31.56	22.46	20.23
		CS part	104.94	77.12	49.97	69.91	49.26
		non-CS part	2.54	24.36	24.35	8.85	10.28
	BLEU↑	All	61.85	47.89	54.43	64.24	69.82

Code-switching Speech to Monolingual Text

Table A.2 shows the detailed result of the speech translation with the Cascade ASR+BERT and the Cascade ASR+NMT from CS speech. First, when we compare those models' performance with "Source text against reference text," although the Mix natural CS test on the All artificial CS model was hard, the other cases improved the WER, CER, and BLUE. Compared between the Cascade ASR+BERT and the Cascade ASR+NMT, the Cascade ASR+NMT is better than the Cascade ASR+BERT in all cases. Even if we see the "CS part" performance, the Cascade ASR+NMT is better than the Cascade ASR+BERT in all cases. It seems the task became too hard for BERT, as the ASR error increased the number of [MASK] tokens.

Table A.3 shows the comparison of translation performance from CS speech to English text between cascade and direct approaches. First, when we compare those models' performance with "Source text against reference text," except for the Mix natural CS test on the All artificial CS model, the "All" and "CS part" of almost all models improved the score of WER, CER, and BLUE. Although the CER of Direct single-task ST on the Mix natural CS test of the Mix natural CS model is higher than the "Source text against reference text," the WER is lower. Compared among the Cascade ASR+NMT, the Direct single-task ST, and the Direct multi-task ST, the Direct multi-task ST tends to perform the best. The Direct single-task ST seems to be more difficult than other models since it needs to transcribe directly from CS speech into monolingual English transcriptions. The Direct multi-task ST is better on the "CS part," even where "All" WER and CER of the Cascade ASR+NMT are better than the Direct multi-task ST on the Mix natural CS test of the Mix natural CS model. Therefore, the Direct multi-task ST tends the best performance. The Direct multi-task+LID ST generally showed even more improvement of the performance.

Table A.3: CS speech translation performances toward English text. P-values compared to Direct single-task ST are shown with $^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$.

Test			Source text against reference text	Model: All artificial CS — Cascade ASR+ NMT	Direct single-task ST	Direct multi-task ST	Direct multi-task +LID ST
All artificial CS	WER%↓	All	27.43	10.76	11.13	10.15	9.63*
		CS part	179.52	36.08	40.96	34.40	33.87
		non-CS part	0.11	6.16	5.73	5.73	5.24
	CER%↓	All	19.79	9.73	9.69	8.85*	9.08
		CS part	106.66	33.39	34.52	31.14	32.14
		non-CS part	0.07	4.22	3.88	3.71	3.67
	BLEU↑	All	66.36	80.79	80.82	81.99	83.02*
Mix natural CS	WER%↓	All	35.23	41.94	38.87	34.56	30.35**
		CS part	164.96	56.50	60.69	55.85	52.38*
		non-CS part	3.44	32.73	29.77	26.16	23.80
	CER%↓	All	24.80	31.56	29.42	28.26	29.67
		CS part	104.94	49.97	50.59	48.91	50.96
		non-CS part	2.54	24.35	23.08	21.66	22.96
	BLEU↑	All	61.85	54.43	56.72	58.28	61.68**

Test			Source text against reference text	Model: Mix Natural CS — Cascade ASR+ NMT	Direct single-task ST	Direct multi-task ST	Direct multi-task +LID ST
All artificial CS	WER%↓	All	27.43	9.20	13.04	8.71*	8.68*
		CS part	179.52	34.13*	37.59	33.16**	33.25**
		non-CS part	0.11	5.37	8.60	4.95*	4.94*
	CER%↓	All	19.79	8.92***	10.83	8.50***	8.36***
		CS part	106.66	31.86*	34.43	30.60**	31.62**
		non-CS part	0.07	3.31	5.16	2.99*	3.20*
	BLEU↑	All	66.36	82.08*	78.67	82.87*	83.38*
Mix natural CS	WER%↓	All	35.23	22.34***	29.63	23.21**	23.28**
		CS part	164.96	57.15	61.20	56.79*	56.65*
		non-CS part	3.44	13.17***	19.79	14.78*	14.92*
	CER%↓	All	24.80	20.23***	24.98	21.55***	21.44***
		CS part	104.94	49.26	53.43	49.05*	51.71
		non-CS part	2.54	10.28***	16.00	12.54*	12.19*
	BLEU↑	All	61.85	69.82*	64.38	68.46*	69.70*

Table A.4 shows that the translated and CS output examples of Direct multi-task ST. The translated and CS output seem to be almost the same in English transcriptions, though example (2) is slightly different.

Table A.4: Translated and CS output examples of Direct multi-task ST.

(1)	Source text		if you want to watch tv , you should 宿題 は 終え た ほう が いい です よ .
	Reference text		if you want to watch tv , you should finish your homework first .
	Direct multi-task ST	Translated output	if you want to watch tv , you should have any homework .
		CS output	if you want to watch tv , you should 宿題 は 壊れ た ほう が いい です よ .
(2)	Source text		i ' ll do my best to find your baggage but first i ' d like you to fill in this 手 荷物 事故 報告 書 .
	Reference text		i ' ll do my best to find your baggage , but first i ' d like you to fill in this property irregularity report .
	Direct multi-task ST	Translated output	i ' ll do my best to find your baggage , but first i like you to fill in this form to yourself in japan .
		CS output	i ' ll do my best to find your baggage but first i ' d like you to fill it this place も 一 時 も 報告 書 .
(3)	Source text		oh , no . i don ' t have any change so , どう し たら いいん でしょ う ?
	Reference text		oh , no . i don ' t have any change so , what should i do ?
	Direct multi-task ST	Translated output	oh , no . i don ' t have any change so , what should i do ?
		CS output	oh , no . i don ' t have any change . どう し たら いいん でしょ う ?
(4)	Source text		how would you like to pay 現金 です か , カード です か ?
	Reference text		how would you like to pay , cash or charge ?
	Direct multi-task ST	Translated output	how would you like to pay , cash or charge ?
		CS output	how would you like to pay 現金 です か , カード です か ?

Table A.5 shows the comparison of translation performance from CS speech to Japanese text between cascade and direct approaches. The WER and CER of “All” and “CS part” improved from the score of “Source text against reference text” in almost all test cases. However, the Cascade ASR+NMT and Direct single-task ST have a higher WER and CER than the score of “Source text against reference text” in the “All” of the Mix natural CS test on the All artificial CS model. It seems to be because the untrained Mix natural CS test was difficult. The Direct single-task ST has a higher WER than the “Source text against reference text” in the Mix natural CS test on the Mix natural CS model. It transcribes both English and Japanese speech into the same Japanese transcriptions without distinction, so the model performance degraded and increased the error. The Direct single-task ST also has a lower BLEU than the “Source text against reference text” in the Mix natural CS test on the All artificial CS model and the Mix natural CS model for the same reason. Compared among the Cascade ASR+NMT, Direct single-task ST, Direct multi-task ST, the Direct single-task ST tends to have the worst performance, and the Direct multi-task ST tends to have the best performance. However, the Cascade ASR+NMT tends to be better in the “All” and “CS part” of the All artificial CS test on the All artificial CS model. It seems to be because it has a fewer ASR error propagation since it is a comparatively easier task owing to that the JaEnCS includes one word CS with the only noun. In the WER of the Mix natural CS test on the Mix natural CS model, the Direct multi-task ST is better on the “All.” Although the Cascade ASR+NMT is better on the “CS part” in WER, it is a slight difference due to the difference of the tokenization since the Direct multi-task ST is better in the case of CER. In addition to that, the Direct multi-task ST has the best performance both on the “All” and “CS part” in the Mix natural CS test on the All artificial CS model and the All artificial CS test on the Mix natural CS model. The Direct multi-task+LID ST generally further improved the performance from the Direct multi-task ST on the BLEU score.

Table A.5: CS speech translation performances toward Japanese text. P-values compared to Direct single-task ST are shown with $^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$.

			Source text against reference text	Model			
				All artificial CS			
Test				Cascade ASR+NMT	Direct single-task ST	Direct multi-task ST	Direct multi-task +LID ST
All artificial CS	WER%↓	All	17.52	13.11***	19.96	13.88***	13.33***
		CS part	106.42	32.59***	42.25	36.40**	35.01***
		non-CS part	0.61	9.19***	15.51	9.44***	9.14***
	CER%↓	All	35.23	12.10***	18.93	13.10***	13.23***
		CS part	187.66	27.32***	36.05	30.85***	30.91***
		non-CS part	0.53	9.05***	15.55	9.40***	9.65***
	BLEU↑	All	74.14	77.42***	68.14	77.35***	78.19***
Mix natural CS	WER%↓	All	28.67	41.80	44.81	27.51***	27.48***
		CS part	104.56	61.46***	68.67	61.18***	62.13***
		non-CS part	3.66	25.78***	33.85	16.31***	16.37***
	CER%↓	All	54.29	33.97**	46.70	25.69***	27.71***
		CS part	202.69	56.87***	66.90	54.98***	61.43***
		non-CS part	1.90	20.85***	35.65	15.95***	17.40***
	BLEU↑	All	64.64	59.02**	52.81	64.47***	65.48***

			Source text against reference text	Model			
				Mix natural CS			
Test				Cascade ASR+NMT	Direct single-task ST	Direct multi-task ST	Direct multi-task +LID ST
All artificial CS	WER%↓	All	17.52	12.94***	16.25	12.05***	11.89***
		CS part	106.42	33.48*	36.28	31.96***	33.16*
		non-CS part	0.61	8.90***	12.29	8.18***	7.78***
	CER%↓	All	35.23	12.23***	15.38	11.68***	11.59***
		CS part	187.66	28.59*	31.10	26.92***	28.71*
		non-CS part	0.53	8.86***	12.34	8.41***	8.20***
	BLEU↑	All	74.14	77.77***	72.31	79.36***	80.18***
Mix natural CS	WER%↓	All	28.67	25.77***	30.51	25.52***	25.64***
		CS part	104.56	60.40**	63.44	61.67	61.75
		non-CS part	3.66	14.41***	19.04	13.46***	13.39***
	CER%↓	All	54.29	23.85***	28.04	22.98***	23.45***
		CS part	202.69	53.88**	57.72	53.43	54.54
		non-CS part	1.90	14.29***	18.91	13.18***	13.53***
	BLEU↑	All	64.64	65.75***	60.32	67.24***	68.38***

Results on Multi-pair Code-switching

Table A.6 shows the performance results for translation from multi-pair CS speech to English text. As it shows, among a cascade of ASR and NMT, a Direct single-task ST, and a Direct multi-task ST, the Direct multi-task ST tends to have the best performance. Moreover, the Direct multi-task ST also has the possibility of more performance improvement with LID.

Table A.7 shows the performance results for translation from multi-pair CS speech to Japanese text. As the same with the case of the translation toward English text, among a cascade of ASR and NMT, a Direct single-task ST, and a Direct multi-task ST, the Direct multi-task ST tends to have the best performance. Moreover, the Direct multi-task ST also has the possibility of more performance improvement with LID.

Table A.6: Comparison of translation performance from multi-pair CS speech to English text between cascade and direct approaches. P-values compared to Direct single-task ST are shown with $^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$.

Test			Source text against reference text	Model: All artificial CS — Cascade ASR+ NMT	Direct single-task ST	Direct multi-task ST	Direct multi-task +LID ST
All artificial CS	WER%↓	EnJaCS	27.43	10.95	11.20	9.49***	9.34***
		EnZhCS	50.88	22.91***	26.06	22.23***	21.31***
	CER%↓	EnJaCS	19.79	9.04	9.17	7.88***	7.85***
		EnZhCS	44.92	19.90***	22.86	19.34***	18.17***
	BLEU↑	EnJaCS	66.36	82.75	81.89	84.43***	84.53***
		EnZhCS	49.70	72.85***	68.97	74.16***	75.08***
Mix natural CS	WER%↓	EnJaCS	35.23	45.43	44.58	42.22***	42.71***
	CER%↓	EnJaCS	24.80	33.51	35.71	30.26***	30.05***
	BLEU↑	EnJaCS	61.85	55.26	52.47	57.21**	57.06**

Test			Source text against reference text	Model: Mix natural CS — Cascade ASR+ NMT	Direct single-task ST	Direct multi-task ST	Direct multi-task +LID ST
All artificial CS	WER%↓	EnJaCS	27.43	9.87	11.19	9.85*	9.79*
		EnZhCS	50.88	21.78**	23.65	22.58	19.14***
	CER%↓	EnJaCS	19.79	8.18*	8.87	8.19*	8.01**
		EnZhCS	44.92	19.12	20.04	19.65	16.33***
	BLEU↑	EnJaCS	66.36	83.60*	82.23	84.00*	84.30**
		EnZhCS	49.70	73.07	72.51	73.36	77.33***
Mix natural CS	WER%↓	EnJaCS	35.23	26.28	26.79	25.90	25.27
	CER%↓	EnJaCS	24.80	20.88	21.03	21.35	20.81
	BLEU↑	EnJaCS	61.85	68.30	68.36	69.22	70.23*

Table A.7: Comparison of translation performance from multi-pair CS speech to Japanese text between cascade and direct approaches. P-values compared to Direct single-task ST are shown with $^{***}p < .001$, $^{**}p < .01$, $^{*}p < .05$.

Test			Source text against reference text	Model: All artificial CS — Cascade ASR+ NMT	Direct single-task ST	Direct multi-task ST	Direct multi-task +LID ST
All artificial CS	WER%↓	JaEnCS	17.52	8.02	7.28	5.37***	5.23***
		JaZhCS	56.82	8.63***	12.96	10.90***	10.63***
	CER%↓	JaEnCS	35.23	7.27	6.76	4.99***	5.00***
		JaZhCS	48.98	8.00***	11.74	9.70*	9.58**
	BLEU↑	JaEnCS	74.14	87.20	89.30	91.82***	92.36***
		JaZhCS	46.71	89.05***	84.02	86.46***	86.73***
Mix natural CS	WER%↓	JaEnCS	28.67	25.47	22.52	22.08	21.05***
	CER%↓	JaEnCS	54.29	24.25	21.51	20.67	20.15***
	BLEU↑	JaEnCS	64.64	72.91	74.87	76.60***	77.46***

Test			Source text against reference text	Model: Mix natural CS — Cascade ASR+ NMT	Direct single-task ST	Direct multi-task ST	Direct multi-task +LID ST
All artificial CS	WER%↓	JaEnCS	17.52	6.90	7.10	5.11***	4.68***
		JaZhCS	56.82	8.56***	10.90	9.60*	7.39***
	CER%↓	JaEnCS	35.23	6.48	6.35	4.99***	4.41***
		JaZhCS	48.98	7.83***	9.79	8.69*	6.67***
	BLEU↑	JaEnCS	74.14	89.29	89.85	92.62***	93.39***
		JaZhCS	46.71	89.08***	86.18	88.10**	90.58***
Mix natural CS	WER%↓	JaEnCS	28.67	20.62***	24.30	19.84***	19.36***
	CER%↓	JaEnCS	54.29	18.73***	20.17	17.32***	16.76***
	BLEU↑	JaEnCS	64.64	75.79	75.23	78.92***	79.39***

Acknowledgements

I would like to express my sincere gratitude to Professor Satoshi Nakamura for giving me the opportunity to research under his guidance and for his invaluable advice.

And this work would not have been possible without the support of Associate Professor Sakriani Sakti. I am grateful for the encouragement, advice, and guidance she gave me.

I also would like to thank all Augmented Human Communication (AHC) Laboratory staff. Ms. Manami Matsuda always cared about us and supported a lot of things for us. I would like to thank her a lot.

The internship opportunity I had with the Human Language Technology (HLT) laboratory of the National University of Singapore was a great chance for learning research technologies and research minds. I would like to thank Professor Haizhou Li, Dr. Emre Yilmaz, Dr. Yuan Min, and all the members of the HLT laboratory for welcoming and supporting me.

I also would like to thank Professor Taro Watanabe as a thesis committee for giving valuable comments and suggestions for this thesis.

I thank all the students of AHC Laboratory for their friendship and supports. Especially, I have to say that I am thankful to members belonging to the same Speech Processing (SP) group in the AHC laboratory. They kindly taught me and supported me in solving my trouble in research.

Finally, I would like to thank my family for supporting me to pursue my study.

References

[1] Japanese Ministry of Health, Labour and Welfare, "Overview of the population statistics in 2016 [in Japanese]," *http://www.mhlw.go.jp/*, 2016.

[2] Japanese Ministry of Education, Culture, Sports, Science, and Technology, "School basic survey in 2015 [in Japanese]," *http://www.mext.go.jp/*, 2015.

[3] Japan Student Service Organization, "Survey on Japanese student abroad situation in 2016 [in Japanese]," *http://www.jasso.go.jp/*, 2016.

[4] Shana Poplack, *Code-switching (linguistic)*, chapter International Encyclopedia of the Social and Behavioral Sciences, pp. 918–925, Elsevier Science Ltd, 2nd edition edition, 2015.

[5] Jeff McSwan, "The architecture of the bilingual language faculty: Evidence from intrasentential code switching," *Bilingualism: Language and Cognition*, vol. 3, no. 1, pp. 37–54, 2000.

[6] Masayo Nakamura, "Developing codeswitching patterns of a Japanese/English bilingual child," in *Proc. ISB4*, Somerville, MA, USA, 2005, pp. 1679–1689.

[7] Sandra S. Fotos, "Japanese-English code switching in bilingual children," *JALT Journal*, vol. 12, no. 1, pp. 75–98, 1990.

[8] Ka Long Roy Chan et al., "Trilingual code-switching in Hong Kong," *Applied Linguistics Research Journal*, vol. 3, no. 4, pp. 1–14, 2019.

[9] Zhili Tan, Xinghua Fan, Hui Zhu, and Ed Lin, "Addressing accent mismatch in Mandarin-English code-switching speech recognition," in *Proc. ICASSP*, 2020, pp. 8259–8263.

[10] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura, "Listening while speaking: Speech chain by deep learning," in *Proc. ASRU*, Okinawa, Japan, 2017, IEEE, pp. 301–308.

[11] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura, "Machine speech chain with one-shot speaker adaptation," in *Proc. INTERSPEECH*, Hyderabad, India, 2018, IEEE, pp. 887–891.

[12] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[13] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. EMNLP*, 2014, pp. 1724–1734.

[14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. ICLR*, San Diego, CA, USA, 2015, pp. 1–15.

[15] Thang Luong, Hieu Pham, and Christopher D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. EMNLP*, Lisbon, Portugal, Sept. 2015, pp. 1412–1421, Association for Computational Linguistics.

[16] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio, "End-to-end attention-based large vocabulary speech recognition," in *Proc. ICASSP*. IEEE, 2016, pp. 4945–4949.

[17] William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. ICASSP*, Shanghai, China, 2016, IEEE, pp. 4960–4964.

[18] Yuxuan Wang, RJ 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*, Stockholm, Sweden, 2017, IEEE, pp. 4006–4010.

[19] Chao Li, Xiaokong Ma, Bing Jiang, Xiangang Li, Xuewei Zhang, Xiao Liu, Ying Cao, Ajay Kannan, and Zhenyao Zhu, "Deep speaker: an end-to-end neural speaker embedding system," *arXiv preprint arXiv:1705.02304*, 2017.

[20] Anik Dey and Pascale Fung, "A Hindi-English code-switching corpus," in *Proc. LREC*, 2014, pp. 2410–2413.

[21] Han-Ping Shen, Chung-Hsien Wu, Yan-Ting Yang, and Chun-Shan Hsu, "CECOS: A Chinese-English code-switching speech database," in *Proc. O-COCOSDA*. IEEE, 2011, pp. 120–123.

[22] Dau-Cheng Lyu, Tien-Ping Tan, Eng Siong Chng, and Haizhou Li, "SEAME: a Mandarin-English code-switching speech corpus in South-East Asia," in *Proc. INTERSPEECH*, 2010, pp. 1986–1989.

[23] Joyce YC Chan, PC Ching, and Tan Lee, "Development of a Cantonese-English code-mixing speech corpus," in *Ninth European Conference on Speech Communication and Technology*, 2005.

[24] Emre Yilmaz, Jelske Dijkstra, H Velde, H Heuvel, and DA van Leeuwen, "Longitudinal speaker clustering and verification corpus with code-switching Frisian-Dutch speech," pp. 37–41, 2017.

[25] Ewald van der Westhuizen and Thomas Niesler, "Automatic speech recognition of English-isiZulu code-switched speech from South African soap operas," *Procedia Computer Science*, vol. 81, pp. 121–127, 2016.

[26] Toshiyuki Takezawa, Genichiro Kikui, Masahide Mizushima, and Eiichiro Sumita, "Multilingual spoken language corpus development for communication research," *The Association for Computational Linguistics and Chinese Language Processing*, vol. 12, no. 3, pp. 303–324, 2007.

[27] Genichiro Kikui, Eiichiro Sumita, Toshiyuki Takezawa, and Seiichi Yamamoto, “Creating corpora for speech-to-speech translation,” in *Proc. ISCA EUROSPEECH*, Geneva, Switzerland, 2003, pp. 381–384.

[28] Pierre Nicolas Durette, “gTTS – Google Text-to-Speech,” https://pypi.org/project/gTTS/.

[29] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, “LibriSpeech: an ASR corpus based on public domain audio books,” in *Proc. ICASSP*, 2015, pp. 5206–5210.

[30] Hui Bu, Jiayu Du, Xingyu Na, Bengu Wu, and Hao Zheng, “AISHELL-1: An open-source Mandarin speech corpus and a speech recognition baseline,” in *Proc. O-COCOSDA*, 2017, pp. 1–5.

[31] Zhiping Zeng, Yerbolat Khassanov, Van Tung Pham, Haihua Xu, Eng Siong Chng, and Haizhou Li, “On the end-to-end solution to Mandarin-English code-switching speech recognition,” in *Proc. INTERSPEECH*, 2019, pp. 2165–2169.

[32] Ye Jia, Melvin Johnson, Wolfgang Macherey, Ron J Weiss, Yuan Cao, Chung-Cheng Chiu, Naveen Ari, Stella Laurenzo, and Yonghui Wu, “Leveraging weakly supervised data to improve end-to-end speech-to-text translation,” in *Proc. ICASSP*. IEEE, 2019, pp. 7180–7184.

[33] Mark Hasegawa-Johnson, Alan Black, Lucas Ondel, Odette Scharenborg, and Francesco Ciannella, “Image2speech: Automatically generating audio descriptions of images,” *Proc. ICNLSSP*, vol. 1, no. 1, pp. 19–27, 2017.

[34] William Havard, Laurent Besacier, and Olivier Rosec, “SPEECH-COCO: 600k visually grounded spoken captions aligned to MSCOCO data set,” *ISCA Workshop on Grounding Language Understanding (GLU2017)*, 2017.

[35] Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu, and Pascale Fung, “Code-switched language models using neural based synthetic data from parallel sentences,” in *Proc. CoNLL*. IEEE, 2019, pp. 271–280.

[36] Yash Sharma, Basil Abraham, Karan Taneja, and Preethi Jyothi, "Improving low resource code-switched ASR using augmented code-switched TTS," in *Proc. INTERSPEECH*. IEEE, 2020, pp. 4771–4775.

[37] Deepak Gupta, Asif Ekbal, and Pushpak Bhattacharyya, "A semi-supervised approach to generate the code-mixed text using pre-trained encoder and transfer learning," in *Proc. EMNLP*, 2020, pp. 2267–2280.

[38] Abhirut Gupta, Aditya Vavre, and Sunita Sarawagi, "Training data augmentation for code-mixed translation," in *Proc. NAACL-HLT*, 2021, pp. 5760–5766.

[39] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.

[40] Sahoko Nakayama, Takatomo Kano, Quoc Truong Do, Sakriani Sakti, and Satoshi Nakamura, "Japanese-English code-switching speech data construction," in *Proc. O-COCOSDA*, Miyazaki, Japan, 2018, IEEE, pp. 1–6.

[41] Brian McFee, Matt McVicar, Oriol Nieto, Stefan Balke, Carl Thome, Dawen Liang, Eric Battenberg, Josh Moore, Rachel Bittner, Ryuichi Yamamoto, et al., "librosa 0.5.0," *https://librosa.github.io/librosa/0.5.0/index.html*, 2017.

[42] Ngoc Thang Vu, Dau-Cheng Lyu, Jochen Weiner, Dominic Telaar, Tim Schlippe, Fabian Blaicher, Eng-Siong Chng, Tanja Schultz, and Haizhou Li, "A first speech recognition system for Mandarin-English code-switch conversational speech," in *Proc. ICASSP*, Kyoto, Japan, 2012, IEEE, pp. 4889–4892.

[43] Ne Luo, Dongwei Jiang, Shuaijiang Zhao, Caixia Gong, Wei Zou, and Xiangang Li, "Towards end-to-end code-switching speech recognition," *arXiv preprint arXiv:1810.13091*, 2018.

[44] Changhao Shan, Chao Weng, Guangsen Wang, Dan Su, Min Luo, Dong Yu, and Lei Xie, "Investigating end-to-end speech recognition for Mandarin-English code-switching," in *Proc. ICASSP*. IEEE, 2019, pp. 6056–6060.

[45] Basem H.A. Ahmed and Tien-Ping Tan, "Automatic speech recognition of code switching speech using 1-best rescoring," in *Proc. IALP*, Hanoi, Vietnam, 2012, pp. 137–140.

[46] Emre Yılmaz, Henk van den Heuvel, and David van Leeuwen, "Investigating bilingual deep neural networks for automatic recognition of code-switching Frisian speech," *Procedia Computer Science*, vol. 81, pp. 159–166, 2016.

[47] Marc A Zissman and Elliot Singer, "Automatic language identification of telephone speech messages using phoneme recognition and n-gram modeling," in *Proc. ICASSP*. IEEE, 1994, vol. 1, pp. 305–308.

[48] David Martinez, Lukáš Burget, Luciana Ferrer, and Nicolas Scheffer, "iVector-based prosodic system for language identification," in *Proc. ICASSP*. IEEE, 2012, pp. 4861–4864.

[49] Ignacio Lopez-Moreno, Javier Gonzalez-Dominguez, Oldrich Plchot, David Martinez, Joaquin Gonzalez-Rodriguez, and Pedro Moreno, "Automatic language identification using deep neural networks," in *Proc. ICASSP*. IEEE, 2014, pp. 5337–5341.

[50] Dau-Cheng Lyu, Ren-Yuan Lyu, Yuang-chin Chiang, and Chun-Nan Hsu, "Speech recognition on code-switching among the Chinese dialects," in *Proc. ICASSP*. IEEE, 2006, vol. 1, pp. 1105–1108.

[51] Ke Li, Jinyu Li, Guoli Ye, Rui Zhao, and Yifan Gong, "Towards code-switching ASR for end-to-end CTC models," in *Proc. ICASSP*. IEEE, 2019, pp. 6076–6080.

[52] Hiroshi Seki, Shinji Watanabe, Takaaki Hori, Jonathan Le Roux, and John R. Hershey, "An end-to-end language-tracking speech recognizer for mixed-language speech," in *Proc. ICASSP*, Calgary, Canada, 2018, IEEE.

[53] Sankar K Pal and Sushmita Mitra, "Multilayer perceptron, fuzzy sets, classifiaction," *IEEE Trans. Neural Networks*, vol. 3, pp. 683–697, 1992.

[54] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[55] Taku Kudo, "Mecab: Yet another part-of-speech and morphological analyzer," *http://taku910.github.io/mecab*, 2006.

[56] Hiroshi Miura, "pykakasi – kakasi library in python," https://pypi.org/project/pykakasi/.

[57] Huang Huang, "pypinyin – pinyin library in python," https://pypi.org/project/pypinyin/.

[58] Hainan Xu, Shuoyang Ding, and Shinji Watanabe, "Improving end-to-end speech recognition with pronunciation-assisted sub-word modeling," in *Proc. ICASSP*. IEEE, 2019, pp. 7110–7114.

[59] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li, "Empirical evaluation of rectified activations in convolutional network," *Deep Learning Workshop, ICML*, pp. 1–5, 2015.

[60] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in pytorch," in *Proc. NIPS Autodiff Workshop*, 2017.

[61] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al., "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[62] Jonathan G Fiscus, Jerome Ajot, Martial Michel, and John S Garofolo, "The rich transcription 2006 spring meeting recognition evaluation," in *International Workshop on Machine Learning for Multimodal Interaction*. Springer, 2006, pp. 309–322.

[63] David S Pallet, William M Fisher, and Jonathan G Fiscus, "Tools for the analysis of benchmark speech recognition tests," in *Proc. ICASSP*. IEEE, 1990, pp. 97–100.

[64] Xiaojin Zhu and Zoubin Ghahramani, "Learning from labeled and unlabeled data with label propagation," *Technical Report*, 2002.

[65] Pengcheng Guo, Haihua Xu, Lei Xie, and Eng Siong Chng, "Study of semi-supervised approaches to improving English-Mandarin code-switching speech recognition," in *Proc. INTERSPEECH.* Sep 2018, pp. 1928–1932, ISCA.

[66] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio, "Zero-data learning of new tasks," in *Proc. AAAI*, 2008, vol. 1, pp. 646–651.

[67] Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean, "Google's multilingual neural machine translation system: Enabling zero-shot translation," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 339–351, Dec 2017.

[68] Peter B. Denes and Elliot N. Pinson, *The Speech Chain: The Physics And Biology Of Spoken Language*, Anchor books. Worth Publishers, 1993.

[69] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura, "End-to-end feedback loss in speech chain framework via straight-through estimator," in *Proc. ICASSP*, Brighton, UK, 2019, IEEE, pp. 6281–6285.

[70] Sahoko Nakayama, Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura, "Speech chain for semi-supervised learning of Japanese-English code-switching ASR and TTS," in *Proc. SLT*, Athens, Greece, 2018, IEEE, pp. 182–189.

[71] Sahoko Nakayama, Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura, "Zero-shot code-switching ASR and TTS with multilingual machine speech chain," in *Proc. ASRU*, Sentosa, Singapore, 2019, IEEE, pp. 964–971.

[72] Philip Gage, "A new algorithm for data compression," *C Users Journal*, vol. 12, no. 2, pp. 23–38, 1994.

[73] Donald J Hernandez, Nancy A Denton, and Suzanne E Macartney, "Children in immigrant families: Looking to America's future. social policy report. volume 22, number 3.," *Society for Research in Child Development*, 2008.

[74] R Mahesh K Sinha and Anil Thakur, "Machine translation of bi-lingual Hindi-English (Hinglish) text," *Proc. MT Summit X*, pp. 149–156, 2005.

[75] Mohamed Amine Menacer, David Langlois, Denis Jouvet, Dominique Fohr, Odile Mella, and Kamel Smaïli, "Machine translation on a parallel code-switched corpus," in *Proc. Canadian AI*, 2019, pp. 426–432.

[76] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Proc. NIPS*, 2017, pp. 5998–6008.

[77] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. EMNLP*, 2013, pp. 1631–1642.

[78] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang, "Squad: 100,000+ questions for machine comprehension of text," in *Proc. EMNLP*, 2016, pp. 2383–2392.

[79] Erik F. Tjong Kim Sang and Fien De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," in *Proc. NAACL-HLT*, 2003, pp. 142–147.

[80] Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer, "Constant-time machine translation with conditional masked language models," in *Proc. EMNLP*, 2019, pp. 6111–6120.

[81] Ron J. Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen, "Sequence-to-sequence models can directly translate foreign speech," in *Proc. INTERSPEECH*, 2017.

[82] Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur, "Audio augmentation for speech recognition," in *Proc. INTERSPEECH*, 2015, pp. 3586–3589.

[83] Tom Ko, Vijayaditya Peddinti, Daniel Povey, Michael L Seltzer, and Sanjeev Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *Proc. ICASSP*. IEEE, 2017, pp. 5220–5224.

[84] Mike Schuster and Kaisuke Nakajima, "Japanese and Korean voice search," in *Proc. ICASSP*. IEEE, 2012, pp. 5149–5152.

[85] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu, "BLEU: a method for automatic evaluation of machine translation," in *Proc. ACL*, 2002, pp. 311–318.

[86] Philipp Koehn, "Statistical significance tests for machine translation evaluation," in *Proc. EMNLP*, Barcelona, Spain, July 2004, pp. 388–395, Association for Computational Linguistics.

[87] Heting Gao, Junrui Ni, Yang Zhang, Kaizhi Qian, Shiyu Chang, and Mark Hasegawa-Johnson, "Zero-shot cross-lingual phonetic recognition with external language embedding," in *Proc. INTERSPEECH*, 2021, pp. 1304–1308.

[88] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli, "wav2vec: Unsupervised pre-training for speech recognition," in *Proc. INTERSPEECH*, 2019, pp. 3465–3469.

[89] Xinjian Li, Siddharth Dalmia, Juncheng Li, Matthew Lee, Patrick Littell, Jiali Yao, Antonios Anastasopoulos, David R Mortensen, Graham Neubig, Alan W Black, et al., "Universal phone recognition with a multilingual allophone system," in *Proc. ICASSP*. IEEE, 2020, pp. 8249–8253.

[90] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le, "Specaugment: A simple data augmentation method for automatic speech recognition," in *Proc. INTERSPEECH*, 2019, pp. 2613–2617.

[91] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al., "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. INTERSPEECH*, 2020, pp. 5036–5040.

[92] Abhinav Jain, Minali Upreti, and Preethi Jyothi, “Improved accented speech recognition using accent embeddings and multi-task learning,” in *Proc. INTERSPEECH*, 2018, pp. 2454–2458.

[93] Javier Gonzalez-Dominguez, David Eustis, Ignacio Lopez-Moreno, Andrew Senior, Françoise Beaufays, and Pedro J Moreno, “A real-time end-to-end multilingual speech recognition architecture,” *IEEE Journal of selected topics in signal processing*, vol. 9, no. 4, pp. 749–759, 2015.

List of Publications

Journal Paper (peer-reviewed)

1. "Code-Switching ASR and TTS using Semisupervised Learning with Machine Speech Chain"
   Sahoko Nakayama, Andros Tjandra, Sakriani Sakti, Satoshi Nakamura
   IEICE Transactions on Information and Systems, Vol. E104-D, No.10, pp.1661-1677, Oct. 2021
   (Related to Chapter 4, 5)

2. "単言語話者のための日英コードスイッチング音声の認識と翻訳"
   中山佐保子, サクティ サクリアニ, 中村哲
   情報処理学会論文誌, Vol.62 No.3, pp.903–914, Mar. 2021
   (Related to Chapter 6)

International Conference Paper (peer-reviewed)

1. "Zero-Shot Code-Switching ASR and TTS with Multilingual Machine Speech Chain"
   Sahoko Nakayama, Andros Tjandra, Sakriani Sakti, Satoshi Nakamura
   Proc. Automatic Speech Recognition and Understanding (ASRU), pp. 964-971, Dec. 2019
   (Related to Chapter 4, 5)

2. "Recognition and Translation of Code-switching Speech Utterances"
   Sahoko Nakayama, Takatomo Kano, Andros Tjandra, Sakriani Sakti, Satoshi Nakamura
   Proc. Oriental COCOSDA (O-COCOSDA), IEEE, pp. 1-6 , Oct. 2019
   (Related to Chapter 6)

3. "Speech Chain for Semi-supervised Learning of Japanese-English Code-switching ASR and TTS"
   Sahoko Nakayama, Andros Tjandra, Sakriani Sakti, Satoshi Nakamura
   Proc. IEEE Spoken Language Technology (SLT), IEEE, pp.182-189, Dec.

2018
(Related to Chapter 5)

4. “Japanese-English Code-switching Speech Data Construction”
Sahoko Nakayama, Takatomo Kano, Quoc Truong Do, Sakriani Sakti, Satoshi Nakamura
Proc. Oriental COCOSDA (O-COCOSDA), IEEE, pp. 67-71, May. 2018
(Related to Chapter 3)

Domestic Conference Paper

1. “マルチリンガルマシーンスピーチチェーンを用いたゼロショットコードスイッチングの音声認識と音声合成”
中山佐保子, チャンドラ アンドロス, サクティ サクリアニ, 中村哲
日本音響学会2021年春季研究発表会講演論文集, 日本音響学会, pp. 887–888, Mar. 2021
(Related to Chapter 4, 5)

2. “Machine Speech Chainに基づく半教師あり学習を用いた日英コードスイッチング音声の認識”
中山佐保子, チャンドラ アンドロス, サクティ サクリアニ, 中村哲
言語処理学会第25回年次大会発表論文集,言語処理学会, pp.179-182, Mar. 2019
(Related to Chapter 5)

3. “日英コードスイッチング 音声データの構築”
中山佐保子, ド クオック チュオン, サクティ サクリアニ, 中村哲
日本音響学会2018年秋季研究発表会講演論文集,日本音響学会, pp.1013-1014, Sep. 2018
(Related to Chapter 3)