END-TO-END SIMULTANEOUS SPEECH TRANSLATION

by

Xutai Ma

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

October, 2022

Abstract

Speech translation is the task of translating speech in one language to text or speech in another language, while simultaneous translation aims at lower translation latency by starting the translation before the speaker finishes a sentence. The combination of the two, simultaneous speech translation, can be applied in low latency scenarios such as live video caption translation and real-time interpretation.

This thesis will focus on an end-to-end or direct approach for simultaneous speech translation. We first define the task of simultaneous speech translation, including the challenges of the task and its evaluation metrics. We then progressly introduce our contributions to tackle the challenges. First, we proposed a novel simultaneous translation policy, monotonic multihead attention, for transformer models on text-to-text translation. Second, we investigate the issues and potential solutions when adapting text-to-text simultaneous policies to end-to-end speech-to-text translation models. Third, we introduced the augmented memory transformer encoder for simultaneous speech-to-text translation models for better

computation efficiency. Fourth, we explore a direct simultaneous speech translation with variational monotonic multihead attention policy, based on recent speech-to-unit models. At the end, we provide some directions for potential future research.

Primary Reader and Advisor: Philipp Koehn

Secondary Readers: Sanjeev Khudanpur, Juan Pino

Acknowledgments

First of all, I would like to express my sincere gratitude to my PhD advisor Philipp Kohen. When I first started the journey of machine translation in 2017 as a master student, Philipp provided me the opportunity to extend a class project to a research project, which later became my first published paper. Since I started my PhD program in 2018, besides the guidance on research projects, he has given me considerable freedom on research topics, encouraging me to explore different directions. I really appreciate Philipp for the opportunity, the guidance and the freedom that lead me to a meaningful PhD life.

I would like to thank Sanjeev Khudanpur who is on the thesis committee. Sanjeev gave me very constructive suggestions during the graduate board oral and thesis defense. Moreover, he has provided me enormous help throughout my master and PhD life. He taught me knowledge in speech recognition which turned out to be the foundation of this thesis.

I would like to specially thank Juan Pino, and Facebook (Now Meta). Juan has basically

been another advisor of mine since 2019. I had two research internships with Juan in 2019/2020 working on simultaneous speech translation. Between and after the internships, I kept collaborating with Juan on the simultaneous research projects. When we first started in 2019, simultaneous / speech translation were still extremely difficult topics with only a small population working on it. Now we have achieved what we could not imagine at that time. This thesis would not happen without Juan's support over the years. I am fortunate to have the opportunity to continue working with Juan after the PhD program. I look forward to achieving significant breakthroughs in the future.

It is an honor for me to spend my six year in the world class institute of speech and language research, Center of Language and Speech Processing (CLSP). I would like to thank the wonderful faculties in CLSP who have been extremely supportive over the past few years: Najim Dehak, Hynek Hermansky, Benjamin Van Durme, Kevin Duh and Shinji Watanab. Najim provided me the first research opportunity after I came to JHU, which brings to the world of speech application, or even general machine learning. Hynek introduced solid foundations on speech processing which is super important to my current research. Benjamin provided me an opportunity to work in his lab on semantics parsing with Sheng Zhang, as a research assistant (my first paid job) in early 2018 and as PhD student working on the qualifying project. Even though I did not pursue the direction of semantics parsing, those experiences helped me a lot in building foundations for NLP research. Kevin was the

advisor on one of my qualifying projects on cross-lingual information retrieval. However, his support is way more than just a project. Shinji, who is an expert in speech application, greatly inspired me on several research projects. We also have good collaborations on papers and organizing IWSLT shared tasks on simultaneous speech translation. I would like to thank Ruth Scally for her support during my life in CLSP.

CLSP also has the best PhD students in speech and language research. I would like to thank Shuoyang Ding, who generously provided much support and guidance to me as the senior student in the machine translation group; Nanxin Chen, who was my roommate for two years, a good friend and my first teacher on machine learning; Ke Li, who provided great support and guidance on my PhD life. I would like to thank the CLSP colleagues whom I've been working with: Hongyuan Mei, Sheng Zhang, Shuo Sun, Huda Khayrallah, Kelly Marchisio, Arya McCarthy, Adi Renduchintala, Liz Salesky, Pamela Shapiro, Xuan Zhang, Dingquan Wang, Yiming Wang, Zach Wood-Doughty, Sheng Zhang, Tongfei Chen, Hainan Xu

I sincerely appreciate my family: father Shaolin Ma (马少林), mother Hua Su (苏华) and brother Xuda Ma (马旭达) . I am from Urumqi, Xinjiang, China. I would not have such a long but wonderful journey to where I am now without their support.

This thesis is dedicated to my grandparents(马敬贤、李玉兰、苏永瑞、马文英). my grandmother (李玉兰) on my father's side and both grandparents (苏永瑞、马文英) on

my mother's side passed away during my time in JHU. I was not able to go back to attend their funerals, but I will keep exploring the wider world with their names

I have spent much playing asketball during my time in JHU, which is one of my best memories. I was the leader of the JHU Chinese Basketball Team. I met my best friends in JHU in the team: Shuang Liu (刘爽), Shuran Zhang (张舒然), Quanjun Lang (郎泉钧) and Shenghao Guo (郭晟昊)

I would like to thank to Nanxi Zeng (曾楠希) who not only supports me during my master and PhD life, but also has a profound influence on me that make me a better person.

I would like to thank Minah Kang (강민아) who has been super supportive during my most struggling time writing the thesis and given my PhD life a wonderful ending.

紀念

昌吉爺爺、奶奶：馬敬賢、李玉蘭

銀行爺爺、奶奶：蘇永瑞、馬文英

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Machine translation quality has been greatly improved over the past few years, especially with the rise of neural methods. However, most recent work on machine translation focuses on the translation from text. Given that one of the important situations with translation applied is oral communication between people from different linguistic backgrounds, the direct translation from human speech in real-time is also of great importance. Such real-time systems can be deployed in scenarios such as translation of streaming videos, interpretation of speakers in international conferences, or personal translator. More specifically, for a real-time speech translation system, two important factors should be considered:

1. The translation system is capable of processing the speech input, and generating correct translation in text or speech output.

2. The latency of the translation system should be low enough without huge sacrifice on translation quality for real-time communication.

The first factor defines a speech translation system, which maps a source speech utterance to target text or speech. In fact, speech translation has been heavily investigated since 1980s. However, most of the work focuses on the cascaded approach, where several submodules are involved:

1. An automatic speech recognition (ASR) system to transcribe source speech to source text.

2. A text-to-text machine translation system to translate source text to target text.

3. An optional text-to-speech synthesis system to synthesize target text to target speech.

While the cascaded approach can be built on top of several developed submodules, it has two disadvantages. First, errors can accumulate through the pipeline of submodules, and it is non-trivial to recover from such errors. Second, the pipeline structure introduces extra latency to the whole system, due to the additional computation time and a possible lack of synchronism between submodules. Recent neural-based end-to-end speech translation models have been developed as the solution for these two issues. They have shown strong performance over the cascaded approach, especially when large amounts of labeled and weakly labeled data are available. For streaming translation, an end-to-end speech translation

system is an even better solution for following reasones,

- The end-to-end model can over perform the cascaded model under certain settings. For instance, Wang et al. (2022) show that the end-to-end approach for simultaneous speech-to-text translation can be competitive, and sometimes better, compared with cascaded approach.

- The End-to-end model can achieve lower latency because of no synchronization between components. For instance, Chen et al. (2021) shows that the cascaded model will introduce extra latency because the translation module has to wait until streaming ASR's output stabilizes.

- The end-to-end model can save disk space which is crucial for on-device applications. For instance, Inaguma et al. (2020) show that the end-to-end model could achieve similar performance with half of the parameters compared with the cascaded model.

The second factor for a real-time speech translation system is low latency. The latency of a speech translation can come from two parts, one is the computation time of the model, the other one is the delay caused by the algorithm which determines the wait time for translation. The first part is a general topic for neural models, which can be addressed by methods such as model quantilization or compressing. The second part, on the other hand, is a more specific problem on sequence-to-sequence generation. To address this

issue, an algorithm is preferred, in which the model is capable of starting the generation of the target output before reading the whole source input. The study of such algorithms is referred to as simultaneous translation. While great progress has been made on the study of simultaneous translation, most of the work focuses on text-to-text translation. However, the speech translation scenarios are where the most simultaneous translation are needed.

In this thesis, I will focus on the design of an end-to-end simultaneous speech translation model, which can be potentially deployed in real-time applications. The major contributions of the thesis include the following aspects:

- The evaluation of an end-to-end simultaneous speech translation model. It is of great importance to determine the evaluation metrics for the system before designing it. The evaluation of a simultaneous model includes two aspects: quality and latency. While the quality evaluation is similar to the offline system, the evaluation of latency has to be well-defined, with consideration of both algorithm delay and computation time. We present a comprehensive definition and evaluation of the simultaneous speech translation task. We also developed the open-source toolkit SimulEval for simultaneous translation evaluation, as published in Ma et al. (2020b) (Chapter 2)

- The development of the state-of-the-art text-to-text simultaneous translation policy, monotonic multihead attention (MMA), along with novel latency regularization methods, as published in Ma et al. (2019b). MMA adapts monotonic attention based

methods to the state-of-the-art transformer (Vaswani et al., 2017) models, achieving a better latency-quality trade-off than prior work. (Chapter 4)

- The adaptation of text-to-text simultaneous policies to the end-to-end speech translation, as published in Ma et al. (2020). This thesis is one of the first to discuss simultaneous translation for end-to-end speech-to-text translation. We analyze the difficulty of such adaptation, including the longer length and more continuous characteristics of speech input compared with text input. The concept of a pre-decision module is proposed to enable the text-to-text simultaneous policy work on speech input. (Chapter 5)

- The development of augmented memory transformer for streaming translation, as publised in Ma et al. (2021). The augmented memory transformer (Wu et al., 2020) was first introduced as a transducer-based model for streaming automatic speech translation. We apply the augmented memory transformer to simultaneous speech translation to achieve better computation aware latency. (Chapter 6)

- The first attempt of direct simultaneous speech-to-speech translation. A new policy, variational monotonic attention is introduced to handle not only the long speech input but also long speech output. The policy is then applied to the direct speech-to-unit (Lee et al., 2022) model which is independent of intermediate text. (Chapter 7)

The summarization of the remaining chapters is as below

- Chapter 3 provides a comprehensive introduction on the related work, including works on speech-to-text and speech-to-speech translation in both cascade and end-to-end model. Simultaneous policies, especially monotonic attention based methods are also introduced.

- Chapter 8 summarizes the thesis, including the findings, limitations and potential future directions.

Chapter 2

Task Formalization

2.1 Introduction

In order to understand the challenges of simultaneous speech translation, it is of great importance to have a comprehensive definition on the task, along with the evaluation metrics. There are two aspects of the simultaneous speech translation: speech and simultaneous. The goal for speech part is to build a system that is able to process human speech, targeting a more direct translation for communications. A speech translation system automatically generates text or speech in target language from source speech. Similar to text translation, the quality of the translation defines the competence of the system. On the other hand, the simultaneous part indicates that such generation happens concurrently with the input, where

the translation starts before the speaker finish a sentence. It focuses on the promptness of translation, similar to simultaneous human interpretation. Therefore, besides the translation quality, the latency of the translation is another major factor to consider for simultaneous translation.

In this chapter, we will give a comprehensive formalization of the task of simultaneous speech translation. We first introduce the basic concept of speech translation, including the processing of speech signals, speech-to-text and speech-to-speech translation, and quality evaluation of a speech translation system. Next we discuss the concept of simultaneous translation, along with the definition of latency metrics, the simultaneous evaluation setup, and a toolkit for evaluation.

2.2 Speech Translation

The task of speech translation is a sequence-to-sequence problem, which expects a probabilistic model, shown as Equation 2.1

$$\hat{\mathbf{Y}} = \arg\max_{\mathbf{Y}} \mathbf{P}(\mathbf{Y}|\mathbf{X}) \tag{2.1}$$

Where the $\mathbf{X}$ is input, and $\mathbf{Y}$ is output, $\hat{\mathbf{Y}}$ is the best hypothesis given a probabilistic model. Different input and output media and languages result in different tasks. Define f as source

language and e as target language. Denote $\mathcal{S}_f(t)$ and $\mathcal{S}_e(t)$ as source and target speech signals, $\mathbf{W}_f$ and $\mathbf{W}_e$ as source and target text. t is the time variable and $\mathcal{S}_*(t)$ are real time signals. $\mathcal{S}_f(t)$ and $\mathcal{S}_e(t)$ are time series values, while $\mathbf{W}_f$ and $\mathbf{W}_e$ are sequences of text tokens. Given different types of input and output, the sequence-to-sequence tasks can be categorized into four types, shown in Table 2.1. This thesis focus on speech translation (ST), in which the input is speech in source language and the output can be text or speech in target language, as listed speech-to-text (S2T) and speech-to-speech (S2S) translation in Table 2.1. Figure 2.1 illustrates the relationships of tasks in speech applications.

Task	input	output
Automatic Speech Recognition	$\mathcal{S}_f(t)$	$\mathbf{W}_f$
Text-to-text Translation	$\mathbf{W}_f$	$\mathbf{W}_e$
Speech-to-text Translation	$\mathcal{S}_f(t)$	$\mathbf{W}_e$
Speech-to-speech Translation	$\mathcal{S}_f(t)$	$\mathcal{S}_e(t)$
Text-to-speech Translation	$\mathbf{W}_f$	$\mathcal{S}_e(t)$
Text-to-speech Synthesis	$\mathbf{W}_e$	$\mathcal{S}_e(t)$

Table 2.1: Sequence generation tasks with different input and output.

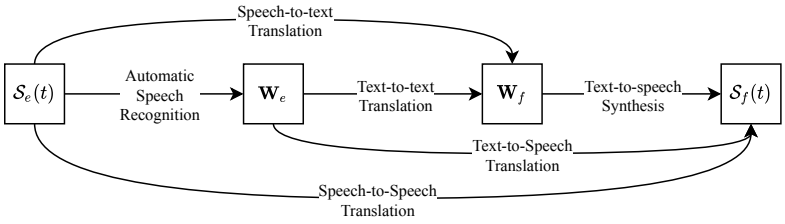

Figure 2.1: Tasks in speech applications.

2.2.1 Speech Processing

Speech signals, or waveform, are a continuous function of time, $\mathcal{S}(t)$, which are usually sampled to reduced to discrete values for storing and processing. Denote the sampling frequency as f_s Hz. The fundamental frequency of a typical adult man ranges from 80 to 180 Hz and that of a typical adult woman from 165 to 255 Hz. Furthermore, there are higher frequencies in human speech containing other information, such as speaker identification and timbre. Thus, the sample rate f_s has to be high enough to capture the adequate human speech information for speech applications. Some common sample rates in different applications are shown as Table 2.2

Sample rate (kHz)	Applications
8	Analogy telephone
16	Modern voice over IP communication
22.05	AM Radio
44.1	Audio CD
48	DV, digital TV, DVD, and films

Table 2.2: Sample rates in different speech applications.

Despite of being discrete, the sampled speech waveform is still too sparse to directly work with. For instance, an utterance of speech of T seconds will be represented by $T \cdot f_s$ scalar values. Therefore, most speech systems operates on speech features, which are parametric representations mapping small segments of speech utterance into vectors. Define the window size as T_w and step size as T_s. A speech waveform $\mathcal{S}(t)$ can be represented as a

sequence of vectors $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, ...]$, such that,

$$\mathbf{s}_i = f\left(\mathcal{S}(t)\right),\ i \cdot T_s - \frac{T_w}{2} < t < i \cdot T_s + \frac{T_w}{2} \tag{2.2}$$

where f is a feature extraction function. The design of the feature extraction has been studied for decades. They are usually functions converting signals from time to frequency domain, over small segments of the speech utterance. $\mathbf{S}$ is also known as speech features, and the popular choices include Short-Time Fourier Transform (STFT) (Oliver, 1952), Mel-Frequency Cepstrum Coefficients (MFCC) (Davis and Mermelstein, 1980), Linear Prediction Coding (LPC) (Portnoff, 1981), Perceptual Linear Prediction (PLP) (Hermansky, 1990), and Log Mel-Filter Bank (LMFB) (Murthy et al., 1999). In practice, the features are low-dimensional vectors (less than 100), and common practice for timing is $T_w = 25$ms and step size as $T_s = 10$ms.

Speech processing is also known as the front-end procedure, which is a huge topic in speech applications. But there will not be further discuss in this thesis, since its not our focus. We will treat the front-end procedure as a black box. Meanwhile, instead of $\mathcal{S}_f(t)$, we will assume the input of the system is source speech feature $\mathbf{S}_f$, On the other hand, the processing of target speech can be different for tasks, and will be introduced in later chapters.

2.2.2 Speech-to-Text Translation

A speech-to-text (S2T) translation system is a system which translation source speech into target text. An illustration of S2T system is shown as Figure 2.2. More specifically, the system takes source speech feature $\mathbf{S}_f$ and generates target text $\mathbf{W}_e$. Also as shown in Figure 2.2, there are two approaches to build a S2T translation system: **cascaded** and **end-to-end**.

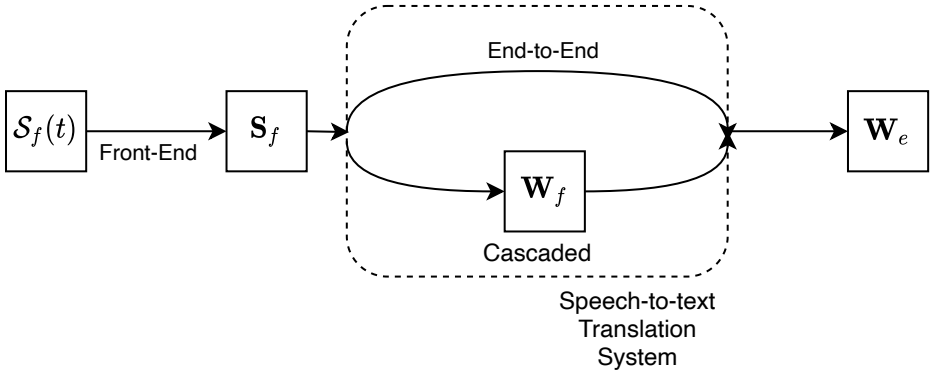

Figure 2.2: Speech-to-text (S2T) translation.

The cascaded system includes two submodules — An automatic speech recognition module that transcribes source speech $\mathbf{S}_f$ to source text $\mathbf{W}_f$, and a text-to-text machine translation system to translate $\mathbf{W}_f$ to $\mathbf{W}_e$. One big advantage of the cascaded approach is that it can be built on top of existing strong systems. As shown in late chapter, the cascaded approach is the most common direction in the early speech translation research. However, the cascaded model needs two pass generation and thus will introduce extra latency and

compounding errors.

The end-to-end system directly generates $\mathbf{W}_e$ without the intermediate text $\mathbf{W}_f$. In spite of requiring more training data, the advantages of the end-to-end approach include low latency and less accumulated errors. Recently, end-to-end systems have become the state-of-the-art for the S2T translation task. In this thesis, we focus on end-to-end approach for translation.

The evaluation of the S2T systems is similar to text-to-text speech translation, in which the quality is usually measured by automatic metrics, such as BLEU (Papineni et al., 2002) TER (Snover et al., 2006) and METEOR (Banerjee and Lavie, 2005). In the rest of the thesis, the translation output is evaluated with the BLEU score. We use the toolkit SacreBLEU (Post, 2018).

2.2.3 Speech-to-Speech Translation

The input and output of a speech-to-speech (S2S) translation system are both speech, shown as Figure 2.3. The S2S can also be categorized into two types: cascaded and direct. Similar to the cascaded S2T, the cascaded S2S is constituted by submodules: a S2T system, either cascaded or end-to-end, and a text-to-speech synthesis (TTS) system which is applied on top of $\mathbf{W}_e$ to generate the final output $\mathcal{S}_e(t)$. Such an approach requires at least target text $\mathbf{W}_e$. On the other hand, the direct approach addresses the independence of intermediate

text, either source or target. The direct approach focus on the immediate generation of target speech features $\mathbf{S}_e$ from $\mathbf{S}_f$. An extra vocoder is applied after $\mathbf{S}_e$. It can be treated as an reversed operation of front-end feature extraction, and is separately trained and light-weight.

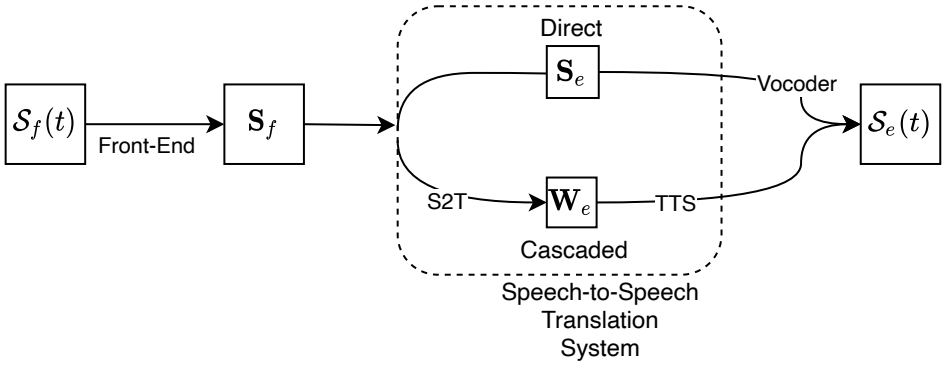

Figure 2.3: Speech-to-text (S2T) translation.

The evaluation of S2S system considers two aspects, the translation quality and the speech quality. For translation quality, the output is transcribed by a ASR system into text, which is then evaluated with text translation quality metrics. The speech quality, or the naturalness of the speech, is usually evaluated by mean opinion score (MOS) from human evaluators, who give a score from 1-5 on each speech output. Such setting has been widely adopted in the TTS system evaluation.

2.3 Simultaneous Translation

Simultaneous translation is the task which starts the translation before the end of source input. It focus on the latency of the translation, which is important for applications such as live video translation, personal translator and international conferences. In this section, we will mathematically define the task of simultaneous translation, and its evaluation scheme. Figure 2.4 shows the difference between an offline and simultaneous model.

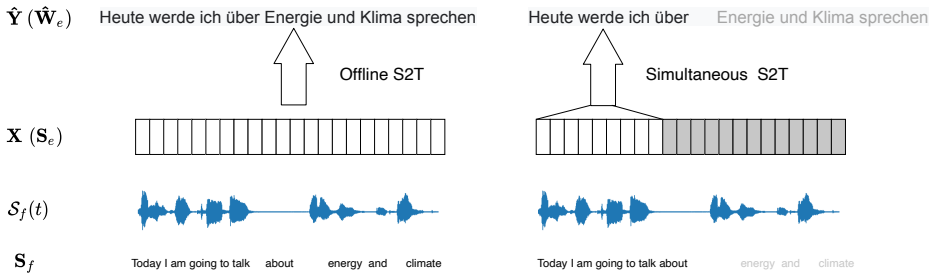

Figure 2.4: Difference between offline and simultaneous translation. Simultaneous translation starts generation before the end of the sentence.

As described in Equation 2.1, the translation system task takes input sequence $\mathbf{X}$ and generate output sequence $\hat{\mathbf{Y}}$. Denote the corresponding output text is $\hat{\mathbf{W}}_e = [\hat{w}_e^1, ..., \hat{w}_e^{(|\hat{\mathbf{W}}_e|)}]$, which is either $\hat{\mathbf{Y}}$ itself if the output is text or transcription of $\hat{\mathbf{Y}}$ if the output is speech. Additionally, for speech output, we define $T_{\hat{w}_e^i}$ as the duration of $\hat{w}_e^i$. We define the delay sequence $\mathbf{D}$, in which each item d_i is the length of input that has been used to generate $\hat{w}_e^i$. Figure 2.5 is an example of the relationships among these variables in a speech-to-text translation setting.

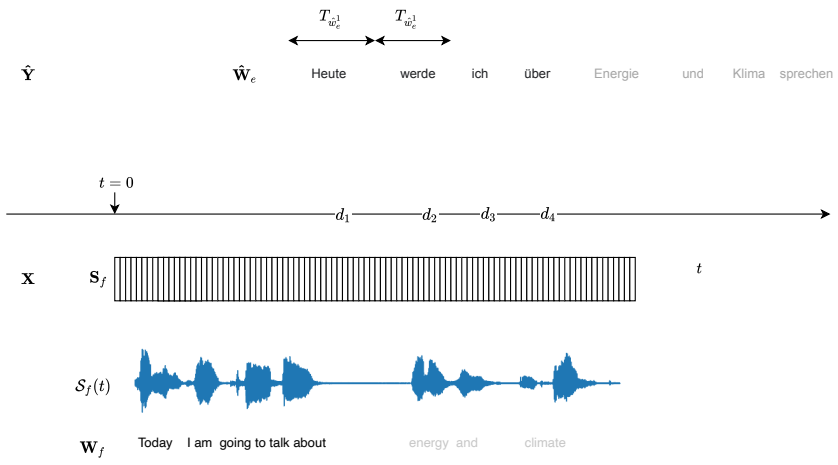

Figure 2.5: An example of variables in simultaneous speech-to-speech translation: time t, source speech signal $\mathcal{S}_f(t)$, source speech feature $\mathbf{S}_f$, source text $\mathbf{W}_f$, target word $\mathbf{W}_e$, duration of the target words $T_{\hat{w}_e^i}$ and delays d_i. $\mathbf{X}$ and $\hat{\mathbf{Y}}$ represent system input and output. In this example, $\mathbf{X} = \mathbf{S}_f$ and $\hat{\mathbf{Y}} = \mathbf{W}_e$

$\mathbf{D}$ is a monotonic increase sequence and $|\mathbf{D}| = |\hat{\mathbf{W}}_e|$. Thus, a translation system is simultaneous as long as there exists $i < |\hat{\mathbf{W}}_e|$ such that $d_i < |\mathbf{X}|$. Meanwhile, an offline translation means $d_i = |\mathbf{X}|$ for all i. The measurement of $\mathbf{D}$ varies with input and output media. Assuming $\hat{w}_e^i$ is generated from $[x_1, ...x_j]$, the measurement of d_i is defined in Table 2.3

Input $\mathbf{X}$	Output $\hat{\mathbf{Y}}$	Delay	Measurement
$\mathbf{W}_f$	$\mathbf{W}_e$	j	Number of tokens
$\mathbf{S}_f$	$\mathbf{W}_e$	$j \cdot T_s$	Time
$\mathbf{S}_f$	$\mathbf{S}_e$	$\max(j \cdot T_s + T_{\hat{w}_e^i}, d_{i-1} + T_{\hat{w}_e^i})$	Time

Table 2.3: Sequence generation tasks with different input and output, and the corresponding when $\hat{y}_i$ is generated by $[x_1, ..., x_j]$. T_s is the feature step size in source feature extractor.

While the delay for text output (first two lines in Table 2.3) is straightforward, the situation can be complicated since the speech output does not finish immediately after generation. For speech output we define the delay at the end of each transcribed word, shown in Figure 2.6 There are two situations. In the first situation, shown as (A) in Figure 2.6,

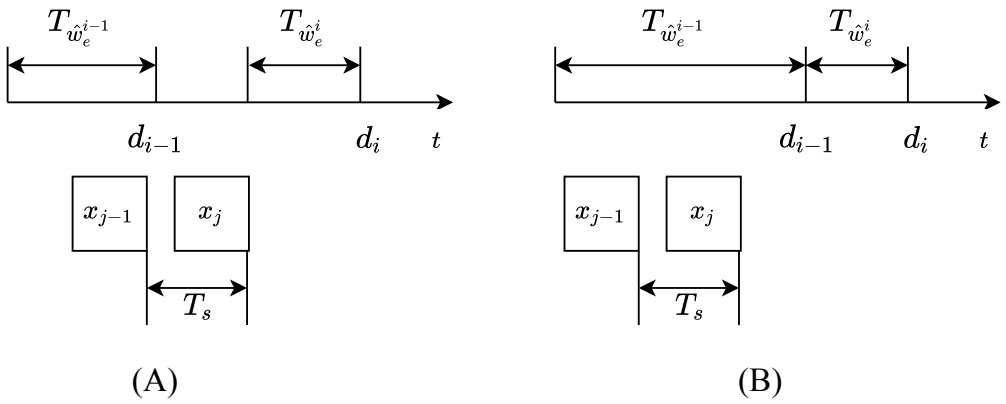

Figure 2.6: Two situations when measure the delay d_i for $\hat{t}_{f(i)}$: (A) $d_i = j \cdot T_s + T_{\hat{w}_e^i}$ (B) $d_i = d_{i-1} + T_{\hat{w}_e^i}$. In (A), the system is about to finish the previous words before the new input, while in (B) the system keep generating speech signal when new input coming in.

the reading of x_j happens after the full generation of previous token $T_{\hat{w}_e^i}$. The delay in this case is straightforward, which is the sum of source speech duration $j \cdot T_s$ and the current prediction duration $T_{\hat{w}_e^i}$. The other situation, shown as (B) in Figure 2.6, is that even though the system already read x_j, the generation of previous word $\hat{w}_e^{i-1}$ is not finished. In this case, the delay is simply $d_{i-1} + T_{\hat{w}_e^i}$. We use $\max$ operation in Table 2.3 to cover both situation. The long delay in (B) can be either from long target tokens or computation time.

In real application, one has to consider the computation during generation. Thus, in Ma

et al. (2020), we introduce the concept of computation-aware (CA) and a non computation-aware (NCA) delay. The CA delay of y_i, $d_{\text{CA}}(y_i)$, is defined as the time that elapses from the beginning of the process to the prediction of y_i, while the NCA delay for y_i, $d_{\text{CA}}(y_i)$ is defined in Table 2.3. Note that d_{NCA} is an ideal case for d_{CA} where the computational time for the model is ignored. In other words, d_{NCA} measures the latency caused by the policy, while d_{CA} additionally takes the system runtime into consideration. Figure 2.7 is an example of both delays.

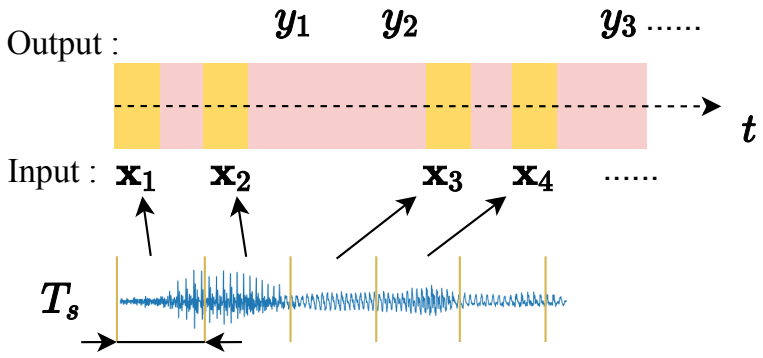

Figure 2.7: Illustration the difference between d_{CA} and d_{NCA}. Yellow block is the window shift time for feature extraction and red blocks are computational time for an SimulS2T model. d_{CA} consider both yellow and red blocks, while d_{NCA} only consider yellow ones

The delay sequence $\mathbf{D}$ is of great importance for simultaneous translation. The way to evaluate latency of a simultaneous translation system is to compute latency metrics from $\mathbf{D}$. In the rest of the section, we will introduce the automatic latency metric calculated from $\mathbf{D}$, and the toolkit used to evaluate latency.

2.3.1 Latency Metrics

We first introduce the three most common metrics, all of which were first introduced in text-to-text simultaneous translation. All the latency metrics are defined as functions, of which the input is $\mathbf{D}$ and output is a scalar.

Average Proportion (AP) (Cho and Esipova, 2016), defined in Equation 2.3, measures the average of proportion of source input read when generating a target prediction.

$$\text{AP} = \frac{1}{|\mathbf{X}||\hat{\mathbf{Y}}|} \sum_{i=1}^{|\hat{\mathbf{Y}}|} d_i \tag{2.3}$$

Despite AP's simplicity, several concerns have been raised. Specifically, AP is not length invariant, i.e. the value of the metric depends on the input and output lengths. Moreover, AP is not evenly distributed over the $[0, 1]$ interval, i.e., values below 0.5 represent models that have lower latency than an ideal policy, and an improvement of 0.1 from 0.7 to 0.6 is much more difficult to obtain than the same absolute improvement from 0.9 to 0.8 (Ma et al., 2019a). Therefore, AP has become less popular in recent simultaneous translation papers.

Average Lagging (AL) (Ma et al., 2019a) first defined an ideal policy, which is equivalent to a wait-0 policy that has the same prediction as the system to be evaluated. Ma et al. (2019a) define AL as

$$\text{AL} = \frac{1}{\tau(|\mathbf{X}|)} \sum_{i=1}^{\tau(|\mathbf{X}|)} d_i - \frac{(i-1)}{\gamma} \tag{2.4}$$

where $\tau(|\mathbf{X}|) = \min\{i|d_i = |\mathbf{X}|\}$ is the index of the target token when the policy first reaches the end of the source sentence and $\gamma = |\hat{\mathbf{Y}}|/|\mathbf{X}|$. $(i-1)/\gamma$ term is the ideal policy for the system to compare with. AL has good properties such as being length-invariant and intuitive. Its value directly describes the lagging behind the ideal policy.

While Equation 2.4 is defined on text input and output, we extend AL to speech translation in Ma et al. (2020) as

$$\text{AL}_{\text{speech}} = \frac{1}{\tau'(|\mathbf{X}|)} \sum_{i=1}^{\tau'(|\mathbf{X}|)} d_i - d_i^*, \tag{2.5}$$

where $\tau'(|\mathbf{X}|) = \min\{i|d_i = |\mathcal{S}(t)|\}$, $|\mathcal{S}(t)|$ is the length of speech input in time. d_i^* are the delays of an ideal policy, of which the straightforward adaption is $d_i^* = (i-1) \times |\mathcal{S}(t)|/|\hat{\mathbf{Y}}|$. However such adaptation is not robust for models that tend to stop hypothesis generation too early and generate translations that are too short. This is more likely to happen in simultaneous speech translation where a model can generate the end of sentence token too early, for example when there is a long pause even though the entire source input has not been consumed. Figure 2.8 illustrate this phenomenon. The red line in Figure 2.8 corresponds to the ideal policy defined in Ma et al. (2019a). We can see that when the model stops generating the translation, the lagging behind the ideal policy is negative. This is because the model stops reading any input after completing hypothesis generation. This kind of model can obtain relatively good latency-quality trade-offs as measured by AL (and BLEU), which does not reflect the reality. Thus, given the reference translation $\mathbf{Y}$, we

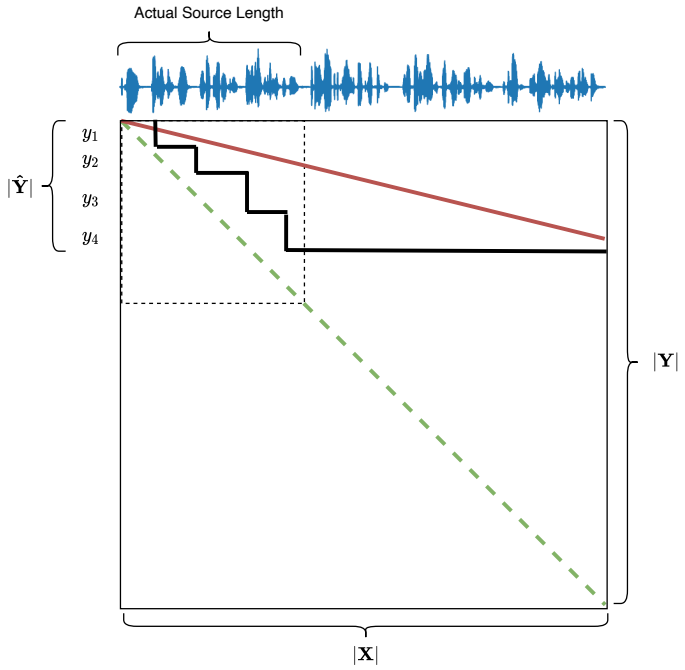

Figure 2.8: An example of original AL failed on early stop translation. Red (solid straight) line shows the ideal policy in (Ma et al., 2019a). Green (dotted straight) line depicts the modified ideal policy in this paper. Black (solid zigzag) line demonstrates the alignment between source and target.

define,

$$d_i^* = (i-1) \cdot \frac{|\mathcal{S}(t)|}{|\mathbf{Y}|} \tag{2.6}$$

to prevent this issue, i.e., it is assumed that the ideal policy generates the reference rather than the system hypothesis. The newly defined ideal policy is represented by the green line in Figure 2.8.

Although (Arivazhagan et al., 2019) proposed the differentiable average lagging, the original lagging itself is also differentiable. For instance, in a monotonic attention policy parameter-

ized by θ, the expected alignment between i-th target and j-th source can be represented as a differentiable function $\alpha_{ij}(\theta)$ [1], and further the expected delays can be represented as

$$d_i = \sum_{j=1}^{|\mathbf{X}|} \alpha_{ij}(\theta) \tag{2.7}$$

Considering Equation 2.4, the derivative is computed as follow

$$\frac{\partial AL}{\partial \theta} = \frac{1}{\tau(|\mathbf{X}|)} \sum_{i=1}^{\tau(|\mathbf{X}|)} \frac{\partial d_i}{\partial \theta} \tag{2.8}$$

$$= \frac{1}{\tau(|\mathbf{X}|)} \sum_{i=1}^{\tau(|\mathbf{X}|)} \sum_{j=1}^{|\mathbf{X}|} \frac{\partial}{\partial \theta} \alpha_{ij}(\theta) \tag{2.9}$$

where during the training time, both $\tau(|\mathbf{X}|)$ and $|\mathbf{X}|$ are known constants. Hence the AL is also differentiable during training.

Differentiable Average Lagging (DAL) (Arivazhagan et al., 2019) introduced a minimum delay of $1/\gamma$ after each operation. Unlike AL, it considers the tokens when $i > \tau(|\mathbf{X}|)$ (Cherry and Foster, 2019). It is defined in Equation 2.10:

$$\text{DAL} = \frac{1}{|\mathbf{Y}|} \sum_{i=1}^{|\mathbf{Y}|} d'_i - \frac{i-1}{\gamma}, \tag{2.10}$$

[1] Details are introduced in Chapter 4

where

$$d'_i = \begin{cases} d_i & i = 0 \\ \max(d_i, d'_{i-1} + \gamma) & i > 0 \end{cases}. \tag{2.11}$$

A minimum delay prevent DAL recovering from lagging once it has been incurred. For the speech task still uses Equation 2.10 and Equation 2.11 with a new γ defined as

$$\gamma_{\text{speech}} = |\mathbf{Y}| / \sum_{j=1}^{|\mathbf{X}|} T_s \tag{2.12}$$

2.3.2 Latency Evaluation Toolkit: SimulEval

To make the latency evaluation processes across different works consistent is non-trivial:

- the latency metric definitions are not precise enough with respect to text segmentation;
- the definitions are also not precise enough with respect to the speech segmentation, for example some models are evaluated on speech segments (Ren et al., 2020) while others are evaluated on time duration (Ansari et al., 2020; Anastasopoulos et al., 2021; Anastasopoulos et al., 2022);
- little prior work has released implementations of the decoding process and latency measurement.

The lack of clarity and consistency of the latency evaluation process makes it challenging to compare different works and prevents tracking the scientific progress of this field.

In order to provide researchers in the community with a standard, open and easy-to-use method to evaluate simultaneous speech and text translation systems, we introduce the toolkit SIMULEVAL in Ma et al. (2020b), an open source evaluation toolkit which automatically simulates a real-time scenario. SIMULEVAL creates a fully simultaneous translation environment and has been used for shared tasks such as the IWSLT 2020/2021/2022 shared task on simultaneous speech translation [2].

SIMULEVAL simulates a real-time scenario by setting up a server and a client. The server and client can be run separately or jointly, and are connected through RESTful APIs. An overview is shown in Figure 2.9. The server provides source input (text or audio) upon

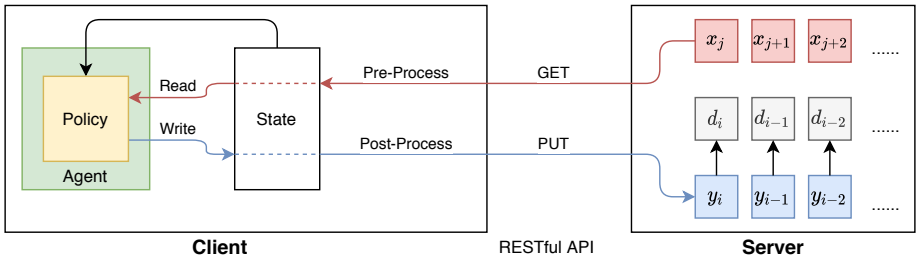

Figure 2.9: The architecture of SIMULEVAL. The client executes the policy and the server operates the evaluation. The server and client communicate through RESTful API to simulate the streaming setting.

the reading request from the client. It also receives predictions from the client and returns

[2] http://iwslt2020.ira.uka.de/doku.php?id=simultaneous_translation

different evaluation metrics when the translation process is complete. It has primarily four functions.

1. Read source and reference files.

2. Send source segments to the client upon a `READ` action.

3. Receive predicted segments from the client upon a `WRITE` action, and record the corresponding delays.

4. Run the evaluation on instances.

The evaluation process by the server on one instance is shown in Algorithm 1. Note that in line 18 in Algorithm 1, the server only runs sentence-level metrics. The server will collect $\mathbf{Y}$ and $\mathbf{D}$ for every instance in the evaluation corpus, and calculate corpus-level metrics after all hypotheses are complete.

The client contains two components — an agent and a state. The agent is a user-defined class that operates the policy and generates hypotheses for simultaneous translation, the latter provides functions such as pre-processing, post-processing and memorizing context. The purpose of this design is to make the user free from complicated setups, and focus on the policy. The client side algorithm is shown in Algorithm 2.

Algorithm 1 Server side algorithm. The functions of the server includes: 1. Provide source input (text or speech) when receiving the request from the client; 2. Record the translation and delay when receiving a prediction from the client, and compute latency at the end of evaluation.

Input: $\mathbf{X} = [x_1, ..., x_{|\mathbf{X}|}], \mathbf{Y} = [y_1, ..., y_{|\mathbf{Y}|}]$
Input: $\mathbf{Y} = [], \mathbf{D} = []$
Input: $i = 0, j = 0, y_0 = \text{BOS}, d_0 = 0$

while $\hat{y}_i \neq$ EOS **do**
    $r =$ await_request_from_client()
    if r.action == READ **then**
        if $j < |\mathbf{X}|$ **then**
            $j = j + 1$
            send_segment_to_client(x_j)
        else
            send_segment_to_client(EOS)
    else
        $i = i + 1$
        $\hat{y}_i = r$.segment
        $\hat{\mathbf{Y}} = \hat{\mathbf{Y}} + [\hat{y}_i]$
        Record current d_i
        $\mathbf{D} = \mathbf{D} + [d_i]$
return evaluate$(\hat{\mathbf{Y}}, \mathbf{Y}, \mathbf{D})$

Algorithm 2 Client side algorithm. The client is a wrapper over a simultaneous policy. When the policy decides to read, the client sends request to server for new input; when the policy decides to write, the client sends the prediction to the server.

Input: $\mathbf{X} = [], i = 0, j = 0, y_0 = \text{BOS}$, State, Agent
while $\hat{y}_i \neq \text{EOS}$ **do**
    action = Agent.policy(State)
    if action == READ **then**
        x = request_segment_from_server(segment_size)
        if x is not EOS **then**
            $j = j + 1$
            x_j = State.preprocess(x)
            States.update_source(x_j)
            continue
    $i = i + 1$
    $\hat{y}_i$ = Agent.predict(State)
    $\hat{y}_i$ = State.postprocess($\hat{y}_i$)
    States.update_target($\hat{y}_i$)
    send_segment_to_server($\hat{y}_i$)

2.4 Conclusion

In this chapter, we formalize the tasks of speech translation and simultaneous translation. Furthermore, we introduce the latency evaluation metrics for simultaneous speech translation. Finally, We propose an open source toolkit SimulEval (Ma et al., 2020b) for evaluation of simultaneous translation, which is published in EMNLP 2020 Demo track.

Chapter 3

Related Work

3.1 Speech Processing

The research of speech processing involves a wide range of topics, including the generation of the human speech, digital processing of speech signals and recognition and synthesis of speech. While speech processing covers many techniques, this section will only give a brief introduction on the background required for this thesis, including digitalizing speech signals, the feature extraction of speech signal, and automatic speech recognition, which is a task of transcribing speech to text.

Speech signals or waveforms, generated by the human speech production organs, are analog and stored digitally by sampling. The signals are represented as a sequence of

numbers, each of which measures the amplitude of the signals at a given time. Speech features, are usually frequency-based scalars or vectors extracted from small segments of speech signals. The popular choices of speech features include, Short-Time Fourier Transform (STFT) (Oliver, 1952), Mel-Frequency Cepstrum Coefficients (MFCC) (Davis and Mermelstein, 1980), Linear Prediction Coding (LPC) (Portnoff, 1981), Perceptual Linear Prediction (PLP) (Hermansky, 1990), and Log Mel-Filter Bank (LMFB) (Murthy et al., 1999). The detailed process can be found in Section 2.2

Automatic speech recognition (ASR) is the task to convert continuous speech to the corresponding text. The first practical large vocabulary and speaker independent ASR system Sphinx was introduced by Lee et al. (1990) and Huang et al. (1993), which includes a hidden Markov model (HMM) (Baum and Petrie, 1966) acoustic model and n-gram language model. More recently, the neural network based models, including hybrid model (Graves et al., 2006) and end-to-end model (Graves and Jaitly, 2014), have become the start-of-the-art architecture. Nowadays, high quality ASR systems can be built from several open source projects (Povey et al., 2011; Hannun et al., 2014; Collobert et al., 2016; Watanabe et al., 2018; Kuchaiev et al., 2018; Wang et al., 2020a).

3.2 Offline Speech Translation

Speech translation is the task to translate speech in one language to either text or speech in another language. In following sections, for simplicity, ST represents general speech translation, while S2T stands for speech-to-text translation and S2S stands for speech-to-speech translation. ST has been an active research topic over decades since 1990s, and the architecture of the ST systems have been changing dramatically over the time, especially after the great success of deep neural networks. In this section, a brief introduction of speech translation will be given, with the two different approaches — cascaded and end-to-end.

3.2.1 Cascaded Approach

A cascaded ST system usually contains a pipeline of multiple submodules. More specifically, an automatic speech recognition (ASR) module serves as the front end of the system that processes the speech signals to linguistics representations, usually text or grammar structures; a text-to-text (T2T) machine translation then translates the output of ASR module to another language; an additional text-to-speech synthesis module is applied on the text translation to generate the final speech output for a S2S system. The cascaded approach was explored first by researchers due to its advantage of building the ST system on top of existing submodules.

The earliest attempts to build a fully functional ST system even happened before strong statistical MT systems arrived. Many of the early ST systems included a rule-based or knowledge-based MT module. Tomita et al. (1990) proposed the Universal Parser Architecture for spoken language translation, which passes the output of a speech recognition system to a knowledge-based system to generate the translation. Morimoto et al. (1990) introduced SL-TRANS, which is a three-stage system that integrated speech recognition and language processing. Kitano (1991) introduced ΦDM-Dialog, with the concept of "processing flow" with a hidden markov based (HMM) ASR system, a knowledge-based machine translation system, and a voice generation system. Wahlster (1993) introduced Verbmobil system which is designed to translate face-to-face dialogs. Wasyliw and Clarke (1994) introduced speech translation system for pilot air traffic control (ATC) communication. Rayner et al. (1995) used limited-domain data to build a air travel planning speech translation system. JANUS (Jain et al., 1991; Lavie et al., 1996; Woszczcyna et al., 1998) is also an early effort on speech translation, which is capable of translating spontaneous speech in limited domain. Lavie et al. (2002) introduced NESPOLE!, which is a S2S machine translation system designed with real-world settings of common users involved in e-commerce applications. Konuma et al. (2002) introduced an experimental S2S translation hardware platform, which outputs 2 to 10 times faster than a conventional translation device. Shimohata et al. (2003) proposed an S2S translation system with example based translation module. Dillinger and

Seligman (2004) introduced a S2S translation system to help Spanish-speaking patients to communicate with English-speaking doctors, nurses, and other health-care staff. Gu and Gao (2004) proposed the concept based S2S system in which the feature is selected based on maximum entropy. Bouillon et al. (2005) proposed MedSLT, which is an open source platform for developing limited-domain medical speech translation systems.

The break-through of statistical phrase-based MT system (Koehn et al., 2003) also provides more possibilities for ST. Bach et al. (2007) proposed a portable two-way S2S translation system with phrase-based MT system. Pérez et al. (2007) discussed the benefit from additional knowledge sources such as semantically motivated segmentation or statistical categorization. Kao et al. (2008) report a rapid development of S2S system with phrase-based MT for low resource languages. Kolss et al. (2008) build a simultaneous S2S system for German-English lecture translation. Lim et al. (2010) introduced a real-time language identification and recognition system, for S2ST translation. Raybaud et al. (2011) proposed an integrated S2T translation system for news translation. Maergner et al. (2011) proposed a method for vocabulary selection for simultaneous speech translation for lectures. Ahmed et al. (2012) proposed a hierarchical phrased-based statistical machine translation module for speech translation. Cho et al. (2012) addressed the issue on segmentation and punctuation prediction for ST. Kano et al. (2012) proposed a method to use paralinguistic information for speech translation. Khadivi and Vakil (2012) proposed a speech-enabled interactive-

predictive computer-assisted translation system. Prasad et al. (2012) introduced a ST system with capability of active error detection and resolution. Kumar et al. (2015) proposed error-tolerant S2S system for Arabic-English translation. Microsoft introduced a near real-time S2T system, Skype Translator (Lewis, 2015), for translation in online communication. Do et al. (2015) proposed a method which improve the emphasis with pause prediction for S2S systems.

One of the disadvantages of the cascaded approach is that the errors are difficult to recover from the down streaming task. For instance, Ruiz and Federico (2014) pointed out that " the linguistic properties of ASR errors have ramifications on SMT quality in speech translation". Thus, more coupled systems are also introduced. Ney (1999) first proposed the concept coupling speech recognition and translation with local averaging approximation and the monotone alignments. Zhang et al. (2005) proposed a decoding algorithm which runs on speech recognition word lattice instead of output text. Zhang et al. (2005) considered using multiple recognition hypotheses to improve the translation quality. Alabau et al. (2007) included word posterior probabilities for S2T decoding. Patry and Langlais (2008) proposed an open source decoder MISTRAL for speech translation on word lattice. Pérez et al. (2010) considered the potential scope of a fully-integrated architecture for speech translation. Peitz et al. (2012) proposed a method to train the S2T model transcribed text. Ananthakrishnan et al. (2013) tried to alleviating the impact of unrecoverable ASR errors

by locally-derived penalties on language model. Cho et al. (2013) introduced a conditional random field (CRF)-based speech disfluency detection system to improve the performance.

3.2.2 End-to-End Approach

Recently, a growing amount of work has focused on the neural end-to-end approach, with the expectation that a less complex system has the capability of reducing compounding errors between the subsystems and improving the overall efficiency with the end-to-end or direct translation. We will first discussed speech-to-text and then speech-to-speech translation.

When the neural machine translation became the state-of-the-art approach for text translation task (Anastasopoulos et al., 2022), people start to explore adapting the same architecture to speech-to-text translation task, which soon starts to show competitive results compared with the cascade approaches. Duong et al. (2016) first proposed an attention-based sequence-to-sequence structure for the task. They use an recurrent neural networks (RNN) based encoder-decoder architecture on speech-to-text translation task. They also introduced the stacked and pyramidal RNN encoder to reduce the sequence length, because the speech features are usually much longer than the text output. Despite the novelty of Duong et al. (2016)'s work, there is huge downgrade of translation quality compared with cascaded approaches. Some later work (Berard et al., 2016; Weiss et al., 2017; Bansal et al.,

2018; Bérard et al., 2018) added convolutional layers before the RNN sequence-to-sequence structure. The conventional layers help to better encode the speech features and significantly improve the end-to-end model performance, which are then comparable with cascaded models. With the great success of Transformer (Vaswani et al., 2017) on text translation task, Di Gangi et al. (2019b) and Inaguma et al. (2020) applied the Transformer architecture to the speech translation task, which further improves both translation quality and training speed.

Similar to many deep-learning models, the biggest challenge for end-to-end the speech translation model is data scarcity. Multitask-learning and pre-training (Weiss et al., 2017; Bérard et al., 2018; Bansal et al., 2018; Stoian et al., 2020; Wang et al., 2020b), data augmentation and self-training (Jia et al., 2019a; Salesky et al., 2019; Pino et al., 2019; Pino et al., 2020) are widely used for the task. Furthermore, other techniques in machine learning, such as knowledge distillation (Liu et al., 2019) and meta-learning (Indurthi et al., 2020), are also introduced to improve the translation quality.

A general architecture of a state-of-the-art offline end-to-end speech-to-text translation model is shown as Figure 3.1. Different from text-to-text encoder-decoder models, of which the inputs are usually word representations, the inputs of the speech models are usually features extracted from a fixed size sliding window. As introduced in Section 3.1, the speech waveforms are firstly converted to a sequence of speech features, which will be fed into the

speech encoder. The encoder is a stack of two modules, convolutional layers and recurrent layers, The convolutional layers learns local speech feature information, and often reduce the input length with pooling. The recurrent layers, which can be LSTM or self-attention based architecture, capture the sequential information. Finally, the decoder is similar to the text translation decoder. which takes a weighted sum of encoder states and the previous decoder state at each step, and generate the prediction of next target token.

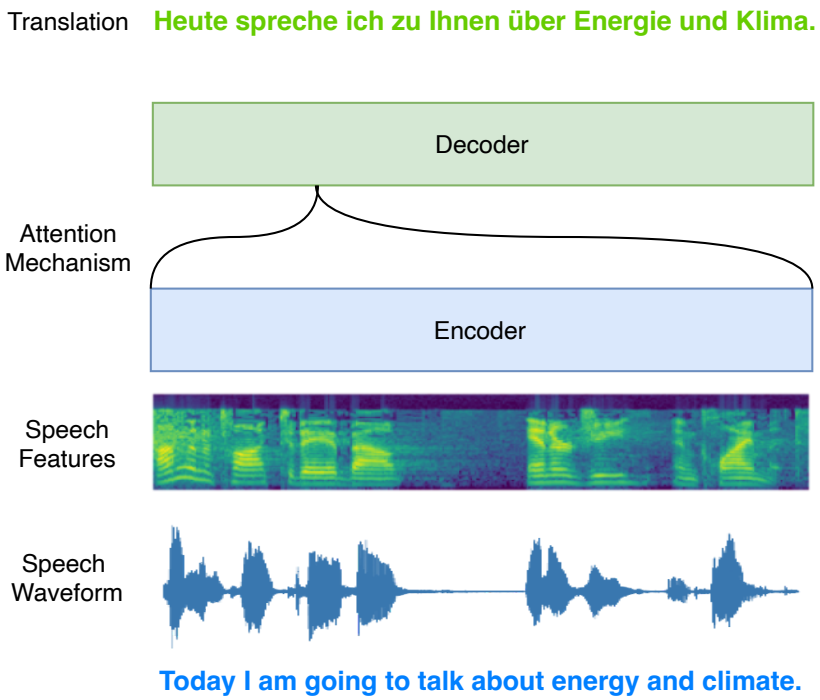

Figure 3.1: The general architecture of offline end-to-end speech-to-text translation model. The speech signals are converted into speech features which are then processed by a deep learning encoder. The decoder produces output text based on the encoder representations, with a soft attention mechanism.

With the respect of end-to-end S2S translation, Jia et al. (2019b) explored the very

first end-to-end speech-to-speech translation model, Translatotron. Translatotron is able to directly generate speech features without relying on an intermediate text representation. The follow-up work Translatotron 2 (Jia et al., 2021) used an auxiliary target phoneme decoder to address the over-generation issue, where the model tends to generates long output. Kano et al. (2021) proposed a scheme where the end-to-end model is stacked by separately pretrained ASR, MT and TTS submodules. Lee et al. (2022) proposed a direct model which predicts self-supervised discrete representations of the target speech instead of features. Unlike the speech-to-text task, a gap still exists between the end-to-end and the cascade approaches.

We will introduce the details of Lee et al. (2022)'s speech-to-unit model, which we use in Chapter 7. In the speech-to-unit model, the target speech units are prepared beforehand. Unsupervised continuous representations for every 20ms frame are learned on the target speech corpus by a HuBERT model (Hsu et al., 2021). Then the k-means algorithm is applied to the representations to generate K cluster centroids. For each window, its closest centroid index is used as discrete label. We denote the discrete sequence as $\mathbf{Z} = z_1, ...z_L$. Given the discrete units, Lee et al. (2022) build a transformer-based (Vaswani et al., 2017) model, of which inputs are speech feature and output are discrete units. In the encoder, a stack of 1D-convolutional layers serve as downsampler for the speech input. Since the target sequence is discrete, the S2U model can be trained with cross-entropy loss. Finally, the

vocoder converts the discrete units to speech signal. Lee et al. (2022) use a modified version of the HiFi-GAN neural vocoder (Polyak et al., 2021) for unit-to-waveform conversion.

3.3 Simultaneous Text-to-text Translation

The core of simultaneous translation is to develop a policy, which could determine the optimal timing for read and write actions. The difficulty of find optimal timing depending on the similarity of languages. Or more specifically the reordering between the source and target languages. For instance, we show the amount of the of reordering between most common 11 European Languages, Figure 3.2. We can see that building an English to Spanish system is far more difficult than English to Danish. Such observation can also be seen in the IWSLT shared task of simultaneous speech translation (Anastasopoulos et al., 2022; Anastasopoulos et al., 2021). English-Japanese systems tend to have greater BLEU score drop comparing with English-German given the same latency.

In this section we will introduce the modeling of simultaneous policy, including the concept of READ and WRITE action in the policy, and two common policices: monotonic attention (Raffel et al., 2017) and wait-k policy (Ma et al., 2019a).

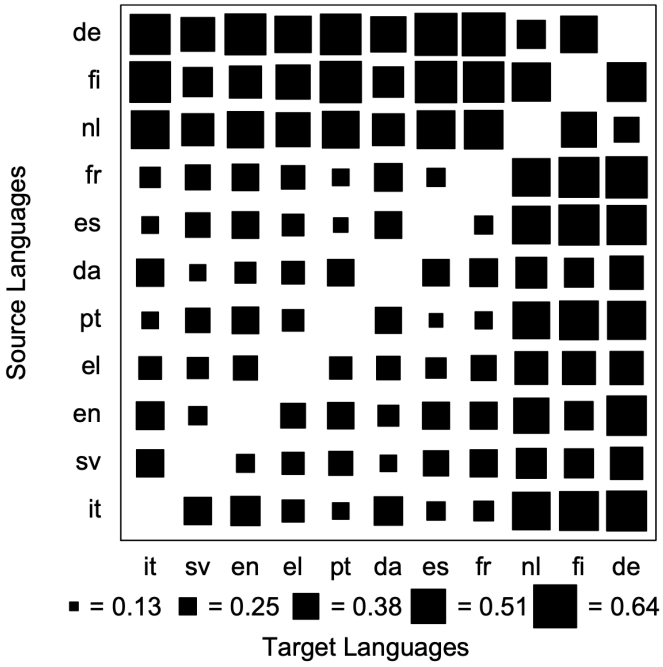

Figure 3.2: The reordering amount measured by RQuantity from Birch et al. (2008)

3.3.1 Simultaneous policies

Simultaneous translation means starting the translation before finishing reading the source input. While most of work on simultaneous translation focuses on the text translation, it can be generalized and applied to speech translation. Figure 3.3 is an illustration of a generalized simultaneous translation process. At a given time when input length is j and output length is i, the simultaneous policy takes current context (source and target sequence) as inputs and generate a Bernoulli distribution parameterized by $p_{i,j}$ on two actions, READ and WRITE. READ means that the model pauses the generation and reads one input unit

from the source side, while WRITE stops taking new inputs and generates a new prediction. Specifically, in an offline system, $p_{i,j} = 0$ for any i, j, and $p_{i,j} = 1$ if j equals to the length of the source input. More formally, a general simultaneous translation algorithm is described as Algorithm 3.

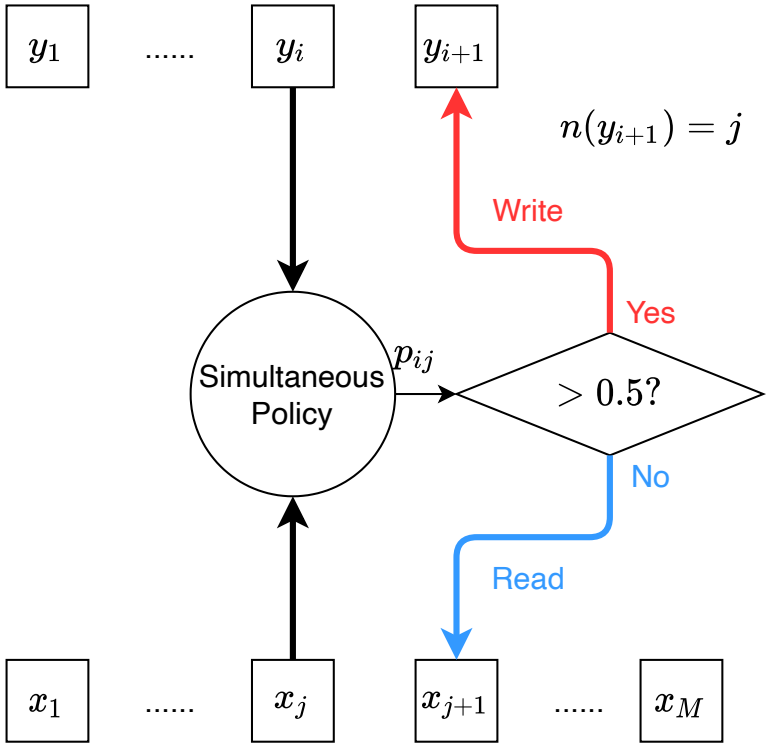

Figure 3.3: Generalized simultaneous decoding process.

The simultaneous policies fall into three categories. The first one is the pre-defined context-free rule-based policies. Cho and Esipova (2016) proposed a Wait-If-* policy to enable an offline model to decode simultaneously. Dalvi et al. (2018) modified the Wait-If-* policy (Cho and Esipova, 2016) to enable decoder for consecutive prediction. Ma et al.

(2019a) proposed a Wait-k policy where the model first reads k input, then reads and writes alternatively. In the second category, a learnable flexible policy with an agent is introduced and reinforcement learning is applied. Grissom II et al. (2014) introduced an agent based on Markov chain to phrase-based machine translation models for simultaneous machine translation, in which reinforcement learning is used to learn the read-write policy based on states. Gu et al. (2017) introduced an agent which learns to make decisions on when to translate from the interaction with a pre-trained offline neural machine translation model. Luo et al. (2017) used continuous rewards policy gradient for online alignments for speech recognition. Lawson et al. (2018) proposed a hard alignment with variational inference for online decoding. Alinejad et al. (2018) proposed a new operation PREDICT which predicts future source tokens. Zheng et al. (2019b) introduced a restricted dynamic oracle and restricted imitation learning for simultaneous translation. Zheng et al. (2019a) trained the agent with an action sequence from labels that are generated based on the rank of the gold target word given partial input. Models from the last category leverage monotonic attention and replace the Softmax attention with a closed form expected attention calculated from a stepwise Bernoulli selection probability. Raffel et al. (2017) first introduced the concept of monotonic attention for online linear time decoding, where the attention only attends to one encoder state at a time. Chiu* and Raffel* (2018) extended (Raffel et al., 2017)'s work to enable the model to attend to a chunk of encoder states. Arivazhagan et al. (2019) also made

use of the monotonic attention but introduced an infinite lookback to improve the translation quality. Ma et al. (2019b) adapted monotonic attention to transformer architecture with multihead attention.

Algorithm 3 Generalized simultaneous translation algorithm

Input: Simultaneous policy $\mathcal{P}$, model $\mathcal{M}$
Input: Input sequence $\mathbf{X} = [x_1, ...x_M]$, Predicted sequence $\mathbf{Y}$.
Input: $i = 1, j = 1, y_1 = \text{StartOfTranslation}$

while $y_i \neq \text{EndOfTranslation}$ **do**
    $\mathbf{X}_{1:j} = [x_1, ...x_j], \mathbf{Y}_{1:i} = [y_1, ...y_i]$ # Current context.
    $p_{ij} = \mathcal{P}(\mathbf{X}_{1:j}, \mathbf{Y}_{1:i})$ # Stepwise probability.
    if $p_{ik} > 0.5$ **or** $j == M$ **then**
        $y_{i+1} = \mathcal{M}(\mathbf{Y}_{1:i}, \mathbf{X}_{1:j})$
        $i = i + 1$ # Write a target prediction.
    else
        $j = j + 1$ # Read a source input.

3.3.2 Monotonic Attention

Since the monotonic attention approaches have been the state-of-the-art for the simultaneous translation task, we now introduce more details on this method. The hard monotonic attention mechanism (Raffel et al., 2017) was first introduced in order to achieve online linear time decoding for RNN-based encoder-decoder models. We denote the input sequence as $\mathbf{X} = \{x_1, ..., x_T\}$, and the corresponding encoder states as $\mathbf{M} = \{m_1, ..., m_T\}$, with T being the length of the source sequence. The model generates a target sequence $\mathbf{Y} = \{y_1, ..., y_U\}$ with U being the length of the target sequence. At the i-th decoding step,

the decoder only attends to one encoder state m_{t_i} with $t_i = j$. When generating a new target token y_i, the decoder chooses whether to move one step forward or to stay at the current position based on a Bernoulli selection probability $p_{i,j}$, so that $t_i \geq t_{i-1}$. Denoting the decoder state at the i-th position, starting from $j = t_{i-1}, t_{i-1} + 1, t_{i-1} + 2, ...$, this process can be calculated as follows:

$$e_{i,j} = \text{MonotonicEnergy}(s_{i-1}, m_j) \tag{3.1}$$

$$p_{i,j} = \text{Sigmoid}\,(e_{i,j}) \tag{3.2}$$

$$z_{i,j} \sim \text{Bernoulli}(p_{i,j}) \tag{3.3}$$

When $z_{i,j} = 1$, we set $t_i = j$ and start generating a target token y_i; otherwise, we set $t_i = j + 1$ and repeat the process. During training, an expected alignment $\boldsymbol{\alpha}$ is introduced to replace the softmax attention. Furthermore, to encourage discreteness, Raffel et al. (2017) added a zero mean, unit variance pre-sigmoid noise to $e_{i,j}$.

Different from a reinforcement learning approach, monotonic attention utilizes an closed form calculation of expectation alignment α_{ij}, given the stepwise probability p_{ij} shown in

line 3 of Algorithm 3. α_{ij} can be calculated in a recurrent manner, shown in Equation 3.4:

$$\begin{aligned} \alpha_{i,j} &= p_{i,j} \sum_{k=1}^{j} \left(\alpha_{i-1,k} \prod_{l=k}^{j-1} (1 - p_{i,l}) \right) \\ &= p_{i,j} \left((1 - p_{i,j-1}) \frac{\alpha_{i,j-1}}{p_{i,j-1}} + \alpha_{i-1,j} \right) \end{aligned} \tag{3.4}$$

Raffel et al. (2017) also introduce a closed-form parallel solution for the recurrence relation in Equation 3.5:

$$\alpha_{i,:} = p_{i,:} \text{cumprod}(1 - p_{i,:}) \text{cumsum} \left(\frac{\alpha_{i-1,:}}{\text{cumprod}(1 - p_{i,:})} \right) \tag{3.5}$$

where $\text{cumprod}(\boldsymbol{x}) = [1, x_1, x_1 x_2, ..., \prod_{i=1}^{|\boldsymbol{x}|-1} x_i]$ and $\text{cumsum}(\boldsymbol{x}) = [x_1, x_1 + x_2, ..., \sum_{i=1}^{|\boldsymbol{x}|} x_i]$. In practice, the denominator in Equation 3.5 is clamped into a range of $[\epsilon, 1]$ to avoid numerical instabilities introduced by `cumprod`. The expected alignment α then replaces the Softmax attention during the training time.

Although the monotonic attention mechanism achieves online linear time decoding, the decoder can only attend to one encoder state. This limitation can diminish translation quality as there may be insufficient information for reordering. Moreover, the model lacks a mechanism to adjust latency based on different requirements at decoding time. To address these issues, Chiu* and Raffel* (2018) introduced Monotonic Chunkwise Attention (MoChA), which allows the decoder to apply soft attention on a fixed-length subsequence

of encoder states. Alternatively, Arivazhagan et al. (2019) introduced Monotonic Infinite Lookback Attention (MILk) which allows the decoder to access encoder states from the beginning of the source sequence. Instead of the expected alignment, MILk applied the expected soft attention for training shown in Equation 3.6:

$$\beta_{ij} = \sum_{k=j}^{|\mathbf{X}|} \left(\frac{\alpha_{ik} \exp(u_{ij})}{\sum_{l=1}^{k} \exp(u_{il})} \right) \tag{3.6}$$

where u_{ij} is the soft-attention energy for x_j and y_i. Note that the wait-k policy (Ma et al., 2019a) is a special case of MILk where $p_{ij} = 1$ if $i - j < k$ otherwise $p_{ij} = 0$. Arivazhagan et al. (2019) also introduced the latency augmented training for latency control. Given the expected alignment a_{ij}, the expected delays of all the prediction can be calculated as

$$\hat{n}(y_i) = \sum_{j=1}^{|\mathbf{X}|} j \alpha_{ij} \tag{3.7}$$

A latency loss $\mathcal{C}([\hat{n}(y_1), ..., \hat{n}(y_N)])$ is then added to the total loss function, where $\mathcal{C}$ is a latency metric. By adjusting the latency loss weight, one can control the latency of the system.

3.3.3 Wait-k policy

Wait-k, or prefix-to-prefix, policy is a simple and effective baseline for most research in simultaneous translation. It is a rule based policy, which only consider the dynamic length of input and output sequences. First, k source words are read by the model, which then translates concurrently with the rest of source sentence. More specifically, read one token on the source side, and write one token on the target side. Mathematically, the step-wise probability p_{ij} and monotonic alignment α can be expressed as Equation 3.8 and Equation 3.8

$$p_{ij} = \begin{cases} 1, & j - i < k \\ 0, & \text{otherwise} \end{cases} \tag{3.8}$$

$$\alpha_{ij} = \begin{cases} 1, & j - i = k \text{ or } i = |\mathbf{Y}| \\ 0, & \text{otherwise} \end{cases} \tag{3.9}$$

Chapter 4

Monotonic Multihead Attention

4.1 Introduction

Different from an offline machine translation model, in which the decoder starts translation after the encoder reads all the source input sentence, a simultaneous translation model starts the translation with partial input, and then alternates between reading the input and writing the output using either a fixed or learned policy. More specifically, a policy in simultaneous translation controls two actions: read and write. The read action indicates the model waiting for more input context, while the write action indicates the model generating predictions.

Monotonic attention mechanisms has recently become the state-of-the-art approach for

this task. It falls into the flexible policy category, in which the policies are automatically learned from data. As introduced in Section 3.3.2, recent exploration on monotonic attention for simultaneous translation includes several variants: hard monotonic attention (Raffel et al., 2017), monotonic chunkwise attention (MoChA) (Chiu* and Raffel*, 2018) and monotonic infinite lookback attention (MILk) (Arivazhagan et al., 2019). MILk in particular has shown better quality/latency trade-offs than fixed policy approaches, such as wait-k (Ma et al., 2019a) or wait-if-* (Cho and Esipova, 2016) policies. MILk also outperforms hard monotonic attention and MoChA, while the other two monotonic attention mechanisms only consider a fixed reading window during inference. MILk computes a softmax attention over all previous encoder states, which may be the key to its improved latency-quality trade-offs. The monotonic attention approaches provide a closed-form expression for the expected alignment between source and target tokens. Therefore, one of the advantages of monotonic attention is that it does not rely on a reinforcement learning component (Gu et al., 2017). However, monotonic attention-based models, including the state-of-the-art MILk, were all built on top of RNN-based models. RNN-based models have been outperformed by the recent state-of-the-art Transformer model (Vaswani et al., 2017), which features multiple encoder-decoder attention layers and multiple attention heads at each layer.

In this chapter, We will present monotonic multihead attention (MMA), which combines the high translation quality from multilayer multihead attention and low latency from mono-

tonic attention. This is one of the first flexible simultaneous policies that has been proposed to Transformer-based model. The chapter will only discuss on text-to-text translation, with a main focus on the policy design. We proposed two variants of the model, Hard MMA (MMA-H) and Infinite Lookback MMA (MMA-IL). MMA-H is designed with streaming systems in mind where the attention span must be limited. MMA-IL emphasizes the quality of the translation system. We also propose two novel latency regularization methods. The first encourages the model to be faster by directly minimizing the average latency. The second encourages the attention heads to maintain similar positions, preventing the latency from being dominated by a single or a few heads. The main contributions from this chapter include:

1. A novel monotonic attention mechanism, monotonic multihead attention, which enables the Transformer model to perform online decoding. This model leverages the power of the Transformer and the efficiency of monotonic attention.

2. Better latency/quality tradeoffs compared to the MILk model, the previous state-of-the-art, on two standard translation benchmarks, IWSLT15 English-Vietnamese (En-Vi) and WMT15 German-English (De-En).

3. Analyses on how our model is able to control the attention span and the relationship between the speed of a head and the layer it belongs to. We motivate the design of our model with an ablation study on the number of decoder layers and the number of

decoder heads.

4.2 Methodology

4.2.1 Monotonic Multihead Attention

Previous monotonic attention approaches are based on RNN encoder-decoder models with a single attention head and have not explored the power of the Transformer model[1]. The Transformer architecture (Vaswani et al., 2017) has recently become the state-of-the-art for machine translation. An important feature of the Transformer is the use of a separate multihead attention module at each layer. Thus, we propose a new approach, Monotonic Multihead Attention (MMA), which combines the expressive power of multihead attention and the low latency of monotonic attention.

As introduced in Vaswani et al. (2017), multihead attention allows each decoder layer to have multiple heads, where each head can compute a different attention distribution. Given queries Q, keys K and values V, multihead attention MultiHead(Q, K, V) is defined in

[1]MILk was based on a strengthened RNN-based model called RNMT+. The original RNMT+ model uses multihead attention, computes attention only once, and then concatenates that single attention layer to the output of each decoder layer block. However, the RNMT+ model used for MILk in (Arivazhagan et al., 2019) only uses a single head.

Equation 4.1.

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, ..., \text{head}_H)W^O \\ \text{where } \text{head}_h &= \text{Attention}\left(QW_h^Q, KW_h^K, VW_h^V, \right) \end{aligned} \tag{4.1}$$

The attention function is the scaled dot-product, defined in Equation 4.2:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4.2}$$

There are three applications of multihead attention in the Transformer model:

1. The **Encoder** contains self-attention layers where all of the queries, keys and values come from previous layers.

2. The **Decoder** contains self-attention layers that allow each position in the decoder to attend to all positions in the decoder up to and including that position.

3. The **Encoder-Decoder attention** contains multihead attention layers where queries come from the previous decoder layer and the keys and values come from the output of the encoder. Every decoder layer has its own encoder-decoder attention.

For MMA, we assign each head to operate as separate monotonic attention in encoder-decoder attention.

For a transformer with L decoder layers and H attention heads per layer, given the encoder state m_j at j-th step and the decoder state s_{i-1} at $i-1$-th step, we define the selection process of the h-th head encoder-decoder attention in the l-th decoder layer as

$$e_{i,j}^{l,h} = \left(\frac{m_j W_{l,h}^K (s_{i-1} W_{l,h}^Q)^T}{\sqrt{d_k}} \right)_{i,j} \tag{4.3}$$

$$p_{i,j}^{l,h} = \text{Sigmoid}(e_{i,j}^{l,h}) \tag{4.4}$$

$$z_{i,j}^{l,h} \sim \text{Bernoulli}(p_{i,j}^{l,h}) \tag{4.5}$$

where $W_{l,h}$ is the input projection matrix, d_k is the dimension of the attention head. We make the selection process independent for each head in each layer. We then investigate two types of MMA: MMA-H(hard) and MMA-IL(infinite lookback). For MMA-H, we use Equation 3.4 in order to calculate the expected alignment for each layer each head, given $p_{i,j}^{l,h}$. For MMA-IL, we calculate the softmax energy for each head as follows:

$$u_{i,j}^{l,h} = \text{SoftEnergy} = \left(\frac{m_j \hat{W}_{l,h}^K (s_{i-1} \hat{W}_{l,h}^Q)^T}{\sqrt{d_k}} \right)_{i,j} \tag{4.6}$$

and then use Equation 3.6 to calculate the expected attention. Notice that Equation 4.6 has similar function as Equation 4.3 but different parameters. Each attention head in MMA-H attends to one encoder state. On the other hand, each attention head in MMA-IL can

attend to all previous encoder states. Thus, MMA-IL allows the model to leverage more information for translation, but MMA-H may be better suited for streaming systems with stricter efficiency requirements. Finally, our models use unidirectional encoders: the encoder self-attention can only attend to previous states, which is also required for simultaneous translation.

At inference time, our decoding strategy is shown in Algorithm 4. For each l, h, at decoding step i, we apply the sampling processes discussed in Section 3.3.2 individually and set the encoder step at $t_i^{l,h}$. Then a hard alignment or partial softmax attention from encoder states, shown in Equation 4.7, will be retrieved to feed into the decoder to generate the i-th token. The model will write a new target token only after all the attentions have decided to write. In other words, the heads that have decided to write must wait until the others have finished reading.

$$
\begin{aligned}
c_i^l &= \text{Concat}(c_i^{l,1}, c_i^{l,2}, ..., c_i^{l,H}) \\
\text{where } c_i^{l,h} &= f_{\text{context}}(\boldsymbol{h}, t_i^{l,h}) = \begin{cases} m_{t_i^{l,h}} & \text{MMA-H} \\ \sum_{j=1}^{t_i^{l,h}} \frac{\exp\left(u_{i,j}^{l,h}\right)}{\sum_{j=1}^{t_i^{l,h}} \exp\left(u_{i,j}^{l,h}\right)} m_j & \text{MMA-IL} \end{cases}
\end{aligned} \tag{4.7}
$$

Figure 4.1 illustrates a comparison between our model and the monotonic model with one attention head.

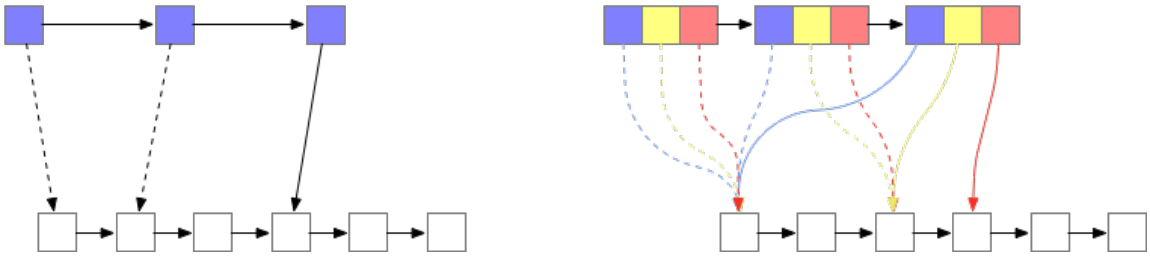

Figure 4.1: Monotonic Attention (Left) versus Monotonic Multihead Attention (Right). The hard monotonic attention only attends to one encoder state. Monotonic multihead attention is able to attend multiple encoder states at the same time.

Compared with the monotonic model, the MMA model is able to set attention to different positions so that it can still attend to previous states while reading each new token. Each head can adjust its speed on-the-fly. Some heads read new inputs, while the others can stay in the past to retain the source history information. Even with the hard alignment variant (MMA-H), the model is still able to preserve the history information by setting heads to past states. In contrast, the hard monotonic model, which only has one head, loses the previous information at the attention layer.

4.2.2 Latency Control

Effective simultaneous machine translation must balance quality and latency. At a high level, latency measures how many source tokens the model has read until a translation is generated. The model we have introduced in Section 4.2.1 is not able to control latency on its own. While MMA allows simultaneous translation by having a read or write schedule

Algorithm 4 MMA monotonic decoding. Because each head is independent, we compute line 3 to 16 in parallel

Input: $\boldsymbol{x}$: source tokens.
Input: $\boldsymbol{h}$: encoder states.
Input: y_0: Start of the target sequence.
Input: L: Number of decoder layers.
Input: H: Number of attention heads in decoder each layer.
Input: $i = 1, j = 1, t_0^{l,h} = 1$

1: **while** $y_{i-1} \neq$ EndOfSequence **do**
2: $\quad t_{\text{max}} = 1$
3: $\quad \boldsymbol{h} =$ empty sequence
4: $\quad$ **for** $l \leftarrow 1$ to L **do**
5: $\quad\quad$ **for** $h \leftarrow 1$ to H **do**
6: $\quad\quad\quad$ **for** $j \leftarrow t_{i-1}^{l,h}$ to $|\boldsymbol{x}|$ **do**
7: $\quad\quad\quad\quad p_{i,j}^{l,h} = \text{Sigmoid}\left(\text{MonotonicEnergy}(s_{i-1,m_j})\right)$
8: $\quad\quad\quad\quad$ **if** $p_{i,j}^{l,h} > 0.5$ **then**
9: $\quad\quad\quad\quad\quad t_i^{l,h} = j$
10: $\quad\quad\quad\quad\quad c_i^{l,h} = f_{\text{context}}(\boldsymbol{h}, t_i^{l,h})$
11: $\quad\quad\quad\quad\quad$ **Break**
12: $\quad\quad\quad\quad$ **else**
13: $\quad\quad\quad\quad\quad$ **if** $j > t_{\text{max}}$ **then**
14: $\quad\quad\quad\quad\quad\quad$ Read token x_j
15: $\quad\quad\quad\quad\quad\quad$ Calculate state h_j and append to $\boldsymbol{h}$
16: $\quad\quad\quad\quad\quad\quad t_{\text{max}} = j$
17: $\quad\quad\quad c_i^l = \text{Concat}(c_i^{l,1}, c_i^{l,2}, ..., c_i^{l,H})$
18: $\quad\quad s_i^l = \text{DecoderLayer}^l(s_{1:i-1}^l, s_{1:i-1}^{l-1}, c_i^l)$
19: $\quad y_i = \text{Output}(s_i^L)$
20: $\quad i = i + 1$

for each head, the overall latency is determined by the fastest head, i.e. the head that reads the most. It is possible that a head always reads new input without producing output, which would result in the maximum possible latency. Note that the attention behaviors in MMA-H and MMA-IL can be different. In MMA-IL, a head reaching the end of the sentence will provide the model with maximum information about the source sentence. On the other hand, in the case of MMA-H, reaching the end of sentence for a head only gives a hard alignment to the end-of-sentence token, which provides very little information to the decoder. Furthermore, it is possible that an MMA-H attention head stays at the beginning of sentence without moving forward. Such a head would not cause latency issues but would degrade the model quality since the decoder would not have any information about the input. In addition, this behavior is not suited for streaming systems.

To address these issues, we introduce two latency control methods. The first one is *weighted* average latency, shown in Equation 4.8:

$$d_i^W = \frac{\exp(d_i^{l,h})}{\sum_{l=1}^{L}\sum_{h=1}^{H}\exp(d_i^{l,h})} d_i^{l,h} \tag{4.8}$$

where $d_i^{l,h} = \sum_{j=1}^{|\boldsymbol{x}|} j\alpha_{i,j}$. Then we calculate the average latency loss with a differentiable latency metric $\mathcal{C}$.

$$L_{avg} = \mathcal{C}\left(\boldsymbol{d}^W\right) \tag{4.9}$$

Like Arivazhagan et al. (2019), we use the Differentiable Average Lagging. It is important to note that, unlike the original latency augmented training in (Arivazhagan et al., 2019), Section 4.2.2 is not the expected latency metric given $\mathcal{C}$, but weighted average $\mathcal{C}$ on all the attentions. The real expected latency is $\hat{\boldsymbol{d}} = \max_{l,h} \left(\boldsymbol{d}^{l,h} \right)$ instead of $\bar{\boldsymbol{d}}$, but using this directly would only affect the speed of the fastest head. Section 4.2.2 can control every head in a way that the faster heads will be automatically assigned to larger weights and slower heads will also be moderately regularized. For MMA-H models, we found that the latency of are mainly due to outliers that skip almost every token. The weighted average latency loss is not sufficient to control the outliers. The reason use weighted average over maximum is because the maximum function is too sparse to regularize all the outliers. We therefore introduce the head divergence loss, the average variance of expected delays at each step, defined in Section 4.2.2:

$$L_{var} = \frac{1}{LH} \sum_{l=1}^{L} \sum_{h=1}^{H} \left(d_i^{l,h} - \bar{d}_i \right)^2 \tag{4.10}$$

where $\bar{d}_i = \frac{1}{LH} \sum_{i}^{|\mathbf{Y}|} d_i$ The final objective function is presented in Equation 4.11:

$$L(\theta) = -\log(\boldsymbol{y} \mid \boldsymbol{x}; \theta) + \lambda_{avg} L_{avg} + \lambda_{var} L_{var} \tag{4.11}$$

where λ_{avg}, λ_{var} are hyperparameters that control both losses. Intuitively, while λ_{avg} controls

the overall speed, λ_{var} controls the divergence of the heads. Combining these two losses, we are able to dynamically control the range of attention heads so that we can control the latency and the reading buffer. For MMA-IL model, we only use L_{avg}; for MMA-H we only use L_{var}.

4.3 Experimental Setup

4.3.1 Datasets

Dataset	RNN	Transformer
IWSLT15 En-Vi	25.6 [2]	28.7
WMT15 De-En	28.4 (Arivazhagan et al., 2019)	32.3

Table 4.1: Offline model performance with unidirectional encoder and greedy decoding.

Dataset	Beam Search	Bidirectional Encoder	Unidirectional Encoder
WMT15 De-En	1	32.6	32.3
	4	33.0	33.0
IWSLT15 En-Vi	1	28.7	29.4
	10	28.8	29.5

Table 4.2: Effect of using a unidirectional encoder and greedy decoding to BLEU score. Greedy search and unidirectional encoder can potentially both affect the translation quality.

We evaluate our method on two standard machine translation datasets, IWSLT14 En-Vi and WMT15 De-En. For each dataset, we apply tokenization with the Moses (Koehn et al.,

[2]Luong et al. (2015) report a BLEU score of 23.0 but they didn't mention what type of BLEU score they used. This score is from our implementation on the data aquired from https://nlp.stanford.edu/projects/nmt/

2007) tokenizer and preserve casing.

IWSLT15 English-Vietnamese TED talks from IWSLT 2015 Evaluation Campaign (Cettolo et al., 2016). We follow the settings from Luong et al. (2015) and Raffel et al. (2017). We replace words with frequency less than 5 by $<unk>$. We use tst2012 as a validation set tst2013 as a test set.

WMT15 German-English We follow the setting from Arivazhagan et al. (2019). We apply byte pair encoding (BPE) (Sennrich et al., 2016) jointly on the source and target to construct a shared vocabulary with 32K symbols. We use newstest2013 as validation set and newstest2015 as test set.

4.3.2 Models

We evaluate MMA-H and MMA-IL models on both datasets. The MILK model we evaluate on IWSLT15 En-Vi is based on Luong et al. (2015) rather than RNMT+ (Chen et al., 2018). In general, our offline models use unidirectional encoders, i.e. the encoder self-attention can only attend to previous states, and greedy decoding. We report offline model performance in Table 4.1 and the effect of using unidirectional encoders and greedy decoding in Table 4.2. For MMA models, we replace the encoder-decoder layers with MMA and keep other hyperparameter settings the same as the offline model. Detailed

hyperparameter settings can be found in Appendix A.1. We use the Fairseq library (Ott et al., 2019) for our implementation.

4.4 Results

In this section, we present the main results of our model in terms of latency-quality tradeoffs, ablation studies and provide further analysis. In the first study, we analyze the effect of the variance loss on the attention span. Then, we study the effect of the number of decoder layers and decoder heads on quality and latency. We also provide a case study for the behavior of attention heads in an example. Finally, we study the relationship between the rank of an attention head and the layer it belongs to.

4.4.1 Latency-Quality Tradeoffs

We plot the quality-latency curves for MMA-H and MMA-IL in Figure 4.2. The BLEU and latency scores on the test sets are generated with different regularization weights. We select the checkpoint with best BLEU score on the validation set. We use differentiable average lagging (Arivazhagan et al., 2019) when setting the latency range. We find that for a given latency, our models obtain a better translation quality. While MMA-IL tends to have a decrease in quality as the latency decreases, MMA-H has a small gain in quality as latency

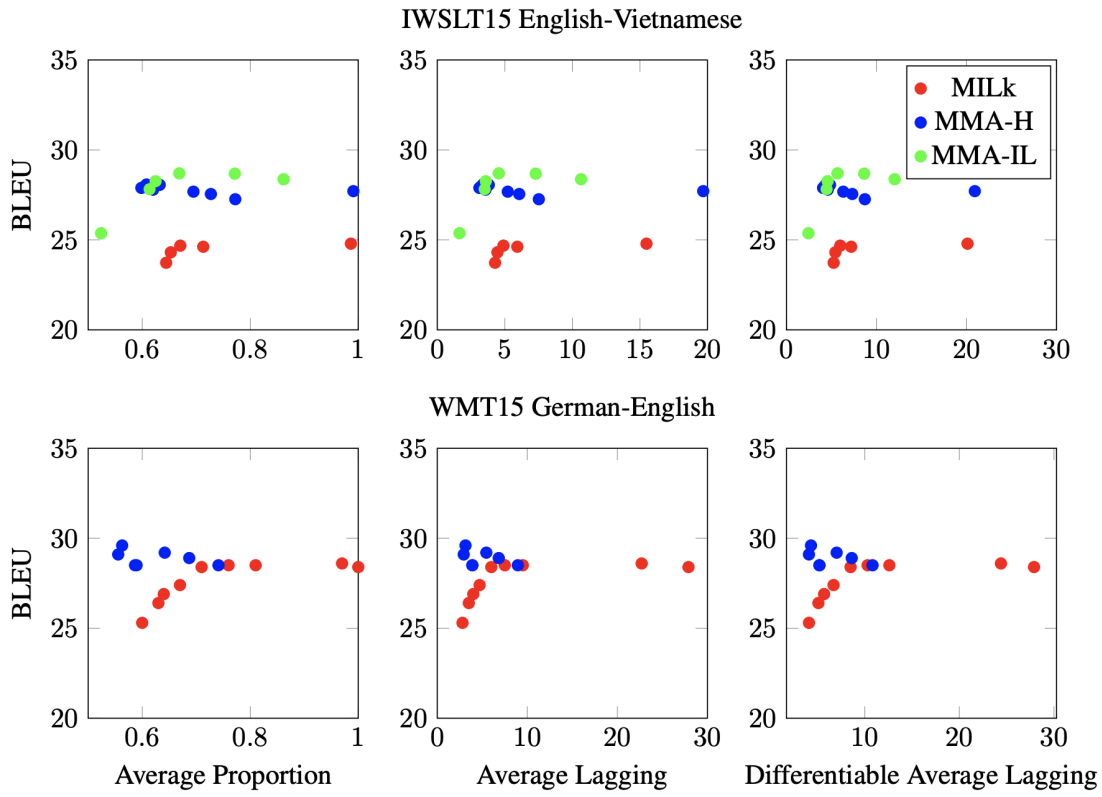

Figure 4.2: Latency-quality tradeoffs for MILk and MMA on IWSLT15 En-Vi and WMT15 De-En. Detailed results can be Appendix A.2

decreases: a larger latency does not necessarily mean an increase in source information available to the model. In fact, the large latency is from the outlier attention heads, which skip the entire source sentence and point to the end of the sentence. The outliers not only increase the latency but they also do not provide useful information. We introduce the attention variance loss to eliminate the outliers, as such a loss makes the attention heads focus on the current context for translating the new target token. It is interesting to observe

that MMA-H has a better latency-quality tradeoff than MILk[3] even though each head only attends to only one state. Although MMA-H is not yet able to handle an arbitrarily long input (without resorting to segmenting the input), since both encoder and decoder self-attention have an infinite lookback, that model represents a good step in that direction.

4.4.2 Attention Span

In Section 4.2.2, we introduced the attention variance loss to MMA-H in order to prevent outlier attention heads from increasing the latency or increasing the attention span. We have already evaluated the effectiveness of this method on latency in Section 4.4.1. We also want to measure the difference between the fastest and slowest heads at each decoding step. We define the average attention span in Equation 4.12:

$$\bar{S} = \frac{1}{|\boldsymbol{y}|} \left(\sum_{i}^{|\boldsymbol{y}|} \max_{l,h} t_i^{l,h} - \min_{l,h} t_i^{l,h} \right) \tag{4.12}$$

It estimates the reading buffer we need for streaming translation. We show the relation between the average attention span versus λ_{var} in Figure 4.3. As expected, the average attention span is reduced as we increase λ_{var}.

[3]The numbers of MILk on WMT15 De-En are from Arivazhagan et al. (2019)

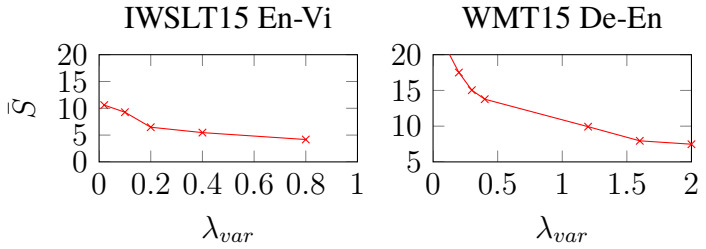

Figure 4.3: Effect of λ_{var} on the average attention span. The variance loss works as intended by reducing the span with higher weights.

4.4.3 Effect on Number of Layers and Number of Heads

One motivation to introduce MMA is to adapt the Transformer, which is the current state-of-the-art model for machine translation, to online decoding. Important features of the Transformer architecture include having a separate attention layer for each decoder layer block and multihead attention. In this section, we test the effect of these two components on the offline, MMA-H, and MMA-IL models from a quality and latency perspective. We report quality as measured by detokenized BLEU and latency as measured by DAL on the WMT13 validation set in Figure 4.4. We set $\lambda_{avg} = 0.2$ for MMA-IL and $\lambda_{var} = 0.2$ for MMA-H.

The offline model benefits from having more than one decoder layer. In the case of 1 decoder layer, increasing the number of attention heads is beneficial but in the case of 3 and 6 decoder layers, we do not see much benefit from using more than 2 heads. The best performance is obtained for 3 layers and 2 heads (6 effective heads). The MMA-IL model

behaves similarly to the offline model, and the best performance is observed with 6 layers and 4 heads (24 effective heads). For MMA-H, with 1 layer, performance improves with more heads. With 3 layers, the single-head setting is the most effective (3 effective heads). Finally, with 6 layers, the best performance is reached with 16 heads (96 effective heads).

The general trend we observe is that performance improves as we increase the number of effective heads, either from multiple layers or multihead attention, up to a certain point, then either plateaus or degrades. This motivates the introduction of the MMA model.

We also note that latency increases with the number of effective attention heads. This is due to having fixed loss weights: when more heads are involved, we should increase λ_{var} or λ_{avg} to better control latency.

4.4.4 Attention Behaviors

We characterize attention behaviors by providing a running example of MMA-H and MMA-IL, shown in Figure 4.5 and Figure 4.6. Each curve represents the path that an attention head goes through at inference time. For MMA-H, shown in Figure 4.5, we found that when the source and target tokens have the same order, the attention heads behave linearly and the distance between fastest head and slowest head is small. For example, this can be observed from partial sentence pair "I also didn't know that" and target tokens "Tôi cũng không biết rằng", which have the same order. However, when the source tokens

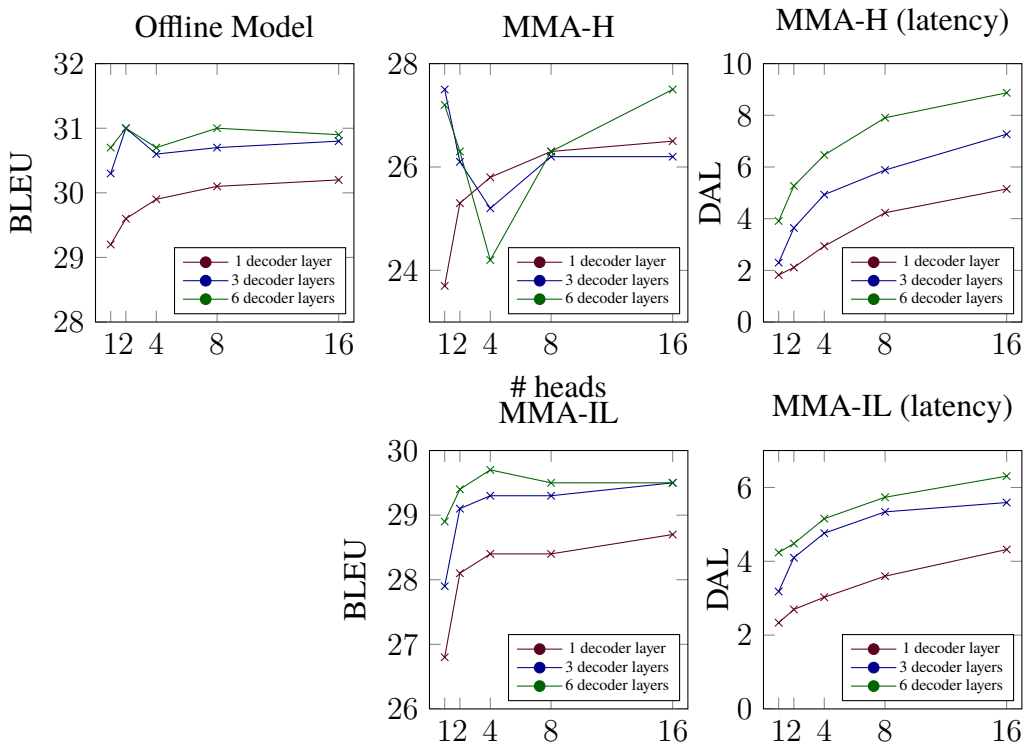

Figure 4.4: Effect of the number of decoder attention heads and the number of decoder attention layers on quality and latency, reported on the WMT13 validation set.

and target tokens have different orders, such as "the second step" and "bước (*step*) thứ hai (*second*)", the model will generate "bước (*step*)" first and some heads will stay in the past to retain the information for later reordered translation "thứ hai (*second*)". We can also see that the attention heads have a near-diagonal trajectory, which is appropriate for streaming inputs.

The behavior of the heads in MMA-IL models is shown in Figure 4.6. Notice that we remove the partial softmax alignment in this figure. We don't expect streaming capability

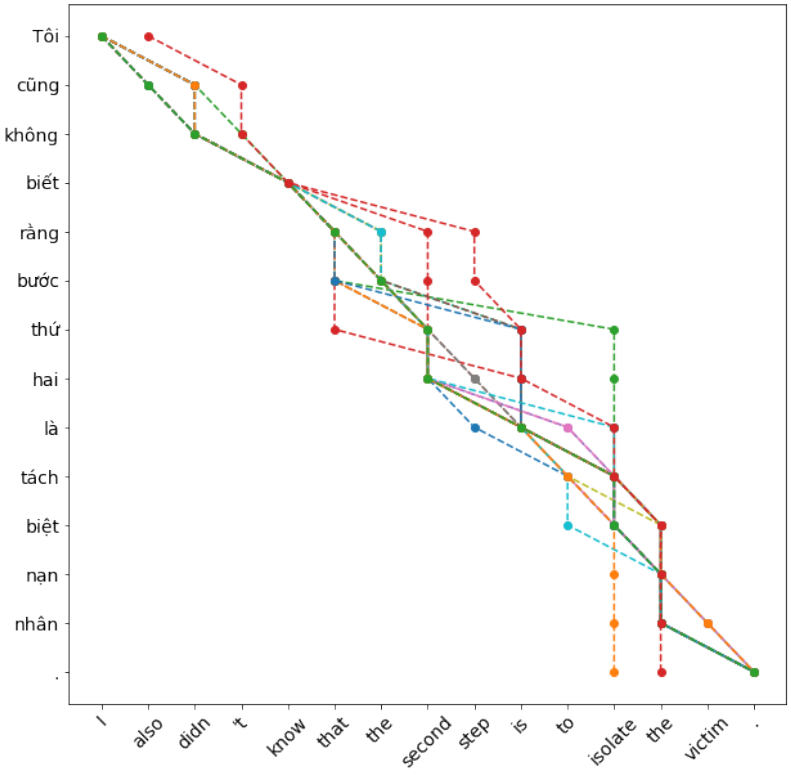

Figure 4.5: Attention heads movements of MMA-H, $L_{var} = 1.0$

for MMA-IL: some heads stop at early position of the source sentence to retain the history information. Moreover, because MMA-IL has more information when generating a new target token, it tends to produce translations with better quality. In this example, the MMA-IL model has a better translation on "isolate the victim" than MMA-H ("là cô lập nạn nhân" vs "là tách biệt nạn nhân")

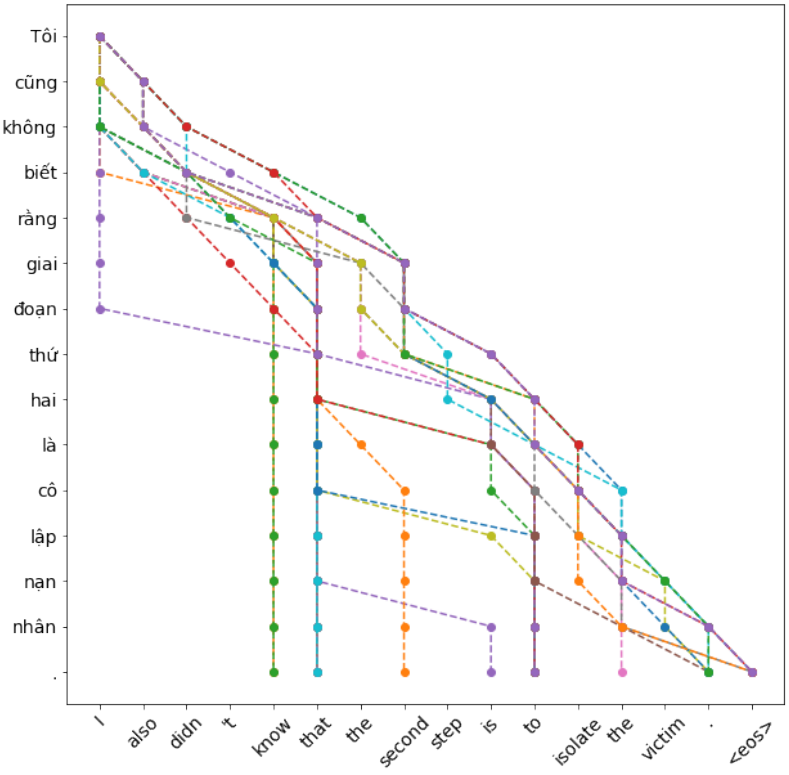

Figure 4.6: Attention heads movements of MMA-IL, $L_{avg} = 0.2$

4.4.5 Rank of the Heads

The rank of the attention is the number place during decoder. For instance, the fastest head has a rank of 1, while the slowest attention head has rank of $L \times H$ In Figure 4.7 and Figure 4.8, we calculate the average and standard deviation of rank of each head when generating every target token. For MMA-IL, we find that heads in lower layers tend to have higher rank and are thus slower. However, in MMA-H, the difference of the average rank are smaller. Furthermore, the standard deviation is very large which means that the order of

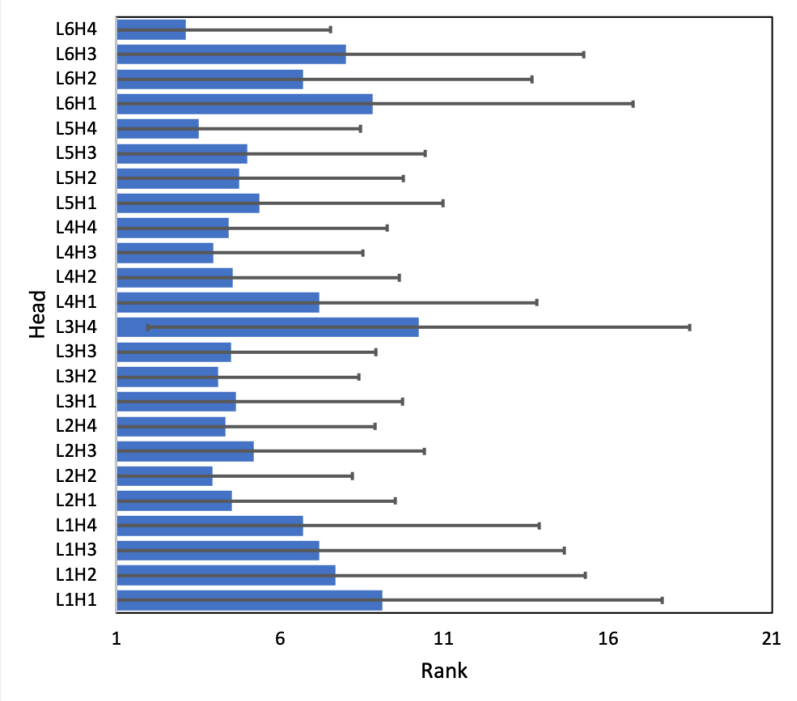

Figure 4.7: Attention heads movements of MMA-H, $L_{var} = 1.0$

the heads in MMA-H changes frequently over the inference process.

4.5 Conclusion

In this chapter, we propose two variants of the monotonic multihead attention model for simultaneous machine translation. By introducing two new targeted loss terms which allow us to control both latency and attention span, we are able to leverage the power of the Transformer architecture to achieve better quality-latency trade-offs than the previous state-of-the-art model. We also present detailed ablation studies demonstrating the efficacy

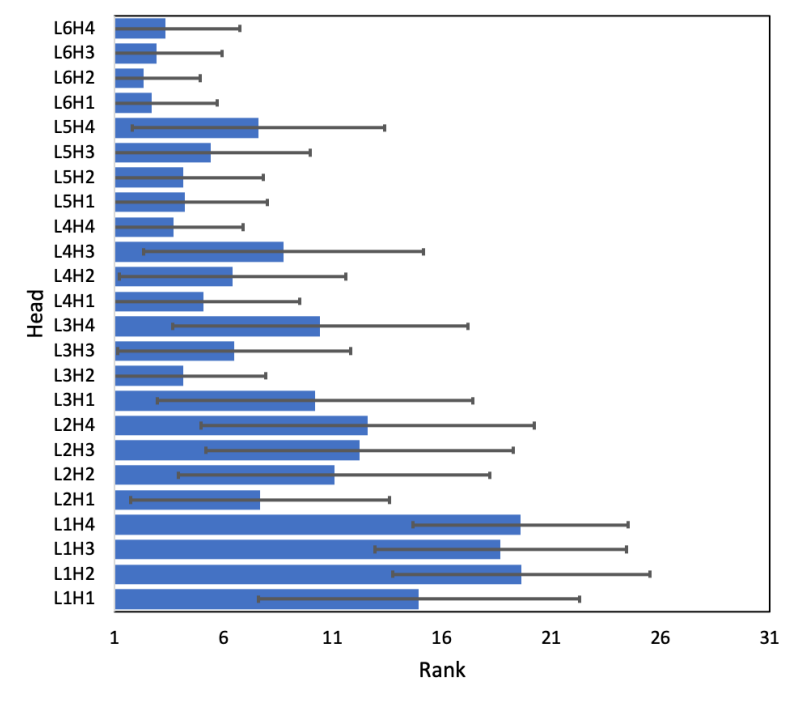

Figure 4.8: Attention heads movements of MMA-IL, $L_{avg} = 0.2$

Figure 4.9: The average rank of attention heads during inference on IWSLT15 En-Vi. Error bars indicate the standard deviation. L indicates the layer number and H indicates the head number.

and rationale of our approach. This work was publish in (Ma et al., 2019b) as a conference

paper in ICLR 2020

Chapter 5

End-to-End Simultaneous Speech-to-Text Translation

5.1 Introduction

The early research of simultaneous translation policies (Grissom II et al., 2014; Gu et al., 2017; Lawson et al., 2018; Zheng et al., 2019a; Zheng et al., 2019b; Arivazhagan et al., 2019; Ma et al., 2019b) focus on text input and output, given that most real-time systems back then is cascade based. Since great progress has recently been achieved on the end-to-end speech translation, there has been motivations to apply some policies to end-to-end models. An end-to-end simultaneous speech translation system can potentially, feature a

smaller model size, greater inference speed and fewer compounding errors compared to cascade systems. Some initial attempt of this kind (Ren et al., 2020) has demonstrated that end-to-end simultaneous speech-to-text systems can have lower latency than cascade systems.

In this chapter, we will introduce our first attempt on end-to-end speech-to-text simultaneous translation (SimulS2T). This work was published in Ma et al. (2020). We study how methods developed for simultaneous text translation (SimulT2T) can be adopted to end-to-end speech-to-text translation. We demonstrate that a direct application of SimulT2T methods to SimulS2T can be challenging:

1. The input of a speech translation system is much longer than text model.

2. The speech inputs are continuous speech features.

In order to address the issues, we introduce the concept of pre-decision module. Such module guides how to group encoder states into meaningful units prior to making a READ/WRITE decision. A detailed analysis of the latency-quality trade-offs when combining a fixed or flexible pre-decision module with a fixed or flexible policy is provided.

5.2 Methodology

5.2.1 Model Architecture

End-to-end S2T models directly map a source speech utterance into a sequence of target tokens. The encoder of a ST model takes a sequence of acoustic features $\mathbf{X} = \mathbf{S}_e$ as input and generates a sequence of hidden representations $\mathbf{H}^E = [h_1^E, ..., h_{|\mathbf{H}^E|}^E]$. The encoder usually includes a subsampling operation, such that $|\mathbf{H}^E| < |\mathbf{X}|$. We define the ratio of downsizing as

$$r_e = \frac{|\mathbf{H}^E|}{|\mathbf{X}|} \tag{5.1}$$

Next, the decoder takes $\mathbf{H}^E$ as input and generates decoder states $\mathbf{H}^D = [h_1^D, ..., h_{|\mathbf{H}_D|}^D]$, which are then converted to target tokens $\mathbf{Y} = [y_1, ..., y_{|\mathbf{Y}|}]$. In this chapter, We use the S-Transformer architecture proposed by Di Gangi et al. (2019b), which achieves competitive performance on the MuST-C dataset (Di Gangi et al., 2019a). In the encoder, two-dimensional attention is applied after the CNN layers and a distance penalty is introduced to bias the attention towards short-range dependencies. More specifically, the two-dimensional attention attends both time and feature dimensions.

We investigate two types of simultaneous translation mechanisms, flexible and fixed policy. In particular, we investigate monotonic multihead attention(MMA) (Ma et al., 2019b), which is an instance of flexible policy and the prefix-to-prefix model (Ma et al.,

2019a), an instance of fixed policy, designated by wait-k from now on, as introduced in Section 3.3.1

5.2.2 Pre-Decision Module

As introduced in Section 2.3, in SimulT2T, READ or WRITE decisions are made at the token (word or BPE) level. However, with speech input, it is unclear when to make such decisions. For instance, one could choose to read or write after each frame or after generating each encoder state. Under a common setting where $r_e = 4$, a frame typically only covers 10ms of the input while an encoder state generally covers 40ms of the input, and the average length of a word in our training dataset is 270ms. Thus, a policy like wait-k will not have enough information to write a token after reading a frame or generating an encoder state. Meanwhile, a flexible or model-based policy, such as MMA, should be able to handle granular input in theory. Our analysis will show, however, that while MMA is more robust to the granularity of the input, it also performs poorly when the input is too fine-grained.

In order to overcome these issues, we introduce the notion of a pre-decision module, which groups frames of encoder states, prior to making a simultaneous decision on READ or WRITE. A pre-decision module generates a series of trigger probabilities $p_{tr}(*)$ on each encoder states to indicate whether a simultaneous decision should be made. If $p_{tr} > 0.5$, the model triggers the simultaneous decision making, otherwise keeps reading new frames.

The full architecture of the model is shown in Figure 5.1. An overview of the SimulS2T

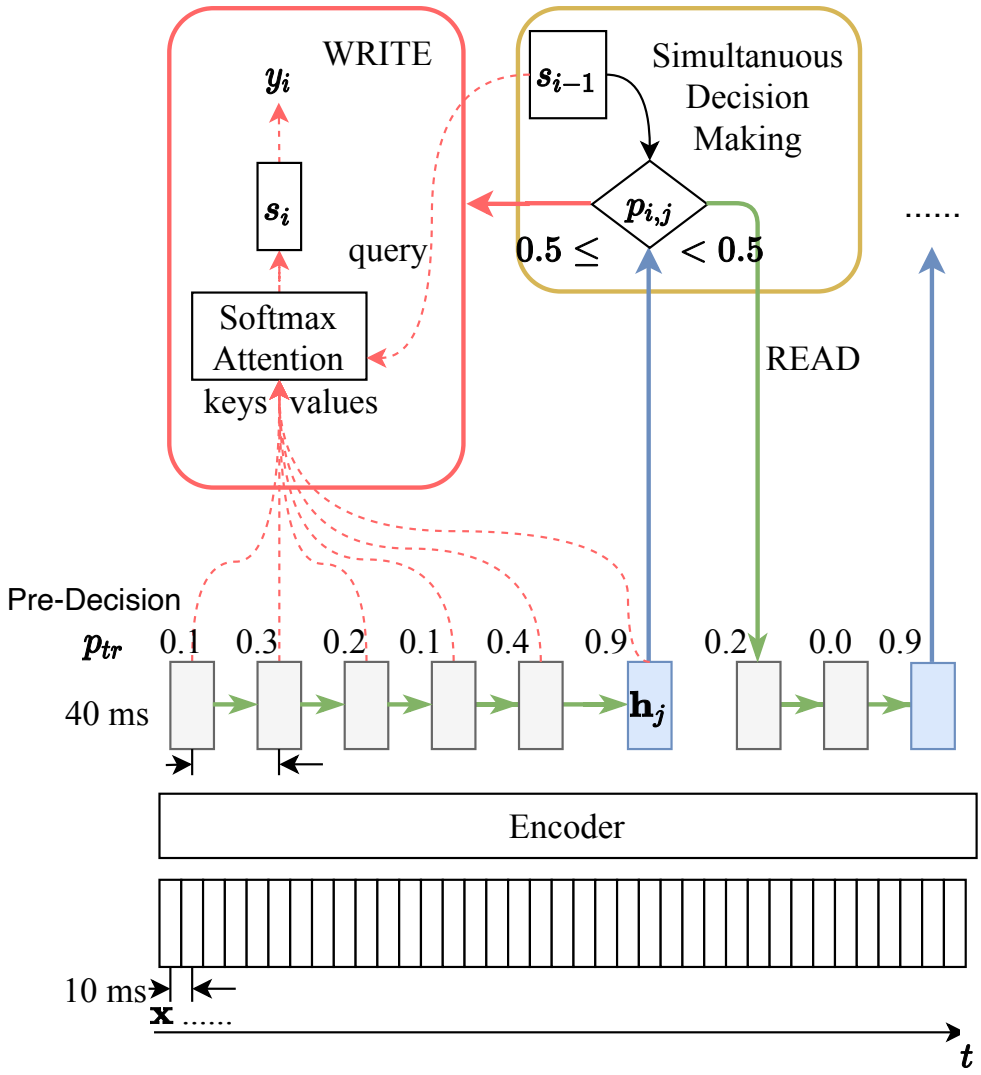

Figure 5.1: The full architecture of the proposed end-to-end simultaneous speech-to-text translation model with pre-decision module. The pre-decision module is an encoder-only module which group the encoder states for policy decision making. During the inference time, the pre-decision module first compute a trigger probabality p_{tr}. The simultaneous policy will only involve when $p_{tr} > 0.5$.

process with a pre-decision module is described in Algorithm 5.

Algorithm 5 Decoding process of end-to-end SimulST with a pre-decision module. The pre-decision module first checks whether to trigger the simultaneous policy decision making process.

Input: Input: $\mathbf{X}$
Input: Encoder subsampling ratio: r_e
Input: $i = 1, j = 1, t_0^{l,h} = 1, y_0 = \text{StartOfSequence}$.
while $y_{i-1} \neq \text{EOS}$ **do**
  $\boldsymbol{h}_j = \text{Encoder}(\mathbf{X}_{1:j \cdot r_e})$
  $p_{tr}(j) = \text{PreDecisionModule}(\boldsymbol{h}_j)$
  if $p_{tr}(j) > \gamma_{tr}$ **then**
    $p_{ij} = \text{MonotonicAttention}(\boldsymbol{s}_{i-1}, \boldsymbol{h}_j)$
    if $p_{ij} > 0.5$ **then**
      $y_i, \boldsymbol{s}_i = \text{Decoder}(\boldsymbol{S}_{1:i-1}, \mathbf{H}_{1:j})$
      $i = i + 1$
      Continue
  $j = j + 1$

We propose two types of pre-decision modules:

Fixed Pre-Decision A straightforward policy for a fixed pre-decision module is to trigger simultaneous decision making every fixed number of frames. Let Δt be the time corresponding to this fixed number of frames, with Δt a multiple of T_s, and $r_e = \text{int}(|\mathbf{X}|/|\mathbf{H}|)$

$$p_{tr}(j) = \begin{cases} 1 & \text{if mod}(j \cdot r_e \cdot T_s, \Delta t) = 0, \\ 0 & \text{Otherwise.} \end{cases} \tag{5.2}$$

Flexible Pre-Decision We use an oracle flexible pre-decision module that uses the source boundaries either at the word or phoneme level. Let $\boldsymbol{A}$ be the alignment between encoder

states and source labels (word or phoneme). $\boldsymbol{A}(h_i)$ represents the token that h_i aligns to.

$$p_{tr}(j) = \begin{cases} 0 & \text{if } \boldsymbol{A}(h_j) = \boldsymbol{A}(h_{j-1}) \\ 1 & \text{Otherwise.} \end{cases} \tag{5.3}$$

5.3 Experiments

We conduct experiments on the English-German portion of the MuST-C dataset (Di Gangi et al., 2019a), where source audio, source transcript and target translation are available. We train on 408 hours of speech and 234k sentences of text data. We use Kaldi (Povey et al., 2011) to extract 80 dimensional log-mel filter bank features, computed with a 25 ms window size and a 10 ms window shift. For text, we use SentencePiece (Kudo and Richardson, 2018) to generate a unigram vocabulary of size 10,000. We use Gentle[1] to generate the alignment between source text and speech as the label to generate the oracle flexible pre-decision module.

All speech translation models are first pre-trained on the ASR task where the target vocabulary is character-based, in order to initialize the encoder. We follow the same hyperparameter settings from (Di Gangi et al., 2019b). We follow the latency regularization method introduced by (Ma et al., 2019b; Arivazhagan et al., 2019). The objective function

[1]https://lowerquality.com/gentle/

to optimize is

$$L = -\log\left(P(\boldsymbol{Y}|\boldsymbol{X})\right) + \lambda \max\left(\mathcal{C}(\boldsymbol{D}), 0\right) \tag{5.4}$$

Where $\mathcal{C}$ is a latency metric (AL in this case) and $\mathbf{D}$ is described in Section 2.3. Only samples with AL > 0 are regularized to avoid overfitting. For the models with monotonic multihead attention, we first train a model without latency with $\lambda_{\text{latency}} = 0$. After the model converges, λ_{latency} is set to a desired value and we continue training the model until convergence.

5.4 Results

5.4.1 Latency-Quality Tradeoffs

We explore the latency-quality trade-offs of the 4 types of model from the combination of fixed or flexible pre-decision with fixed or flexible policies. The non computation-aware delays are used to calculate the latency metric in order to evaluate those trade-offs from a purely algorithmic perspective.

Fixed Pre-Decision + Fixed Policy We use the wait-k policy with k range from 1 to 10. The results are shown as Figure 5.2. As expected, both quality and latency increase with step size and lagging. In addition, the latency-quality trade-offs are highly dependent on the

step size of the pre-decision module. For example, with step size 120ms, the performance is very poor even with large k because of very limited information being read before writing a target token. Large step sizes improve the quality but introduce a lower bound on the latency. Note that step size 280ms, which yields the most effective latency-quality trade-off, also matches the average word length of 271ms. This motivates the study of a flexible pre-decision module based on word boundaries.

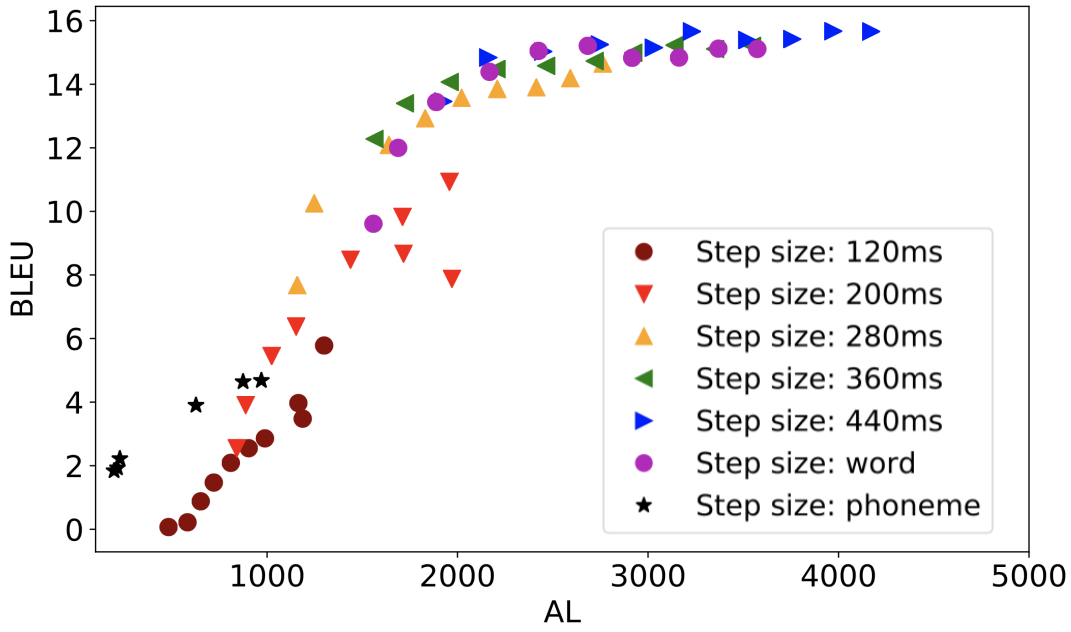

Figure 5.2: Latency-Quality trade-off curves for Wait-k. The unit of average legging (AL) is millisecond. Different colors indicate different step sizes.

Fixed Pre-Decision + Flexible Policy We conduct the experiments with MMA policy for this setting. For each curve, the latency loss weight λ in Equation 5.4 we use are

$[0.004, 0.01, 0.02, 0.04, 0.06, 0.08, 0.1]$. The results are shown in Figure 5.3 Similar to wait-k, MMA obtains very poor performance with a small step size of 120ms. For other step sizes, MMA obtains similar latency-quality trade-offs, demonstrating some form of robustness to the step size.

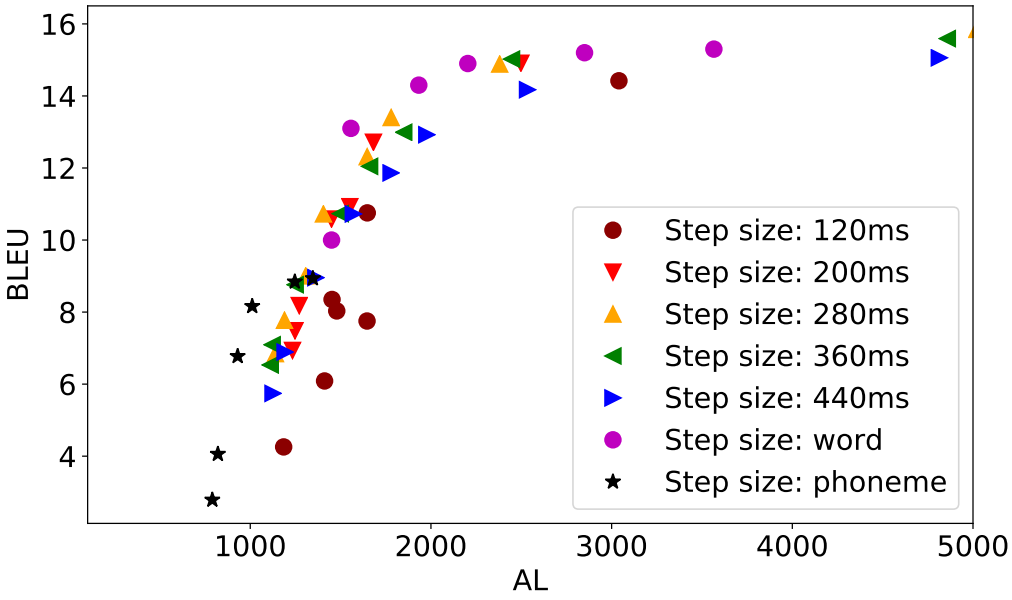

Figure 5.3: Latency-Quality trade-off curves for MMA. The unit of AL is millisecond.

Flexible Pre-Decision Curve ★ and ● in Figure 5.2 and Figure 5.3 show latency-quality trade-offs when the pre-decision module is determined by oracle word or phoneme boundaries. A SimulST model does not have the access to this information and the purpose of this experiment is to guide future design of a flexible pre-decision model. First, as previ-

ously observed, the granularity of the pre-decision greatly influences the latency-quality trade-offs. Models using phoneme boundaries obtain very poor translation quality because those boundaries are too granular, with an average phoneme duration of 77ms. In addition, comparing MMA and wait-k with phoneme boundaries, MMA is found to be more robust to the granularity of the pre-decision module.

Best Curves The best settings for each approach are compared in Figure 5.4. For fixed pre-decision, we choose the setting that has the best quality for each latency bucket of 500ms, while for the flexible pre-decision we use oracle word boundaries. For both wait-k and MMA, the flexible pre-decision module outperforms the fixed pre-decision module. This is expected since the flexible pre-decision module uses oracle information in the form of pre-computed word boundaries but provides a direction for future research. The best latency-quality trade-offs are obtained with MMA and flexible pre-decision from word boundaries.

5.4.2 Computation Aware Latency

We also consider the computation-aware latency described in Section 2.3, shown in Figure 5.5. The focus is on fixed pre-decision approaches in order to understand the relation between the granularity of the pre-decision and the computation time. Figure 5.5 shows that as the step size increases, the difference between the non-computation aware and the

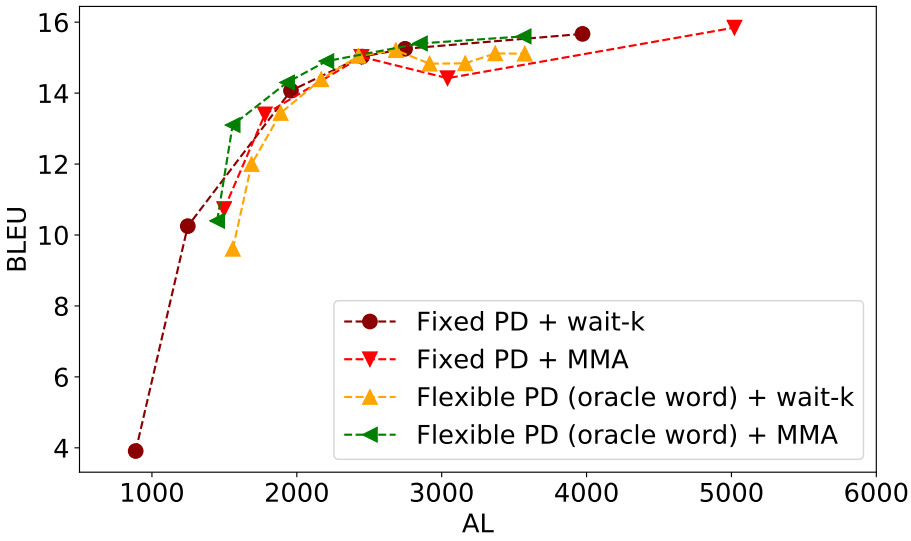

Figure 5.4: Comparison of best models in four settings

computation aware latency shrinks. This is because with larger step sizes, there is less overhead of recomputing the bidirectional encoder states [2]. We recommend future work on SimulS2T to make use of computation aware latency as it reflects a more realistic evaluation, especially in low-latency regimes, and is able to distinguish streaming capable systems.

5.5 Conclusion

This chapter introduced our first attempt on end-to-end simultaneous speech translation model. We investigated how to adapt SimulT2T methods to end-to-end SimulS2T by

[2]This is a common practice in SimulT2T where the input length is significantly shorter than in SimulS2T (Arivazhagan et al., 2019; Ma et al., 2019a)

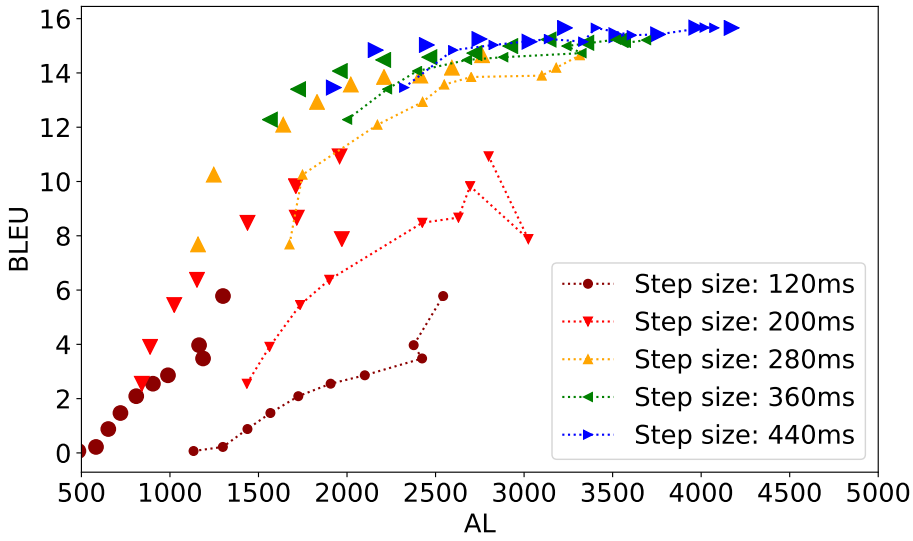

Figure 5.5: Computation-aware latency for fixed pre-decision + wait-k policy. Points on dotted lines are computation-aware, without lines are non-computation-aware

introducing the concept of pre-decision module. We also adapted original average lagging to be computation-aware. The effects of combining a fixed or flexible pre-decision module with a fixed or flexible policy were carefully analyzed. This work was publish in (Ma et al., 2020) as a conference paper in IJCNLP-AACL 2020

Chapter 6

Augmented Memory Transformer for Simultaneous Speech-to-Text Translation

6.1 Introduction

As introduced in previous chapters, while most previous work on simultaneous translation focus on text input (Ma et al., 2019a; Arivazhagan et al., 2019; Ma et al., 2019b) the end-to-end approach for simultaneous speech translation has also very recently attracted interest from the community (Ren et al., 2020; Ma et al., 2020) due to potentially lower

latency compared with cascade models. However, most studies tend to focus on an ideal setup, where the computation time to generate the translation is neglected. This assumption may be reasonable for text to text but not for speech to text translation since the latter has much longer input sequences. A simultaneous speech translation model may have the ability to generate translations with partial input but may not be useful for real-time applications because of slow computation in generating output tokens.

Meanwhile, despite the impressive progress on streaming ASR, most prior work is not directly applicable to translation. Encoder-only or transducer structures are widely implemented for ASR, since the output is assumed monotonically aligned to the input. In order to achieve an efficient streaming speech translation model, we combine streaming ASR and simultaneous translation techniques and introduce an end-to-end transformer-based speech translation model with an augmented memory transformer encoder (Wu et al., 2020).

In this chapter, we introduce a SimulS2T model based on the augmented memory transformer to address this issue. The augmented memory encoder has shown considerable improvements on latency with little sacrifice on quality with hybrid or transducer-based models on the ASR task. It incrementally encodes fixed-length sub-sentence level segments and stores the history information with a memory bank, which summarizes each segment. The self-attention is only performed on the current segment and memory banks. A decoder with simultaneous policies is then introduced on top of the encoder.

6.2 Methodology

The proposed streaming speech translation model, illustrated in Figure 6.1 consists of two components, an augmented memory encoder and a simultaneous decoder. The encoder incrementally and efficiently encodes streaming input, while the decoder starts translation with partial input, then interleaves reading new input and predicting a target token under the guidance of a simultaneous translation policy.

6.2.1 Augmented Memory Encoder

The self-attention module in the original transformer model (Vaswani et al., 2017) attends to the entire input sequence, which precludes streaming capability. We denote $\mathbf{H} = [\mathbf{h}_1, ...]$ as the input of a certain encoder layer, with $\mathbf{h}_t \in \mathbb{R}^D$. Notice that the length of $\mathbf{H}$ can be unbounded for a streaming model. In the encoder, each self-attention projects the input into query $\mathbf{Q}$, key $\mathbf{K}$ and value $\mathbf{V}$.

$$\mathbf{Q} = W_q\mathbf{H}, \mathbf{K} = W_k\mathbf{H}, \mathbf{V} = W_v\mathbf{H} \tag{6.1}$$

At each position j, a weight is calculated as follows

$$\alpha_{j,j'} = \frac{\exp(\beta \cdot \mathbf{Q}_j^T\mathbf{K}_{j'})}{\sum_k \exp(\beta \cdot \mathbf{Q}_j^T\mathbf{K}_k)} \tag{6.2}$$

Where $\beta = \frac{1}{\sqrt{D}}$ is a scaling factor. The self-attention at position j can then be calculated as

$$\mathbf{Z}_j = \sum_{j'} \alpha_{j,j'} \mathbf{V}_{j'} \tag{6.3}$$

The calculation of self-attention makes it inefficient for streaming applications. According to Equation 6.1, the self-attention needs full encoder states $\mathbf{H}$ for calculation. To address this issue, Wu et al. (2020) proposes an augmented memory transformer encoder to address this issue. Instead of attending to entire input sequence $\mathbf{X}$, the self-attention is applied to a sequence of sub-utterance level segments $\mathbf{S} = [\mathbf{s}_1, ...]$. A segment $\mathbf{s}_n$, which contains a span of input features, consists of three parts: left context $\mathbf{l}_n$ of size L, main context $\mathbf{c}_n$ of size C and right context $\mathbf{r}_n$ of size R. Each segment overlaps with adjacent segments — the overlap between current and previous segment is $\mathbf{l}_n$, and between current and the next segment is $\mathbf{r}_n$. Self-attention is computed at the segment level, which reduces the amount of computation. The new query, key and value for each segment are

$$\mathbf{q}_n = \mathbf{W}_\mathbf{q}(\mathbf{l}_n, \mathbf{c}_n, \mathbf{r}_n, \sigma_n) \tag{6.4}$$

$$\mathbf{k}_n = \mathbf{W}_\mathbf{k}(\mathbf{M}_{n-N:n-1}, \mathbf{l}_n, \mathbf{c}_n, \mathbf{r}_n) \tag{6.5}$$

$$\mathbf{v}_n = \mathbf{W}_\mathbf{v}(\mathbf{M}_{n-N:n-1}, \mathbf{l}_n, \mathbf{c}_n, \mathbf{r}_n) \tag{6.6}$$

Where $\sigma_n = \sum_{\mathbf{x}_k \in \mathbf{s}_n} \mathbf{x}_k$ is a summarization of the segment $\mathbf{s}_n$, and $\mathbf{M}_{n-N:n} = [\mathbf{m}_{n-N}, ..., \mathbf{m}_{n-1}]$ are the memory banks. Denote q_{-1} is a projection of σ_n and $\alpha_{-1,j'}$ is the attention weight when the query is q_{-1}. Each memory bank is calculated as follows:

$$\mathbf{m}_n = \sum_{j'} \alpha_{-1,j'} (\mathbf{v}_n)_{j'} \tag{6.7}$$

which is introduced to represent history information. A hyperparameter N controls how many memory banks are retained. Self-attention is then calculated as follows:

$$\alpha_{j,j'} = \frac{\exp((\mathbf{q}_n^T)_j (\mathbf{k}_n)_{j'})}{\sum_k \exp(\beta \cdot (\mathbf{q}_n^T)_j (\mathbf{k}_n)_k)} \tag{6.8}$$

$$(\mathbf{z}_n)_j = \sum_{j'} \alpha_{j,j'} (\beta \cdot \mathbf{v}_n)_{j'} \tag{6.9}$$

Where $N + L < j \leq N + L + C$. Then only the central encoder states are kept and a the concatenation of the segment states $\mathbf{Z} = [\mathbf{z}_1, ...]$ is passed to decoder. Because of the left and right contexts, an arbitrary encoder can run on the segments without boundary mismatch. In this paper, we adapt the encoder of convtransformer (Inaguma et al., 2020). Before the self-attention layers, the encoder consists of two convolutional layers, with stride of 2 on temporal dimension for each layer.

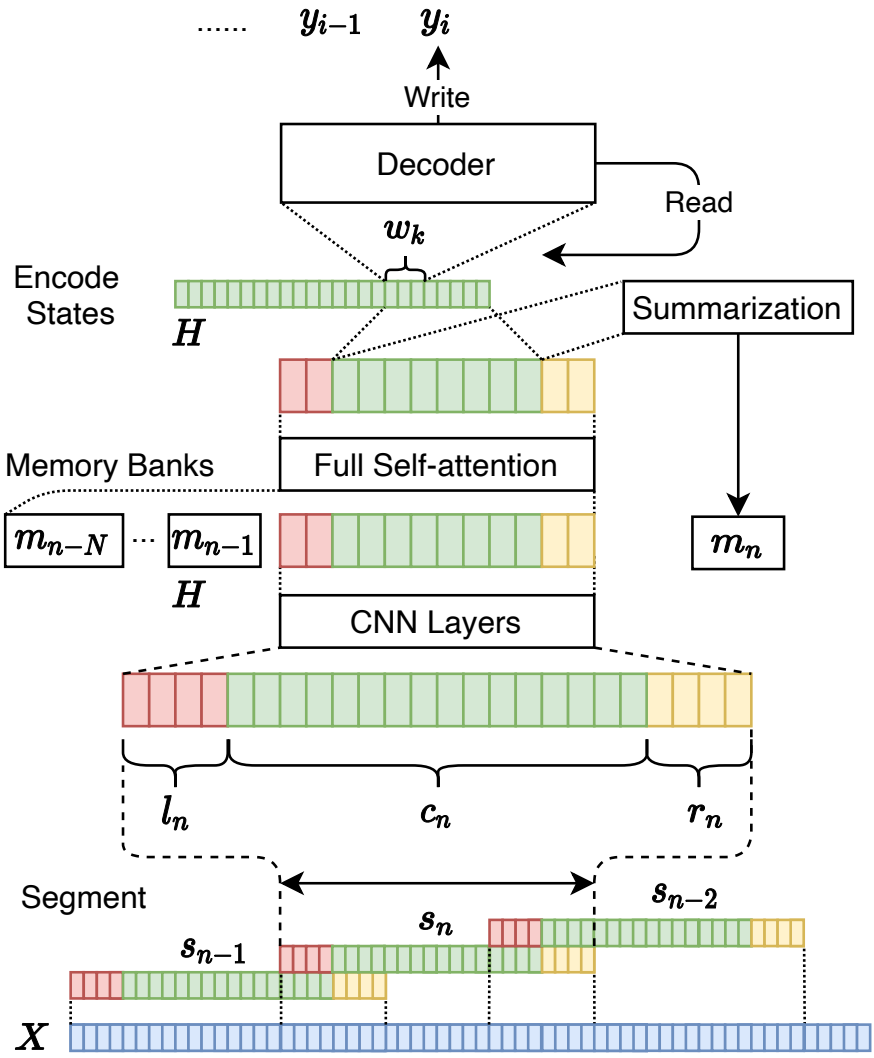

Figure 6.1: Architecture of streaming transformer model with an augmented memory encoder. The encoder only perform fully self-attention on segments of speech to improve efficiency, while the decode operate simultaneous policies.

6.2.2 Simultaneous Decoder

A simultaneous decoder starts translation with partial input based on a policy. Simultaneous policies decide whether the model should read new inputs or generate a new prediction at a given time. However, different from text translation, our preliminary experiments show that for simultaneous speech translation, encoder states are too fine-grained for policy learning. Thus, we adopt the idea of pre-decision introduced in Section 5.2.2, for better efficiency by making simultaneous read and write decision on chunks of encoder states. Here, we use the simple fixed pre-decision strategy where the decision is made every fixed number of encoder states. We denote the sequence of chunks as $\mathbf{W} = [\mathbf{w}_1, ...]$ and the start and end encoder state index of $\mathbf{w}_k$ is $W_s(k), W_e(k)$. We denote the prediction of model as $\mathbf{Y} = [y_1, ...]$. The general decoding algorithm of a simultaneous policy $\mathcal{P}$ with augmented memory transformer is described in Algorithm 6. In theory, Algorithm 6 supports arbitrary simultaneous translation policies. For simplicity, wait-k (Ma et al., 2019a) is used. It waits for k source tokens and then operating then reading and writing alternatively. Notice that our method is compatible with an arbitrary simultaneous translation policy.

Note that the decoder self-attention still has access to all previous decoder hidden states up to previous end-of-sentence token; in order to preserve streaming capability for the decoder, decoder states are reset every time an end-of-sentence token is predicted. Augmented memory is not introduced in the decoder because the target sequence is dramatically smaller

Algorithm 6 Chunk-based simultaneous policy with an augmented memory encoder

Input: Chunk-based simultaneous policy $\mathcal{P}$
Input: Streaming input $\mathbf{X}$. Memory banks $\mathbf{M}$. Prediction $\mathbf{Y}$
Input: Maximum memory size N. Decision chunk size W
Input: Central context size C. Encoder pooling ratio R
Input: $i = 1, n = 1, k = 1$.
Input: $W_e(1) = 1, y_0 = \text{BOS}$

while $y_{i-1} \neq \text{EndOfTranslation}$ **do**
  if $W_e(k) + W > n \cdot C \cdot R$ **then**
    $\mathbf{z}_n, \mathbf{m}_n = \text{Encoder}(\mathbf{s}_n, \mathbf{M}_{n-N:n-1})$
    $\mathbf{Z} = [\mathbf{Z}, \mathbf{z_n}], \mathbf{M} = [\mathbf{M}, \mathbf{m_n}]$
    $n = n + 1$ # Read a new segment of input features
  $\mathbf{w}_k = \text{Summarize}(\mathbf{Z}_{W_s(k):W_e(k)})$
  $p_{ik} = \mathcal{P}([\mathbf{Y}_{1:i-1}], \mathbf{w}_k)$
  if $p_{ik} > 0.5$ **then**
    $y_i = \text{Decoder}([\mathbf{Y}_{1:i-1}], \mathbf{Z})$
    $i = i + 1$ # Predict a target token
  else
    $W_s(k+1) = W_e(k) + 1$
    $k = k + 1$ # Move to the next chunk of encoder states

than the source speech sequence. The decoder can still operate in negligible time, even if the input becomes longer.

6.3 Experiments

Experiments were conducted on the English-German MuST-C dataset (Di Gangi et al., 2019a). The training data consists of 408 hours of speech and 234k sentences of text. We use Kaldi (Povey et al., 2011) to extract 80 dimensional log-mel filter bank features. The features are computed with a $25ms$ window size and a $10ms$ window shift and normalized

with global cepstral mean and variance. Text is tokenized with a SentencePiece (Kudo and Richardson, 2018) 10k unigram vocabulary. Translation quality is evaluated with case-sensitive detokenized BLEU with SacreBLEU[1]. Latency is evaluated by Average Lagging (Ma et al., 2019a; Ma et al., 2020), with the SimulEval (Ma et al., 2020b) toolkit.

The speech translation model is based on the convtransformer architecture (Inaguma et al., 2020; Di Gangi et al., 2019a). It first contains two convolutional layers with subsampling ratio of 4. Both encoder and decoder have a hidden size of 256 and 4 attention heads. There are 12 encoder layers and 6 decoder layers. The model is trained with label smoothed (0.1) cross entropy. We use the Adam optimizer (Kingma and Ba, 2015), with a learning rate of 0.0001 and an inverse square root schedule.

We use a simplified version of Ren et al. (2020) as our baseline model. While in Ren et al. (2020), the simultaneous policy on word boundaries generated by a separate model, we simply utilize a fixed-decision module introduced in Section 5.2.2. Our choice is motivated by the fact that in Chapter 5, a fixed chunk size gave similar quality-latency trade-offs as word boundaries. A unidirectional mask is introduced to prevent the encoder from looking into future information.

All transformer-based speech translation models are first pre-trained on the ASR task, in order to initialize the encoder. Each experiment is run on 8 Tesla V100 GPUs with 32

[1] `https://github.com/mjpost/sacrebleu`

GB memory. The code is developed based on Fairseq[2].

6.4 Results

We first analyze the effect of the segment and context sizes, and use the resulting optimal settings for further analysis on the maximum number of memory banks and for comparison with the baseline.

6.4.1 Effect of Segment and Context Size

We analyze the effect of different segment, left and right context sizes. For all experiments, we use the wait-k ($k = 1, 3, 5, 7$) policy on a chunk of 8 encoder states (Ma et al., 2020). The latency-quality trade-offs with different sizes are shown in Figure 6.2. We first observe that, increasing the left and right context size will improve the quality with very small increase on latency, for instance, from curve "S64 L16 R16" to "S64 L32 R32". This indicates that context of both sides can alleviate the boundary effect. We also notice that when we reduce the segment size from 64 to 32, the BLEU score decreases dramatically. Similar observations are made in (Wu et al., 2020) but the ASR models are more robust to decreasing the segment and context sizes. We hypothesize that reordering in translation

[2] https://github.com/pytorch/fairseq

makes the model more sensitive to these sizes.

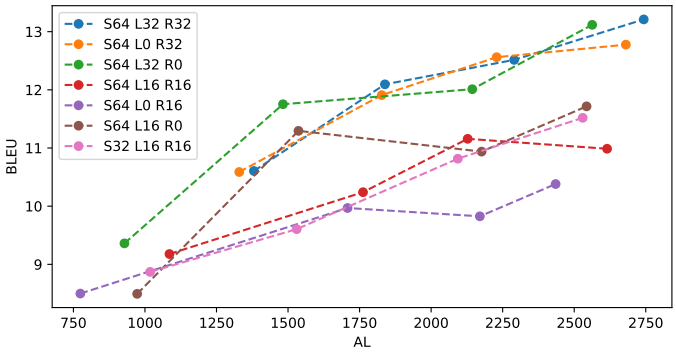

Figure 6.2: Effect of segment, left and right context size. Each curve represents wait-k, $k = 1, 3, 5, 7$ policies. The size is measured on a frame of 10 ms. "S$\{x\}$ L$\{y\}$ R$\{z\}$" means a encoder with segment size x, left size y and right size x.

6.4.2 Number of Memory Banks

In streaming translation, the input is theoretically unbounded. In order to prevent memory explosion, we explore the effect of reducing the number of the memory banks. Figure 6.3 shows the effect of different numbers of memory banks. We can see that the model is very robust to the size of the memory banks. Similar to Wu et al. (2020), when the maximum number of memory banks is large, for instance, larger than 3, there is little or no performance drop. However, we still observe a drop in performance with a maximum number of one memory bank. Finally, we found that training with different maximum numbers of memory banks was necessary as limiting the number of memory banks only at

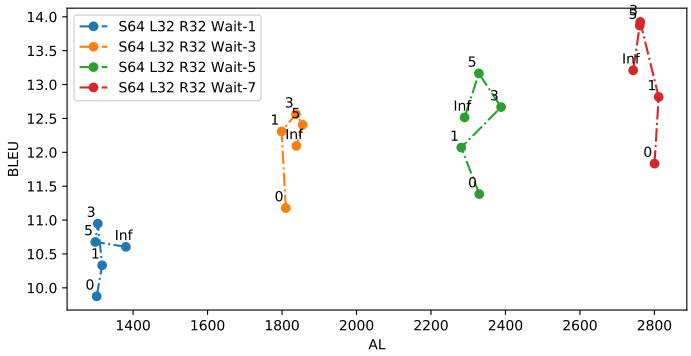

Figure 6.3: Effect of the maximum number of memory banks. Each curve represents one policy. The number on top of the nodes indicates the number of memory banks being kept.

inference time degraded performance.

6.4.3 Comparison with Baseline

In Figure 6.4, we compared our model with the baseline model described in Section 6.3. The proposed model achieves better quality with an increase in computation aware and non computation aware latency. The baseline achieves competitive latency because it only updates encoder states every 8 steps. However, there may be instances where recomputing encoder states every step may be needed, for example in the case of a flexible pre-decision module or when a the model includes a boundary detector (Ren et al., 2020). In Figure 6.4, the computation aware AL for the baseline increases substantially with a chunk of size 1.

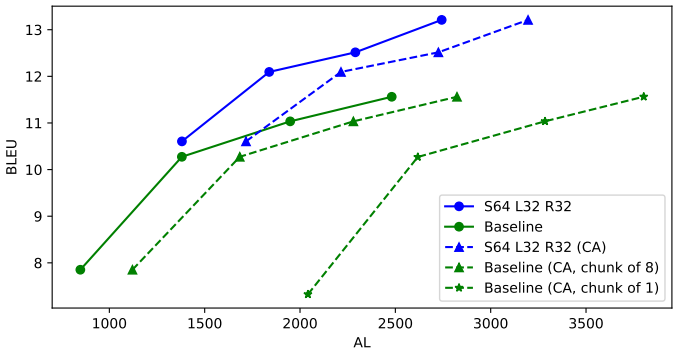

Figure 6.4: Comparison with baseline model. CA indicates measured by computation aware latency. Chunk of x means that the encoder states are updated every x steps.

6.5 Conclusion

In this chapter, we tackle the real-life application of streaming simultaneous speech translation. We propose a transformer-based model, equipped with an augmented memory, in order to handle long or streaming input. We study the effect of segment and context sizes, and the maximum number of memory banks. We show that our model has better quality with an acceptable latency increase compared with a transformer with unidirectional mask baseline and presents better quality-latency trade-offs than that baseline where encoder states are recomputed at every step. This work was publish as Ma et al. (2021) in ICASSP 2021.

Chapter 7

Direct Simultaneous Speech-to-Speech Translation

7.1 Introduction

While a fair amount of research has been conducted on simultaneous translation with text target, only a few works have explored simultaneous speech-to-speech translation (SimulS2S) (Zheng et al., 2020; Sudoh et al., 2020) which mostly adopt the cascaded approach. However, as discussed in Section 3.2, cascaded systems have several disadvantages. First, the pipeline of multiple submodules introduces extra latency. Second, the errors can be propagated and accumulated through the pipeline.

Recent efforts on direct S2ST provide a new possibility for SimulS2S, where the intermediate text representations are no longer needed. Among these models, Lee et al. (2022) proposed a direct S2ST model, where a sequence of discrete units instead of spectrograms are directly generated from the translation model. The discrete units are then passed to a vocoder for target speech synthesis. The labels of units are self-supervised representations learned from HuBERT (Hsu et al., 2021). A seperate vocoder can be trained separately and run on-the-fly given the discrete units. Such approach shows potential computationally advantage compared with spectrogram based model such as Translatotron (Jia et al., 2019b) and Translatotron 2 (Jia et al., 2021)

Meanwhile, we find it can be challenging to adapt simultaneous policies to SimulS2S model. Several prior works on monotonic attention based policies (Raffel et al., 2017; Arivazhagan et al., 2019; Ma et al., 2019b) use a closed form estimation on simultaneous alignment during the training time. We find that such estimation can be biased given long sequences, which usually happens in speech applications.

In this chapter, we introduce one of the first attempts for direct simultaneous speech-to-speech translation SimulS2S model. The model is based on recent progress on speech-to-units (S2U) translation (Lee et al., 2022), along with a novel simultaneous policy — variational monotonic multihead attention (V-MMA). Our approach is featured with two characteristics. First, the model is simultaneous, and able to generate the target speech

based on the partial source speech before obtaining the complete input; second, the model is independent from intermediate text outputs when generating target speech. We carry out experiments on the Fisher Spanish-English (Post et al., 2013) and MuST-C English-Spanish datasets (Di Gangi et al., 2019a) and provide a comprehensive empirical comparison between direct and cascaded models in SimulS2S.

7.2 Methodology

7.2.1 Variational Monotonic Multihead Attention

As introduced in Section 3.3.1, the alignment between source and target sequences is estimated with Equation 3.4. However, in Chapter 5, we found that such approach degrades the model performance on speech-to-text translation. We also observed this deficient policy learnt in the preliminary experiments on speech-to-speech task. We found that as the length of speech sequence increases, the estimation of alignment $\hat{\alpha}_{i,j}$ tends to be divergent, which is due to the recurrent calculation of $\hat{\alpha}_{i,j}$ in Equation 3.4.

To address this issue, we propose the variational monotonic multihead attention (V-MMA), which models the alignment with a latent variable α instead of recurrent estimation in sequence-to-sequence modeling. Suppose that the source sequence is $\mathbf{X}$ and the target sequence is $\mathbf{Y}$. Traditional seq2seq model is trained to maximize the log probability

$\log p(Y|X)$. Instead, we maximize its evidence lower bound (ELBO) derived with the latent variable α as below.

$$\begin{aligned}\mathcal{L}(\omega, \phi, \theta) =& \mathbb{E}_{\alpha \sim q_\phi} \left[\log p_\theta(\mathbf{Y}|\mathbf{X}, \alpha)\right] \\ & - \mathrm{KL}\left[q_\phi(\alpha|\mathbf{Y}, \mathbf{X})||p_\omega(\alpha|\mathbf{X})\right],\end{aligned} \tag{7.1}$$

where $p_\omega(\alpha|\mathbf{X})$ is the prior of the monotonic alignment, $q_\phi(\alpha|\mathbf{Y}, \mathbf{X})$ is the posterior of the monotonic alignment and $p_\theta(\mathbf{Y}|\mathbf{X}, \alpha)$ is the prediction of target sequence given the input and alignment. The latent variable α, in the form of a matrix, is a monotonic alignment between the target and source sequences of size $|\mathbf{X}|$ by $|\mathbf{Y}|$, where α_{ij} indicates the y_i is generated from x_1 to x_j.

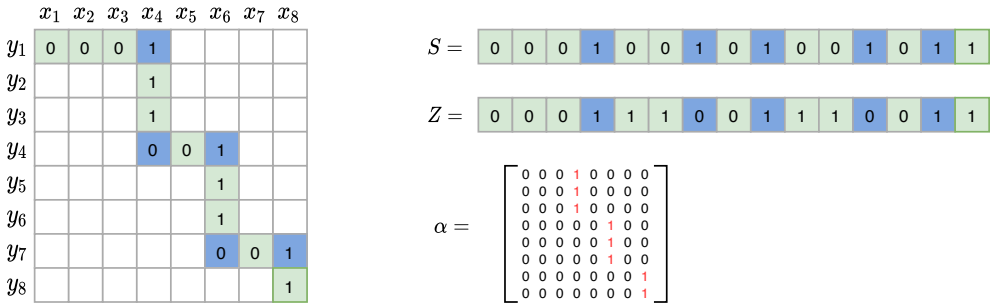

Figure 7.1: Sampling process of variantional monotonic multihead attention. The green path contains the all the actions, while the blue blocks indicate the switch point. We first sample S, which is the linearization of switching actions. Then we found the Z, which is the linearization of the simultaneous actions. Finally from Z we can have α, which is the alignment used for training.

However, sampling α is non-trivial since α is a 2-D matrix with monotonicity. Therefore,

we propose an approach for efficient sampling, as shown in Figure 7.1. We can first find the one-to-one mapping between α and a binary sequence Z of length $|X| + |Y| + 1$, where each $z \in Z$ indicates write action if $z = 1$, otherwise read action. We illustrate an example of matrix α and its corresponding sequence Z in Figure 7.1. Given a sequence Z, we can map it to a unique path in a fixed length 2-D grid, which is then used as alignment matrix α. Because the total number of write actions is the number of target words, we have $\sum_{z \in Z} z = |Y|$. Sampling 1-D sequence of Z is easier than 2-D matrix of α. Furthermore, we want the learned policy to have lower frequency of change for the purpose of smooth speech segment synthesis. Therefore, we sample a sequence of the change of action S for a better control during the training time, while for $1 < k < |X| + |Y| + 1$

$$s_k = \begin{cases} 1 & z_{k-1} \neq z_k \\ 0 & \text{otherwise} \end{cases} \tag{7.2}$$

During sampling, each s_k is sampled from a Bernoulli distribution parameterized by p_k^s,

$$p_k^s = (1 - e^{\lambda(k-k')^2}) f(X_{:i}, Y_{:j}), \tag{7.3}$$

where λ is a hyperparameter controlling the frequency of change, i is the current target size, j is the current source size, and $k' = \text{argmax}_{k'<k} S_{k'} = 1$ is the latest index of action

change. $f(X_{:i}, Y_{:j})$ is the context score, which is used as $p_{i,j}$ in Equation 3.4. Finally, we sample of α from Z, which is derived from S given the one-to-one mapping between S and Z.

Additionally, the calculation of the second term in Equation 7.1 can be written as

$$\begin{aligned}
&\mathrm{KL}\left[q_\phi(\alpha|\mathbf{Y},\mathbf{X})||p_\omega(\alpha|\mathbf{X})\right] \\
&= \log \frac{q_\phi(\alpha|Y,X)}{p_\omega(\alpha|X)} \\
&= \log \frac{\prod_{i=1}^{N} \phi_{i,z_i} \prod_{i=0}^{N} \prod_{j=a_i}^{z_{i+1}-1}(1-\phi_{i,j})}{\prod_{i=1}^{N} \omega_{i,z_i} \prod_{i=0}^{N} \prod_{j=z_i}^{z_{i+1}-1}(1-\omega_{i,j})} \\
&= \sum_{i=1}^{N} \log \frac{\phi_{i,z_i}}{\omega_{i,z_i}} + \sum_{i=1}^{N} \sum_{j=z_i}^{z_{i+1}-1} \log \frac{1-\phi_{i,j}}{1-\omega_{i,j}},
\end{aligned} \tag{7.4}$$

where the ϕ is the posterior probability matrix and ω is the prior probability matrix. We fixed diagonal as the prior.

7.2.2 Simultaneous Speech-to-Units Model

Figure 7.2 illustrate the architecture of the direct Simul-S2ST model with discrete units. The encoder reads the speech features with downsampling, and generates a sequence of hidden representations. Then, the simultaneous policy replaces soft-attention in the offline model to connect encoder and decoder. Because of the granularity of the encoder states, we follow the same setup as Ma et al. (2020) to use a fixed pre-decision module before

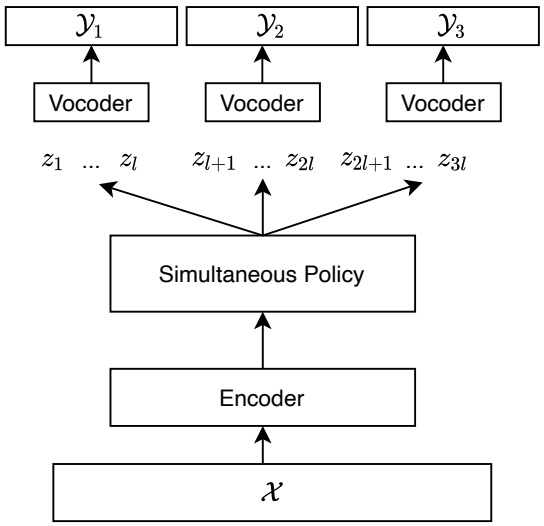

Figure 7.2: Architecture of direct Simul-S2ST model with discrete units.

applying the simultaneous policy. The policy contains two actions: READ and WRITE. The READ action indicates that the model takes another chunk of speech segment to update the encoder states, while the WRITE action predicts the discrete units.

A vocoder will be applied to the discrete units to synthesize the final speech output. Denote the emission rate l, the vocoder will be called every time l units are predicted. Note that the vocoder is not used during training time. Because the vocoder is trained on short speech segment in a non-autoregressive manner, as shown in the results, a small l can achieve good latency without a huge sacrifice on quality.

7.3 Experiments

Two datasets–Fisher and MuST-C–commonly used for speech translation tasks are used in this work.

- Fisher Spanish-English dataset (Post et al., 2013). The dataset consists of $139k$ sentences from telephone conversations in Spanish, the corresponding Spanish text transcriptions and their English text translation. As (Lee et al., 2022), a high-quality in-house TTS is used to prepare English speech with a single female voice.

- MuST-C dataset (Di Gangi et al., 2019a). It is a multilingual speech translation corpus collected from TED Talks. We use English-Spanish data, where we synthesize Spanish speech from Spanish texts provided by MuST-C with the help of an in-house TTS model.

Various models are included as baselines for a comprehensive empirical comparison.

- Offline cascaded S2S translation. The offline cascaded system consists of two components: S2T and TTS. The S2T model is the s2t_transformer_s architecture provided by FAIRSEQ S2T for speech-to-text translation (Wang et al., 2020a). The target vocabulary in S2T model consists of 47 English characters on Fisher data, and 8000 Spanish unigrams on MuST-C data. As for the TTS, we use a Transformer model with 6 layers, 4 attention heads and dimensions of 512 and 2048, and a HiFi-GAN

based vocoder generating speech from the predicted mel-spectrograms (Kong et al., 2020).

- Simultaneous cascaded S2S model consisting of S2T and TTS modules. Its S2T component has the same architecture as the offline S2T Transformer. As for the simultaneous strategies in S2T part, we use two methods: (1) Wait-k strategy, (2) MMA strategy. As for the incremental TTS module in the cascaded model, we adapt the non-autoregressive FastSpeech 2 model (Ren et al., 2022). It incrementally generates utterances word-by-word, utilizing the duration predictor for segmenting output word boundaries (Stephenson et al., 2021). It further uses a lookahead of 1 word, which is equivalent to a wait-k strategy at test-time (Ma et al., 2019a; Ma et al., 2020a), where k is two words[1]. To improve performance on partial inputs, we apply the prefix augmentation procedure described in Liu et al. (2022).

- Offline direct S2S translation (Lee et al., 2022). It is a state-of-the-art direct translation model without reliance on intermediate text outputs. On Fisher dataset, the direct model uses 12 encoder layers, 6 decoder layers, 4 attention heads, an embedding dimension of 256 and a feedforward dimension of 2048. As for MuST-C data, we use a larger model with attention heads increased to 8 and the embedding dimension increased to 512.

[1]Wait-$(k+1)$ is equivalent to lookahead-k following the definition in (Ma et al., 2020a)

- Direct S2S translation with wait-k policy (Ma et al., 2019a). We simply add the policy to attention mechanism in offline direct S2S translation model. We also try to directly add monotonic multihead attention (MMA) module (Ma et al., 2019b), but the model failed to converge.

Offline models would provide an upper bound of the translation quality. As the first work in simultaneous speech-to-speech translation, we explore various simultaneous strategies including wait-k strategy, V-MMA strategy with and without offline knowledge in the direct S2U translation model. We follow the same training setup as Lee et al. (2022), except that a masked attention for our simultaneous policies is used for mode training with partial inputs (Ma et al., 2019a).

Additionally, we also tried to incorporate external guidance for policy learning. We first trained an offline model and exported the alignment labels by finding the most possible target prediction by feeding a partial input. We then leverage the offline labels for V-MMA by minimizing the distance between the alignment matrix α and the offline labels. [2]

			BLEU	CA AL (ms)
Offline	Cascaded (S2T+TTS)		39.5	-
	Direct S2U		37.2	-
Simul	Cascaded (S2T with wait-k + incremental TTS)	k=1	23.9	1317
		k=3	25.4	3109
		k=5	25.9	3004
		k=10	28.1	3476
		k=15	29.2	3803
		k=20	30.1	4099
		k=25	31.4	4460
	Cascaded (S2T with MMA + incremental TTS)	lw=$1e-4$	28.4	3884
		lw=$1e-3$	30.5	3737
		lw=$5e-4$	35.8	4504
	Direct S2U with wait-k	k=5	22.2	1757
		k=10	27.9	3520
		k=15	33.5	4127
		k=20	34.3	4409
	Direct S2U with V-MMA	λ=0.01, w/o label	25.3	3136
		λ=0.01, w/ label	25.9	3215
		λ=0.5, w/o label	33.4	4564
		λ=0.5, w/ label	34	4558

Table 7.1: BLEU scores and latency of models on the Fisher Spanish-English dataset.

			BLEU	CA AL (ms)
Offline	Cascaded (S2T+TTS)		24.4	-
	Direct S2U		23.7	-
Simul	Cascaded (S2T with wait-k + incremental TTS)	k=3	9.7	1094
		k=5	11.7	2892
		k=7	12.8	3108
		k=9	14.3	3159
	Cascaded (S2T with MMA + incremental TTS)	lw=$1.5e-3$	15.8	3249
		lw=$1e-3$	15.9	3283
		lw=$1.2e-3$	15.9	3303
		lw=$8e-4$	16.4	3547
	Direct S2U with wait-k	k=5	9.2	839
		k=10	10.8	2718
		k=15	16.7	3278
		k=20	18.6	4473
	Direct S2U with V-MMA	λ=0.01, w/o label	11.5	2706
		λ=0.01, w/ label	12.2	2927
		λ=0.5, w/o label	18.1	4421
		λ=0.5, w/ label	18.2	4534

Table 7.2: BLEU scores and latency of models on the MuST-C English-Spanish dataset.

7.4 Results

In this section, we report the results of both offline and simultaneous models on Fisher and MuST-C datasets respectively. We compare direct with cascaded approaches to simultaneous translation, and analyze the impact of discontinuity on synthesized speech.

7.4.1 Latency-Quality

We first compare simultaneous translation models on Fisher dataset, and results are shown in Table 7.1.

Cascaded models. Wait-k and MMA strategies demonstrate comparable performance in terms of BLEU and latency.

Direct models. As for direct models, the V-MMA policy controls model by the hyperparameter λ in Equation 7.3. V-MMA policy has similar performance compared with wait-k policy in direct S2S models.

Direct v. cascaded. We include offline speech-to-speech models which provide an upper bound of the translation quality. In the offline setting, the direct S2S model falls behind the cascaded model by 2.3 BLEU. As for the simultaneous setting, latency is an important factor besides the translation quality. With wait-k policy, direct models achieve higher

[2]The results of this setup are not included in the thesis because the work was mostly done by Hongyu Gong from Meta Facebook AI Research.

BLEU than cascaded models in the high-latency setting (> 4000 ms). This demonstrates that direct models have the ability to handle compounding errors while the error propagation hurts the performance of cascaded models. In the low-latency region (< 4000 ms), direct models have comparable BLEU and latency in comparison with cascaded models when wait-k strategy is applied.

We evaluate simultaneous translation models on MuST-C English-Spanish dataset, and summarize their performance in Table 7.2.

Simultaneous policies. For the direct simultaneous model, V-MMA outperforms wait-k given latency lower than 3000 ms, e.g., V-MMA has a BLEU of 11.5 which is higher than wait-k's BLEU of 10.8 with a latency of 2706 ms which is comparable to wait-k's latency of 2718 ms. When it comes to high-latency region, V-MMA has a lower BLEU of 18.1 with 4421-ms latency than wait-k which has a BLEU of 18.6 with 4473-ms latency.

Direct v. cascaded. We again include offline speech-to-speech models which provide an upper bound of the translation quality. In the offline setting, the direct model falls behind the cascaded model by 0.7 BLEU. In the simultaneous setting, both wait-k and V-MMA in direct models have higher BLEU scores in comparison with wait-k and MMA in cascaded models given latency higher than 3000 ms.

7.5 Conclusion

Chapter 8

Conclusion

8.1 Summary of Findings

In this thesis, we conduct research on the end-to-end / direct approach for simultaneous speech translation. The summarization of the main findings and contributions of this thesis are listed as follow.

- In Chapter 2, we formalized the tasks of this thesis, including the definition, challenges and evaluation procedures. We extend existing latency metrics to speech tasks. We develop an open source toolkit, SimulEval (Ma et al., 2020b), for standard and generalized evaluation for simultaneous translation.

- In Chapter 4, we proposed monotonic multihead attention (MMA) to extend mono-

tonic attention to Transformer (Vaswani et al., 2017) to leverage the quality of simultaneous translation model. We proposed two types of MMA models, hard and infinite lookback. We achieve the state-of-the-art performance on quality-latency trade-off compared with baseline.

- In Chapter 5, we adapted the simultaneous policy from text-to-text translation to speech-to-text translation. We analyze the challenge of the adaptation and proposed the concept of pre-decision module to address the issue.

- In Chapter 6, we introduce augmented memory transformer (Wu et al., 2020) to simultaneous translation task to improve the efficiency of simultaneous encoder. Compared with uni-directional masked fully attended baseline, the proposed approach achieved a better computation-aware latency.

- In Chapter 7, we attempt the direct simultaneous speech-to-speech translation based-on recent proposed speech-to-unit (Lee et al., 2022) model. To address the issue that the alignment estimation of MMA is huge on long target speech unit sequence, we proposed the variational monotonic multihead attention. We find that the task is very challenging, where the direct model underperforms the cascaded model.

8.2 Future Works

8.2.1 Direct Simultaneous Speech-to-Speech Translation

The low-latency speech-to-speech (S2S) translation is crucial for lots of applications, such as real-time communications. However, as introduced in chapter 7, the task is very challenging, especially for the direct approach. In this chapter 7, we proposed variational monotonic multihead attention. However, it failed to achieve satisfying quality compared with the cascaded baseline. We proposed the following idea for potential improvement:

- Currently, most MMA-based models have no explicit history dependency. Introducing such independency is important for S2S because we hope the actions to be continues to reduce the discontinuity in the generated speech

- Better training strategies, including data augmentation (such as knowledge distillation), pre-training from offline model and unsupervised encoder, has great potential to improve the translation quality

- New architectures besides encoder-decoder models, such as transuducer-based models, start to show decent performance when massive training data is available. Such new architectures give the new think for the simultaneous translation task.

8.2.2 Improvement on Evaluation

The current evaluation metrics for simultaneous translation are all utterance based. Utterance level latency evaluation on standard test-sets, such as test-COMMON in MuST-C (Di Gangi et al., 2019a) which only has a average length of 6s, can be underestimated. Furthermore, the oracle system delays, which usually used to compute the latency metrics, are usually from an evenly distributed linear-time system. However, for human speech, the information is not evenly distributed. Therefore, we think the following future works on evaluation metrics design can be important:

- A streaming level, segmentation free latency metric.
- A latency metric with consideration on information distribution over speech.
- A standard test-set for streaming translation.
- Evaluation for Read/Write parallel execution.

8.2.3 Incorporation of Human Interpretation

Different from explicit translation, human interpreters usually combine translation and summarization during simultaneous interpretation. which for a long time has been neglected by the machine translation research community during either data collection or evaluation process. In future, we should leverage the human interpretation, such as

- Design systems which combine translation and summarization to generate the results similar to human interpreter.
- Collect human interpreter data for simultaneous model training and evaluation.
- Use human interpreter as the oracle system in the latency calculation for better estimation.
- Conduct human evaluation on the latency of real-time translation

8.2.4 Simultaneous Translation with Multi-Modality

In many applications, such as video chat and TED talks, additional modality, such as video and text are available. Potentially, the data in additional modality can be used for

- Improving the simultaneous policy. Some visual data, such as lip movements, are more deterministic than speech signals.
- Determine the domain of the speech. For instance, the we can infer the domain from the video background to deploy the most in-domain model.
- Improving the speech robustness and Speaker diarisation in multi-speaker senorios.

Appendix A

Appendix of Chapter 4

A.1 Hyperparameters

The hyperparameters we used for offline and monotonic transformer models are defined in Table A.1.

A.2 Detailed results

We provide the detailed results in Figure 4.2 as Table A.2 and Table A.3.

Hyperparameter	WMT15 German-English	IWSLT English-Vietnamese
encoder embed dim	1024	512
encoder ffn embed dim	4096	1024
encoder attention heads	16	4
encoder layers	6	
decoder embed dim	1024	512
decoder ffn embed dim	4096	1024
decoder attention heads	16	4
decoder layers	6	
dropout	0.3	
optimizer	adam	
adam-β	$(0.9, 0.98)$	
clip-norm	0.0	
lr	0.0005	
lr scheduler	inverse sqrt	
warmup-updates	4000	
warmup-init-lr	1e-07	
label-smoothing	0.1	
max tokens	$3584 \times 8 \times 8 \times 2$	16000

Table A.1: Offline and monotonic models hyperparameters.

	BLEU	AP	AL	DAL
λ_{avg}		MMA-IL		
0.05	30.7	0.78	10.91	12.64
0.1	30.5	0.70	7.42	8.82
0.2	30.1	0.63	5.17	6.41
0.3	30.3	0.60	4.18	5.35
0.4	29.2	0.59	3.75	4.90
0.5	26.7	0.59	3.69	4.83
0.75	25.5	0.58	3.40	4.46
1.0	25.1	0.56	3.00	4.03
λ_{var}		MMA-H		
0.1	28.5	0.74	8.94	10.83
0.2	28.9	0.69	6.82	8.622
0.3	29.2	0.64	5.45	7.03
0.4	28.5	0.59	3.90	5.21
0.5	28.5	0.59	3.88	5.19
0.6	29.6	0.56	3.13	4.32
0.7	29.1	0.56	2.93	4.10

Table A.2: Detailed results for MMA-H and MMA-IL on WMT15 DeEn

	BLEU	AP	AL	DAL
λ		MILk		
0.1	24.62	0.71	5.93	7.19
0.2	24.68	0.67	4.90	5.97
0.3	24.31	0.65	4.45	5.43
0.4	23.73	0.64	4.28	5.24
λ_{avg}		MMA-IL		
0.02	28.28	0.76	7.09	8.29
0.04	28.33	0.70	5.44	6.57
0.1	28.42	0.67	4.63	5.65
0.2	28.47	0.63	3.57	4.44
0.3	27.9	0.59	2.98	3.81
0.4	27.73	0.58	2.68	3.46
λ_{var}		MMA-H		
0.02	27.26	0.77	7.52	8.71
0.1	27.68	0.69	5.22	6.31
0.2	28.06	0.63	3.81	4.84
0.4	27.79	0.62	3.57	4.59
0.8	27.95	0.60	3.22	4.19

Table A.3: Detailed results for MILk, MMA-H and MMA-IL on IWSLT15 En-Vi

Bibliography

Ahmed, Zeeshan, Jie Jiang, Julie Carson-Berndsen, Peter Cahill, and Andy Way (Oct. 2012). "Hierarchical Phrase-Based MT for Phonetic Representation-Based Speech Translation". In: *Proceedings of the 10th Conference of the Association for Machine Translation in the Americas: Research Papers*. San Diego, California, USA: Association for Machine Translation in the Americas.

Alabau, Vicente, Alberto Sanchis, and Francisco Casacuberta (Oct. 2007). "Using word posterior probabilities in lattice translation". In: *Proceedings of the Fourth International Workshop on Spoken Language Translation*. Trento, Italy.

Alinejad, Ashkan, Maryam Siahbani, and Anoop Sarkar (Oct. 2018). "Prediction Improves Simultaneous Neural Machine Translation". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 3022–3027.

BIBLIOGRAPHY

Ananthakrishnan, Sankaranarayanan, Wei Chen, Rohit Kumar, and Dennis Mehay (Dec. 2013). "Source aware phrase-based decoding for robust conversational spoken language translation". In: *Proceedings of the 10th International Workshop on Spoken Language Translation: Papers*. Heidelberg, Germany.

Anastasopoulos, Antonios, Loïc Barrault, Luisa Bentivogli, Marcely Zanon Boito, Ondřej Bojar, Roldano Cattoni, Anna Currey, Georgiana Dinu, Kevin Duh, Maha Elbayad, Clara Emmanuel, Yannick Estève, Marcello Federico, Christian Federmann, Souhir Gahbiche, Hongyu Gong, Roman Grundkiewicz, Barry Haddow, Benjamin Hsu, Dávid Javorský, Věra Kloudová, Surafel Lakew, Xutai Ma, Prashant Mathur, Paul McNamee, Kenton Murray, Maria Nădejde, Satoshi Nakamura, Matteo Negri, Jan Niehues, Xing Niu, John Ortega, Juan Pino, Elizabeth Salesky, Jiatong Shi, Matthias Sperber, Sebastian Stüker, Katsuhito Sudoh, Marco Turchi, Yogesh Virkar, Alexander Waibel, Changhan Wang, and Shinji Watanabe (May 2022). "Findings of the IWSLT 2022 Evaluation Campaign". In: *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*. Dublin, Ireland (in-person and online): Association for Computational Linguistics, pp. 98–157.

Anastasopoulos, Antonios, Ondřej Bojar, Jacob Bremerman, Roldano Cattoni, Maha Elbayad, Marcello Federico, Xutai Ma, Satoshi Nakamura, Matteo Negri, Jan Niehues, Juan Pino, Elizabeth Salesky, Sebastian Stüker, Katsuhito Sudoh, Marco Turchi, Alexan-

der Waibel, Changhan Wang, and Matthew Wiesner (Aug. 2021). “FINDINGS OF THE IWSLT 2021 EVALUATION CAMPAIGN”. In: *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*. Bangkok, Thailand (online): Association for Computational Linguistics, pp. 1–29.

Ansari, Ebrahim, Amittai Axelrod, Nguyen Bach, Ondřej Bojar, Roldano Cattoni, Fahim Dalvi, Nadir Durrani, Marcello Federico, Christian Federmann, Jiatao Gu, Fei Huang, Kevin Knight, Xutai Ma, Ajay Nagesh, Matteo Negri, Jan Niehues, Juan Pino, Elizabeth Salesky, Xing Shi, Sebastian Stüker, Marco Turchi, Alexander Waibel, and Changhan Wang (July 2020). “FINDINGS OF THE IWSLT 2020 EVALUATION CAMPAIGN”. In: *Proceedings of the 17th International Conference on Spoken Language Translation*. Online: Association for Computational Linguistics, pp. 1–34.

Arivazhagan, Naveen, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel (July 2019). “Monotonic Infinite Lookback Attention for Simultaneous Machine Translation”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 1313–1323.

Bach, Nguyen, Matthais Eck, Paisarn Charoenpornsawat, Thilo Köhler, Sebastian Stüker, ThuyLinh Nguyen, Roger Hsiao, Alex Waibel, Stephan Vogel, Tanja Schultz, and Alan W. Black (Oct. 2007). “The CMU TransTac 2007 eyes-free two-way speech-to-speech

translation system". In: *Proceedings of the Fourth International Workshop on Spoken Language Translation*. Trento, Italy.

Banerjee, Satanjeev and Alon Lavie (June 2005). "METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments". In: *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*. Ann Arbor, Michigan: Association for Computational Linguistics, pp. 65–72.

Bansal, Sameer, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater (Sept. 2018). "Low-Resource Speech-to-Text Translation". en. In: *Interspeech 2018*. ISCA, pp. 1298–1302.

Baum, Leonard E. and Ted Petrie (1966). "Statistical Inference for Probabilistic Functions of Finite State Markov Chains". In: *The Annals of Mathematical Statistics* 37.6. Publisher: Institute of Mathematical Statistics, pp. 1554–1563.

Berard, Alexandre, Olivier Pietquin, Christophe Servan, and Laurent Besacier (Dec. 2016). *Listen and Translate: A Proof of Concept for End-to-End Speech-to-Text Translation*. arXiv:1612.01744 [cs].

Birch, Alexandra, Miles Osborne, and Philipp Koehn (Oct. 2008). "Predicting Success in Machine Translation". In: *Proceedings of the 2008 Conference on Empirical Methods*

in Natural Language Processing. Honolulu, Hawaii: Association for Computational Linguistics, pp. 745–754.

Bouillon, Pierrette, Manny Rayner, Nikos Chatzichrisafis, Beth Ann Hockey, Marianne Santaholma, Marianne Starlander, Yukie Nakao, Kyoko Kanzaki, and Hitoshi Isahara (May 2005). "A generic multi-lingual open source platform for limited-domain medical speech translation". In: *Proceedings of the 10th EAMT Conference: Practical applications of machine translation*. Budapest, Hungary: European Association for Machine Translation.

Bérard, Alexandre, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin (Apr. 2018). "End-to-End Automatic Speech Translation of Audiobooks". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN: 2379-190X, pp. 6224–6228.

Cettolo, Mauro, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, Rolando Cattoni, and Marcello Federico (Dec. 2016). "The IWSLT 2016 Evaluation Campaign". In: *Proceedings of the 13th International Conference on Spoken Language Translation*. Seattle, Washington D.C: International Workshop on Spoken Language Translation.

Chen, Junkun, Mingbo Ma, Renjie Zheng, and Liang Huang (Aug. 2021). "Direct Simultaneous Speech-to-Text Translation Assisted by Synchronized Streaming ASR". In:

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021. Online: Association for Computational Linguistics, pp. 4618–4624.

Chen, Mia Xu, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes (July 2018). "The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 76–86.

Cherry, Colin and George Foster (May 2019). "Thinking Slow about Latency Evaluation for Simultaneous Machine Translation". In: *arXiv:1906.00048 [cs]*. arXiv: 1906.00048.

Chiu*, Chung-Cheng and Colin Raffel* (Feb. 2018). "Monotonic Chunkwise Attention". en. In:

Cho, Eunah, Than-Le Ha, and Alex Waibel (Dec. 2013). "CRF-based disfluency detection using semantic features for German to English spoken language translation". In: *Proceedings of the 10th International Workshop on Spoken Language Translation: Papers*. Heidelberg, Germany.

Cho, Eunah, Jan Niehues, and Alex Waibel (Dec. 2012). "Segmentation and punctuation prediction in speech language translation using a monolingual translation system". In:

Proceedings of the 9th International Workshop on Spoken Language Translation: Papers. Hong Kong, Table of contents, pp. 252–259.

Cho, Kyunghyun and Masha Esipova (June 2016). "Can neural machine translation do simultaneous translation?" In: *arXiv:1606.02012 [cs]*. arXiv: 1606.02012.

Collobert, Ronan, Christian Puhrsch, and Gabriel Synnaeve (Sept. 2016). *Wav2Letter: an End-to-End ConvNet-based Speech Recognition System*. arXiv:1609.03193 [cs].

Dalvi, Fahim, Nadir Durrani, Hassan Sajjad, and Stephan Vogel (June 2018). "Incremental Decoding and Training Methods for Simultaneous Translation in Neural Machine Translation". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 493–499.

Davis, S. and P. Mermelstein (Aug. 1980). "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28.4. Conference Name: IEEE Transactions on Acoustics, Speech, and Signal Processing, pp. 357–366.

Di Gangi, Mattia A., Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi (June 2019a). "MuST-C: a Multilingual Speech Translation Corpus". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computa-*

tional Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics, pp. 2012–2017.

Di Gangi, Mattia Antonino, Matteo Negri, Roldano Cattoni, Roberto Dessi, and Marco Turchi (Aug. 2019b). "Enhancing Transformer for End-to-end Speech-to-Text Translation". In: *Proceedings of Machine Translation Summit XVII: Research Track*. Dublin, Ireland: European Association for Machine Translation, pp. 21–31.

Dillinger, Mike and Mark Seligman (Sept. 2004). "System description: a highly interactive speech-to-speech translation system". In: *Proceedings of the 6th Conference of the Association for Machine Translation in the Americas: Technical Papers*. Washington, USA: Springer, pp. 58–63.

Do, Quoc Truong, Sakriani Sakti, Graham Neubig, Tomoki Toda, and Satoshi Nakamura (Dec. 2015). "Improving translation of emphasis with pause prediction in speech-to-speech translation systems". In: *Proceedings of the 12th International Workshop on Spoken Language Translation: Papers*. Da Nang, Vietnam, pp. 204–208.

Duong, Long, Antonios Anastasopoulos, David Chiang, Steven Bird, and Trevor Cohn (June 2016). "An Attentional Model for Speech Translation Without Transcription". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, pp. 949–959.

BIBLIOGRAPHY

Graves, Alex, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber (June 2006). "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks". In: *Proceedings of the 23rd international conference on Machine learning*. ICML '06. New York, NY, USA: Association for Computing Machinery, pp. 369–376.

Graves, Alex and Navdeep Jaitly (June 2014). "Towards End-To-End Speech Recognition with Recurrent Neural Networks". en. In: *Proceedings of the 31st International Conference on Machine Learning*. ISSN: 1938-7228. PMLR, pp. 1764–1772.

Grissom II, Alvin, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III (Oct. 2014). "Don't Until the Final Verb Wait: Reinforcement Learning for Simultaneous Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 1342–1352.

Gu, Jiatao, Graham Neubig, Kyunghyun Cho, and Victor O.K. Li (Apr. 2017). "Learning to Translate in Real-time with Neural Machine Translation". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 1053–1062.

BIBLIOGRAPHY

Gu, Liang and Yuqing Gao (Sept. 2004). “On feature selection in maximum entropy approach to statistical concept-based speech-to-speech translation”. In: *Proceedings of the First International Workshop on Spoken Language Translation: Papers*. Kyoto, Japan.

Hannun, Awni, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng (Dec. 2014). *Deep Speech: Scaling up end-to-end speech recognition*. arXiv:1412.5567 [cs].

Hermansky, Hynek (Apr. 1990). “Perceptual linear predictive (PLP) analysis of speech”. In: *The Journal of the Acoustical Society of America* 87.4. Publisher: Acoustical Society of America, pp. 1738–1752.

Hsu, Wei-Ning, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed (2021). “HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29. Conference Name: IEEE/ACM Transactions on Audio, Speech, and Language Processing, pp. 3451–3460.

Huang, Xuedong, Fileno Alleva, Hsiao-Wuen Hon, Mei-Yuh Hwang, Kai-Fu Lee, and Ronald Rosenfeld (Apr. 1993). “The SPHINX-II speech recognition system: an overview”. en. In: *Computer Speech & Language* 7.2, pp. 137–148.

Inaguma, Hirofumi, Shun Kiyono, Kevin Duh, Shigeki Karita, Nelson Yalta, Tomoki Hayashi, and Shinji Watanabe (July 2020). “ESPnet-ST: All-in-One Speech Translation

Toolkit". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Online: Association for Computational Linguistics, pp. 302–311.

Indurthi, Sathish, Houjeung Han, Nikhil Kumar Lakumarapu, Beomseok Lee, Insoo Chung, Sangha Kim, and Chanwoo Kim (May 2020). "End-end Speech-to-Text Translation with Modality Agnostic Meta-Learning". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN: 2379-190X, pp. 7904–7908.

Jain, A. N., A. E. McNair, A. Waibel, H. Saito, A.G. Hauptmann, and J. Tebelskis (July 1991). "Connectionist and Symbolic Processing in Speech-to-Speech Translation: The JANUS System". In: *Proceedings of Machine Translation Summit III: Papers*. Washington DC, USA, pp. 113–117.

Jia, Ye, Melvin Johnson, Wolfgang Macherey, Ron J. Weiss, Yuan Cao, Chung-Cheng Chiu, Naveen Ari, Stella Laurenzo, and Yonghui Wu (May 2019a). "Leveraging Weakly Supervised Data to Improve End-to-end Speech-to-text Translation". In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN: 2379-190X, pp. 7180–7184.

Jia, Ye, Michelle Tadmor Ramanovich, Tal Remez, and Roi Pomerantz (Nov. 2021). "Translatotron 2: Robust direct speech-to-speech translation". en. In:

BIBLIOGRAPHY

Jia, Ye, Ron J. Weiss, Fadi Biadsy, Wolfgang Macherey, Melvin Johnson, Zhifeng Chen, and Yonghui Wu (Sept. 2019b). "Direct Speech-to-Speech Translation with a Sequence-to-Sequence Model". en. In: *Interspeech 2019*. ISCA, pp. 1123–1127.

Kano, Takatomo, Sakriani Sakti, and Satoshi Nakamura (Jan. 2021). "Transformer-Based Direct Speech-To-Speech Translation with Transcoder". In: *2021 IEEE Spoken Language Technology Workshop (SLT)*, pp. 958–965.

Kano, Takatomo, Sakriani Sakti, Shinnosuke Takamichi, Graham Neubig, Tomoki Toda, and Satoshi Nakamura (Dec. 2012). "A method for translation of paralinguistic information". In: *Proceedings of the 9th International Workshop on Spoken Language Translation: Papers*. Hong Kong, Table of contents, pp. 158–163.

Kao, C.-L., S. Saleem, R. Prasad, F. Choi, P. Natarajan, David Stallard, K. Krstovski, and M. Kamali (Oct. 2008). "Rapid development of an English/Farsi speech-to-speech translation system." In: *Proceedings of the 5th International Workshop on Spoken Language Translation: Papers*. Waikiki, Hawaii, pp. 166–173.

Khadivi, Shahram and Zeinab Vakil (Dec. 2012). "Interactive-predictive speech-enabled computer-assisted translation". In: *Proceedings of the 9th International Workshop on Spoken Language Translation: Papers*. Hong Kong, Table of contents, pp. 237–243.

Kingma, Diederik P. and Jimmy Ba (Sept. 2015). "Adam: A Method for Stochastic Optimization". en. In:

BIBLIOGRAPHY

Kitano, H. (June 1991). “Phi DM-Dialog: an experimental speech-to-speech dialog translation system”. In: *Computer* 24.6. Conference Name: Computer, pp. 36–50.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst (June 2007). “Moses: Open Source Toolkit for Statistical Machine Translation”. In: *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*. Prague, Czech Republic: Association for Computational Linguistics, pp. 177–180.

Koehn, Philipp, Franz J. Och, and Daniel Marcu (2003). “Statistical Phrase-Based Translation”. In: *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 127–133.

Kolss, Muntsin, Matthias Wölfel, Florian Kraft, Jan Niehues, Matthias Paulik, and Alex Waibel (Oct. 2008). “Simultaneous German-English lecture translation.” In: *Proceedings of the 5th International Workshop on Spoken Language Translation: Papers*. Waikiki, Hawaii, pp. 174–181.

Kong, Jungil, Jaehyeon Kim, and Jaekyoung Bae (2020). “HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis”. In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 17022–17033.

BIBLIOGRAPHY

Konuma, Tomohiro, Kenji Matsui, Yumi Wakita, Kenji Mizutani, Mitsuru Endo, and Masashi Murata (Mar. 2002). "An experimental mutlilingual bi-directional speech translation system". In: *Proceedings of the 9th Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages: Papers*. Keihanna, Japan.

Kuchaiev, Oleksii, Boris Ginsburg, Igor Gitman, Vitaly Lavrukhin, Carl Case, and Paulius Micikevicius (July 2018). "OpenSeq2Seq: Extensible Toolkit for Distributed and Mixed Precision Training of Sequence-to-Sequence Models". In: *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*. Melbourne, Australia: Association for Computational Linguistics, pp. 41–46.

Kudo, Taku and John Richardson (Nov. 2018). "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Brussels, Belgium: Association for Computational Linguistics, pp. 66–71.

Kumar, Rohit, Sanjika Hewavitharana, Nina Zinovieva, Matthew E. Roy, and Edward Pattison-Gordon (Oct. 2015). "Error-tolerant speech-to-speech translation". In: *Proceedings of Machine Translation Summit XV: Papers*. Miami, USA.

Lavie, Alon, Lori Levin, Robert Frederking, and Fabio Pianesi (Oct. 2002). "The NESPOLE! speech-to-speech translation system". In: *Proceedings of the 5th Conference of the*

Association for Machine Translation in the Americas: System Descriptions. Tiburon, USA: Springer, pp. 240–243.

Lavie, Alon, Lori Levin, Alex Waibel, Donna Gates, Marsal Gavalda, and Laura Mayfield (Oct. 1996). "JANUS: multi-lingual translation of spontaneous speech in limited domain". In: *Conference of the Association for Machine Translation in the Americas*. Montreal, Canada.

Lawson, D., C. Chiu, G. Tucker, C. Raffel, K. Swersky, and N. Jaitly (Apr. 2018). "Learning Hard Alignments with Variational Inference". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN: 2379-190X, pp. 5799–5803.

Lee, Ann, Peng-Jen Chen, Changhan Wang, Jiatao Gu, Sravya Popuri, Xutai Ma, Adam Polyak, Yossi Adi, Qing He, Yun Tang, Juan Pino, and Wei-Ning Hsu (May 2022). "Direct Speech-to-Speech Translation With Discrete Units". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, pp. 3327–3339.

Lee, K.-F., H.-W. Hon, and R. Reddy (Jan. 1990). "An overview of the SPHINX speech recognition system". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 38.1. Conference Name: IEEE Transactions on Acoustics, Speech, and Signal Processing, pp. 35–45.

BIBLIOGRAPHY

Lewis, William (Nov. 2015). “Skype Translator: Breaking down language and hearing barriers. A behind the scenes look at near real-time speech translation”. In: *Proceedings of Translating and the Computer 37*. London, UK: AsLing.

Lim, Daniel Chung Yong, Ian Lane, and Alex Waibel (Dec. 2010). “Real-time spoken language identification and recognition for speech-to-speech translation”. In: *Proceedings of the 7th International Workshop on Spoken Language Translation: Papers*. Paris, France, pp. 307–312.

Liu, Danni, Changhan Wang, Hongyu Gong, Xutai Ma, Yun Tang, and Juan Pino (July 2022). *From Start to Finish: Latency Reduction Strategies for Incremental Speech Synthesis in Simultaneous Speech-to-Speech Translation*. arXiv:2110.08214 [cs, eess].

Liu, Yuchen, Hao Xiong, Jiajun Zhang, Zhongjun He, Hua Wu, Haifeng Wang, and Chengqing Zong (Sept. 2019). “End-to-End Speech Translation with Knowledge Distillation”. en. In: *Interspeech 2019*. ISCA, pp. 1128–1132.

Luo, Y., C. Chiu, N. Jaitly, and I. Sutskever (Mar. 2017). “Learning online alignments with continuous rewards policy gradient”. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN: 2379-190X, pp. 2801–2805.

Luong, Thang, Hieu Pham, and Christopher D. Manning (Sept. 2015). “Effective Approaches to Attention-based Neural Machine Translation”. In: *Proceedings of the 2015 Conference*

on Empirical Methods in Natural Language Processing. Lisbon, Portugal: Association for Computational Linguistics, pp. 1412–1421.

Ma, Mingbo, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang (July 2019a). "STACL: Simultaneous Translation with Implicit Anticipation and Controllable Latency using Prefix-to-Prefix Framework". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 3025–3036.

Ma, Mingbo, Baigong Zheng, Kaibo Liu, Renjie Zheng, Hairong Liu, Kainan Peng, Kenneth Church, and Liang Huang (Nov. 2020a). "Incremental Text-to-Speech Synthesis with Prefix-to-Prefix Framework". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, pp. 3886–3896.

Ma, Xutai, Mohammad Javad Dousti, Changhan Wang, Jiatao Gu, and Juan Pino (Oct. 2020b). "SIMULEVAL: An Evaluation Toolkit for Simultaneous Translation". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, pp. 144–150.

BIBLIOGRAPHY

Ma, Xutai, Juan Pino, and Philipp Koehn (Dec. 2020). “SimulMT to SimulST: Adapting Simultaneous Text Translation to End-to-End Simultaneous Speech Translation”. In: *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*. Suzhou, China: Association for Computational Linguistics, pp. 582–587.

Ma, Xutai, Juan Miguel Pino, James Cross, Liezl Puzon, and Jiatao Gu (Sept. 2019b). “Monotonic Multihead Attention”. en. In:

Ma, Xutai, Yongqiang Wang, Mohammad Javad Dousti, Philipp Koehn, and Juan Pino (2021). “Streaming Simultaneous Speech Translation with Augmented Memory Transformer”. In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7523–7527.

Maergner, Paul, Kevin Kilgour, Ian Lane, and Alex Waibel (Dec. 2011). “Unsupervised vocabulary selection for simultaneous lecture translation”. In: *Proceedings of the 8th International Workshop on Spoken Language Translation: Papers*. San Francisco, California, pp. 214–221.

Morimoto, Tsuyoshi, Kiyohiro Shikano, Hitoshi Iida, and Akira Kurematsu (1990). “Integration of speech recognition and language processing in spoken language translation system (SL-TRANS).” In: *ICSLP*.

BIBLIOGRAPHY

Murthy, H.A., F. Beaufays, L.P. Heck, and M. Weintraub (Sept. 1999). "Robust text-independent speaker identification over telephone channels". In: *IEEE Transactions on Speech and Audio Processing* 7.5. Conference Name: IEEE Transactions on Speech and Audio Processing, pp. 554–568.

Ney, H. (Mar. 1999). "Speech translation: coupling of recognition and translation". In: *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)*. Vol. 1. ISSN: 1520-6149, 517–520 vol.1.

Oliver, B. M. (July 1952). "Efficient coding". In: *The Bell System Technical Journal* 31.4. Conference Name: The Bell System Technical Journal, pp. 724–750.

Ott, Myle, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli (June 2019). "fairseq: A Fast, Extensible Toolkit for Sequence Modeling". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 48–53.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu (July 2002). "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, pp. 311–318.

BIBLIOGRAPHY

Patry, Alexandre and Philippe Langlais (May 2008). “MISTRAL: a Statistical Machine Translation Decoder for Speech Recognition Lattices”. In: *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC’08)*. Marrakech, Morocco: European Language Resources Association (ELRA).

Peitz, Stephan, Simon Wiesler, Markus Nußbaum-Thom, and Hermann Ney (Dec. 2012). “Spoken language translation using automatically transcribed text in training”. In: *Proceedings of the 9th International Workshop on Spoken Language Translation: Papers*. Hong Kong, Table of contents, pp. 276–283.

Pino, Juan, Liezl Puzon, Jiatao Gu, Xutai Ma, Arya D. McCarthy, and Deepak Gopinath (Nov. 2019). “Harnessing Indirect Training Data for End-to-End Automatic Speech Translation: Tricks of the Trade”. In: *Proceedings of the 16th International Conference on Spoken Language Translation*. Hong Kong: Association for Computational Linguistics.

Pino, Juan, Qiantong Xu, Xutai Ma, Mohammad Javad Dousti, and Yun Tang (Oct. 2020). “Self-Training for End-to-End Speech Translation”. en. In: *Interspeech 2020*. ISCA, pp. 1476–1480.

Polyak, Adam, Lior Wolf, Yossi Adi, Ori Kabeli, and Yaniv Taigman (June 2021). “High Fidelity Speech Regeneration with Application to Speech Enhancement”. In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN: 2379-190X, pp. 7143–7147.

BIBLIOGRAPHY

Portnoff, M. (June 1981). “Time-scale modification of speech based on short-time Fourier analysis”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 29.3. Conference Name: IEEE Transactions on Acoustics, Speech, and Signal Processing, pp. 374–390.

Post, Matt (Oct. 2018). “A Call for Clarity in Reporting BLEU Scores”. In: *Proceedings of the Third Conference on Machine Translation: Research Papers*. Brussels, Belgium: Association for Computational Linguistics, pp. 186–191.

Post, Matt, Gaurav Kumar, Adam Lopez, Damianos Karakos, Chris Callison-Burch, and Sanjeev Khudanpur (2013). “Improved Speech-to-Text Translation with the Fisher and Callhome Spanish–English Speech Translation Corpus”. en. In: *IWSLT 2013*, p. 7.

Povey, Daniel, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely (2011). *The Kaldi Speech Recognition Toolkit*. en. Conference Name: IEEE 2011 Workshop on Automatic Speech Recognition and Understanding Number: CONF Publisher: IEEE Signal Processing Society.

Prasad, Rohit, Rohit Kumar, Sankaranarayanan Ananthakrishnan, Wei Chen, Sanjika Hewavitharana, Matthew Roy, Frederick Choi, Aaron Challenner, Enoch Kan, Arvid Neelakantan, and Prem Natarajan (Dec. 2012). “Active error detection and resolution

for speech-to-speech translation". In: *Proceedings of the 9th International Workshop on Spoken Language Translation: Papers*. Hong Kong, Table of contents, pp. 150–157.

Pérez, Alicia, Víctor Guijarrubia, Raquel Justo, M. Inés Torres, and Francisco Casacuberta (Oct. 2007). "A comparison of linguistically and statistically enhanced models for speech-to-speech machine translation". In: *Proceedings of the Fourth International Workshop on Spoken Language Translation*. Trento, Italy.

Pérez, Alicia, María Inés Torres, and Francisco Casacuberta (May 2010). "Potential scope of a fully-integrated architecture for speech translation". In: *Proceedings of the 14th Annual conference of the European Association for Machine Translation*. Saint Raphaël, France: European Association for Machine Translation.

Raffel, Colin, Minh-Thang Luong, Peter J. Liu, Ron J. Weiss, and Douglas Eck (Aug. 2017). "Online and linear-time attention by enforcing monotonic alignments". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, pp. 2837–2846.

Raybaud, Sylvain, David Langlois, and Kamel Smaïli (Sept. 2011). "Broadcast news speech-to-text translation experiments". In: *Proceedings of Machine Translation Summit XIII: System Presentations*. Xiamen, China.

BIBLIOGRAPHY

Rayner, Manny, Pierrette Bouillon, and David Carter (Nov. 1995). “Using corpora to develop limited-domain speech translation systems”. In: *Proceedings of Translating and the Computer 17*. London, UK: Aslib.

Ren, Yi, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu (Mar. 2022). “FastSpeech 2: Fast and High-Quality End-to-End Text to Speech”. en. In:

Ren, Yi, Jinglin Liu, Xu Tan, Chen Zhang, Tao Qin, Zhou Zhao, and Tie-Yan Liu (July 2020). “SimulSpeech: End-to-End Simultaneous Speech to Text Translation”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 3787–3796.

Ruiz, Nicholas and Marcello Federico (Oct. 2014). “Assessing the impact of speech recognition errors on machine translation quality”. In: *Proceedings of the 11th Conference of the Association for Machine Translation in the Americas: MT Researchers Track*. Vancouver, Canada: Association for Machine Translation in the Americas, pp. 261–274.

Salesky, Elizabeth, Matthias Sperber, and Alan W Black (July 2019). “Exploring Phoneme-Level Speech Representations for End-to-End Speech Translation”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 1835–1841.

Sennrich, Rico, Barry Haddow, and Alexandra Birch (Aug. 2016). “Neural Machine Translation of Rare Words with Subword Units”. In: *Proceedings of the 54th Annual Meeting*

of the Association for Computational Linguistics (Volume 1: Long Papers). Berlin, Germany: Association for Computational Linguistics, pp. 1715–1725.

Shimohata, Mitsuo, Eiichiro Sumita, and Yuji Matsumoto (Sept. 2003). "Example-based rough translation for speech-to-speech translation". In: *Proceedings of Machine Translation Summit IX: Papers*. New Orleans, USA.

Snover, Matthew, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul (Aug. 2006). "A Study of Translation Edit Rate with Targeted Human Annotation". In: *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*. Cambridge, Massachusetts, USA: Association for Machine Translation in the Americas, pp. 223–231.

Stephenson, Brooke, Thomas Hueber, Laurent Girin, and Laurent Besacier (Aug. 2021). "Alternate Endings: Improving Prosody for Incremental Neural TTS with Predicted Future Text Input". en. In: *Interspeech 2021*. ISCA, pp. 3865–3869.

Stoian, Mihaela C., Sameer Bansal, and Sharon Goldwater (May 2020). "Analyzing ASR Pretraining for Low-Resource Speech-to-Text Translation". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN: 2379-190X, pp. 7909–7913.

BIBLIOGRAPHY

Sudoh, Katsuhito, Takatomo Kano, Sashi Novitasari, Tomoya Yanagita, Sakriani Sakti, and Satoshi Nakamura (Nov. 2020). *Simultaneous Speech-to-Speech Translation System with Neural Incremental ASR, MT, and TTS*. arXiv:2011.04845 [cs].

Tomita, Masaru, Hideto Tomabechi, and Hiroaki Saito (1990). "Speech Trans: An Experimental Real-Time Speech-To-Speech Translation System". In:

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc.

Wahlster, Wolfgang (July 1993). "Verbmobil: Translation of Face-To-Face Dialogs". In: *Proceedings of Machine Translation Summit IV*. Kobe, Japan, pp. 127–136.

Wang, Changhan, Yun Tang, Xutai Ma, Anne Wu, Dmytro Okhonko, and Juan Pino (Dec. 2020a). "Fairseq S2T: Fast Speech-to-Text Modeling with Fairseq". In: *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: System Demonstrations*. Suzhou, China: Association for Computational Linguistics, pp. 33–39.

Wang, Chengyi, Yu Wu, Shujie Liu, Zhenglu Yang, and Ming Zhou (Apr. 2020b). "Bridging the Gap between Pre-Training and Fine-Tuning for End-to-End Speech Translation". en.

In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.05. Number: 05, pp. 9161–9168.

Wang, Minghan, Jiaxin Guo, Yinglu Li, Xiaosong Qiao, Yuxia Wang, Zongyao Li, Chang Su, Yimeng Chen, Min Zhang, Shimin Tao, Hao Yang, and Ying Qin (May 2022). "The HW-TSC's Simultaneous Speech Translation System for IWSLT 2022 Evaluation". In: *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*. Dublin, Ireland (in-person and online): Association for Computational Linguistics, pp. 247–254.

Wasyliw, Boh and Douglas Clarke (Nov. 1994). "Natural language analysis and machine translation in pilot-ATC communication". In: *Proceedings of the Second International Conference on Machine Translation: Ten years on*. Cranfield University, UK.

Watanabe, Shinji, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai (Mar. 2018). *ESPnet: End-to-End Speech Processing Toolkit*. arXiv:1804.00015 [cs].

Weiss, Ron J., Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen (Aug. 2017). "Sequence-to-Sequence Models Can Directly Translate Foreign Speech". en. In: *Interspeech 2017*. ISCA, pp. 2625–2629.

BIBLIOGRAPHY

Woszczcyna, Monika, Matthew Broadhead, Donna Gates, Marsal Gavaldá, Alon Lavie, Lori Levin, and Alex Waibel (Oct. 1998). "A modular approach to spoken language translation for large domains". In: *Proceedings of the Third Conference of the Association for Machine Translation in the Americas: Technical Papers*. Langhorne, PA, USA: Springer, pp. 31–49.

Wu, Chunyang, Yongqiang Wang, Yangyang Shi, Ching-Feng Yeh, and Frank Zhang (Oct. 2020). "Streaming Transformer-Based Acoustic Models Using Self-Attention with Augmented Memory". en. In: *Interspeech 2020*. ISCA, pp. 2132–2136.

Zhang, Ruiqiang, Genichiro Kikui, and Hirofumi Yamamoto (Oct. 2005). "Using multiple recognition hypotheses to improve speech translation". In: *Proceedings of the Second International Workshop on Spoken Language Translation*. Pittsburgh, Pennsylvania, USA.

Zhang, Ruiqiang, Genichiro Kikui, Hirofumi Yamamoto, and Wai-Kit Lo (Oct. 2005). "A decoding algorithm for word lattice translation in speech translation". In: *Proceedings of the Second International Workshop on Spoken Language Translation*. Pittsburgh, Pennsylvania, USA.

Zheng, Baigong, Renjie Zheng, Mingbo Ma, and Liang Huang (Nov. 2019a). "Simpler and Faster Learning of Adaptive Policies for Simultaneous Translation". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the*

9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics, pp. 1349–1354.

Zheng, Baigong, Renjie Zheng, Mingbo Ma, and Liang Huang (July 2019b). "Simultaneous Translation with Flexible Policy via Restricted Imitation Learning". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 5816–5822.

Zheng, Renjie, Mingbo Ma, Baigong Zheng, Kaibo Liu, Jiahong Yuan, Kenneth Church, and Liang Huang (Nov. 2020). "Fluent and Low-latency Simultaneous Speech-to-Speech Translation with Self-adaptive Training". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, pp. 3928–3937.