End-to-End Speech Recognition on Conversations

Submitted in partial fulfillment of the requirements for
the degree of
Doctor of Philosophy
in
Department of Electrical and Computer Engineering

Suyoun Kim

B.S., Engineering, Konkuk University
M.S., Computer Science, Georgia Institute of Technology
M.S., Language Technologies, Carnegie Mellon University

Carnegie Mellon University
Pittsburgh, PA

December 2019

Acknowledgements

Without my advisor Florian Metze, co-advisor Richard M. Stern, and former advisor Ian Lane, this thesis would definitely have not been possible. Thank you for providing valuable guidance, and support throughout my Ph.D. journey at CMU.

I would like to thank Bhiksha Raj, Shinji Watanabe, and Mike Seltzer for serving on the thesis committee and providing helpful comments on the thesis. I would like to thank my collaborators in MERL and MSR, Takaaki, Jinyu, and many more for giving great opportunities to work with.

I would like to thank Heewon for encouraging and supporting me during the Samsung Fellowship that ultimately led me to come to CMU. I would like to thank CMLH for supporting one year Ph.D. research.

I would like to thank people at CMU for being great friends and mentors: Madhavi, my former M.S. advisor, William and Bing, great seniors and my role models, Wonkyum, Jungsuk, Akshay, Guan-Lin in Silicon Valley speech group, Siddharth, Yun, Zhong, Ramon, Eric, Xinjian, Jessica, Shruti, Elizabeth, Roshan in Sin-bad group, Rita, Benjamin, Abelino, Anjali, Raymond, Wenbo Liu, Wenbo Zhao, Anurag, Yang, Yandong, and many more in Robust-MLSP speech group and Sphinx group, Karen and Willie in Group-X.

I would like to thank my Pittsburgh family for making my Pittsburgh life happy: Min Kyung, Eunyong, Jungeun, Sungho, Shane, Deagun, Junsung, Serim, Jiyeon Kim, Seungmoon, Jaejun, Jay-yoon, Eunjin, Euiwoong, Hyeju, Yongsuk, Jihoon, Joohyung, Jooli, Hyokyung, Haewon, Kiwan, Jiyeon Lee, Jisu, Gihyuk, Philgoo, Jinhee. Especially, I would like to thank Min Kyung, for being my unofficial advisor in academic career, social life, healthier life, etc.

I would like to thank my lifelong friends, Jooyoung, Ah Jin, Nari, Hyoyoung, Jihea, Boyeon, Seoyeon, Hyunna, Esther, Miso, who despite the distance, supported me all these years.

Finally, I would like to thank my family: My mom, my dad, and my brother for giving endless support and encouragement for my entire life.

Abstract

Processing of conversations is a core technique in conversational AI. However, current speech recognition solutions, even the state-of-the-art systems, model a single, isolated utterance, not an entire conversation. These systems are therefore unable to use potentially important contextual information that spans across multiple utterances or speakers in a conversation. This thesis focuses on designing an End-to-End speech recognition system that processes entire conversations. To achieve this goal, I propose three novel techniques: 1) an efficient way to preserve long conversational contexts by creating a context encoder that maps spoken utterance histories to a single context vector; 2) an effective way to integrate conversational contexts into End-to-End models using a gating mechanism; and 3) various methods to encode conversational contexts by using previously spoken utterances and augmenting with world knowledge using external linguistic resources (e.g. BERT, fastText). I show accuracy improvements with three different large corpora, Switchboard (300 hours), Fisher (2,000 hours), and Medical conversation (1,700 hours), and share the analysis to demonstrate the effectiveness of my approach. This thesis will provide insight into designing conversational speech recognition systems and spoken language understanding systems, which are becoming increasingly important as voice-driven device interfaces become mainstream.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

As voice-driven interfaces to devices become mainstream, many real-world applications that can recognize and translate spoken language are becoming increasingly important. Especially, conversational AI applications, such as spoken dialog systems, AI assistants, and audio/video recorded meeting summarization, require a machine to recognize and understand long conversations.

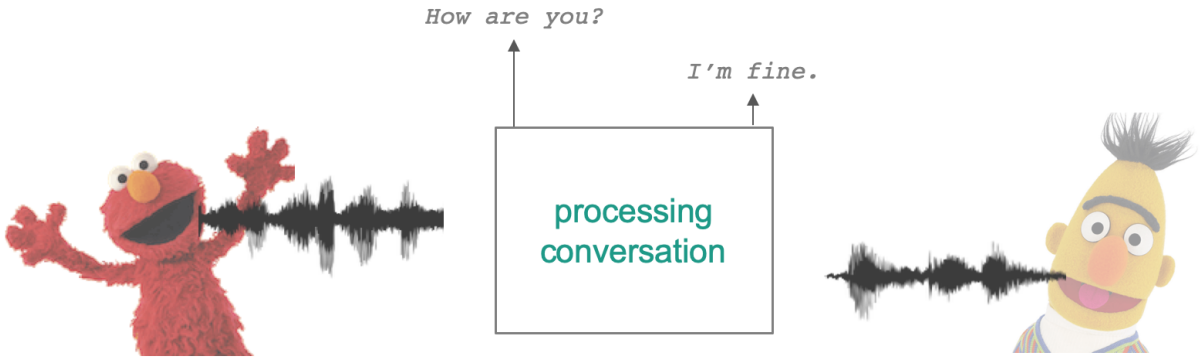


Figure 1.1: Processing of conversations of spoken dialog systems.

Conversational automatic speech recognition (ASR) that takes a spoken audio input and generates transcription is a core technique in such applications (in Figure 1.1). Compared to general speech recognition, conversational speech recognition techniques require to recognize not just a single isolated utterance, but multiple utterances in the entire conversation. The focus of this thesis is on conversational speech recognition.

In a long conversation, there exists global conversational-level consistency, especially in terms of conversation topic and tendency that the same or semantically related words and phrases occurring across utterances. For example, when the user requested to play specific music in the conversation with AI assistants, requests or questions related to the music tend to be followed, and in the conversation between doctor and patients, medical terms such as medicine or disease names tend to be spoken. Even in the chit-chat types of conversation, the same words/phrases tend to be re-used. Therefore, an utterance can be more recognizable based on the previous utterance history, and modeling such conversational context is the key to conversational speech recognition tasks.

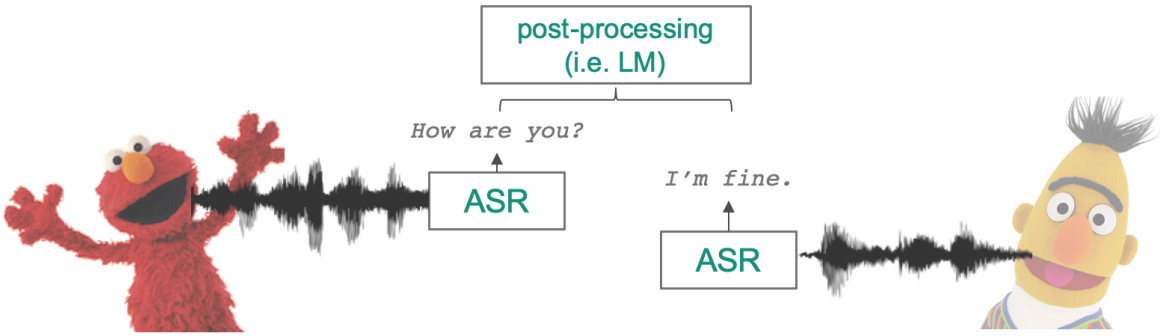


Figure 1.2: Current ASR solution in processing conversations.

However, current automatic speech recognition (ASR) solutions, even state-of-the-art systems, are modeling fragments, not entire conversations. As Figure 1.2 shows, to make building systems computationally feasible, long conversations are typically split into shorter blocks with

utterance-level audio and systems that are built at such individual, isolated utterance level. This modeling process makes the model lose important conversational context information, which can help improve speech recognition performance.

There have been many studies that have attempted to use global, conversational level context information [1, 2, 3, 4, 5, 6] and modeling entire conversations, or entire documents and has been shown the importance of exploiting context information in processing a conversation or document. All of these models were however developed on text data for dialog models or language models. Such a language model that captures long context information is trained separately and then the context information can be added to the ASR system only as a post-fix by re-scoring with the ASR outputs. The conversational ASR systems based on such disjoint components do not fully exploit the useful context information.

The recent End-to-End speech recognition approach [7, 8, 9, 10, 11, 12, 13] integrates all available information within a single neural network model, and directly transcribes speech to text. Unlike the traditional speech recognition systems that are complicated and composed of multiple components: pronunciation models, acoustic models, language models, the End-to-End speech recognition models do not use a disjoint training procedure and are optimized towards the final objectives. This property of End-to-End modeling motivates us to include conversational context information within the End-to-End speech recognition model and optimize towards final objective of interest.

1.2 Thesis Statement

With the previous motivations, I propose a novel End-to-End conversational speech recognition framework that processes entire conversations with modeling "**conversational context**". I define "conversational context", higher knowledge that spans across multiple utterances, which is

helpful to process long conversations. The main objective of this thesis is to show that:

Learning a long conversational context beyond utterance level
is feasible and useful for better processing of long conversations.

To achieve this goal, I address the following main research questions:

- How to preserve long conversational contexts?
- How to integrate conversational context information?
- How to encode conversational context information?
- How to validate the effectiveness of the conversational context?

The thesis also shows accuracy improvements with three different large corpora, Switchboard (300 hours), Fisher (2,000 hours), and Medical conversation (1,700 hours), and present my analysis to demonstrate the effectiveness of my proposed framework.

1.3 Thesis Contributions

To the best of my knowledge, this is the first work to model entire conversations, rather than isolated utterance, by combining acoustic and language information in End-to-End manner. My thesis shows that modeling entire conversations is feasible and useful for better processing of long conversations.

The technical contributions of this thesis are the followings:

1. **An improved way to build End-to-End speech recognition models** - I jointly train the End-to-End ASR with CTC and Seq2Seq objectives to achieve better accuracy and faster convergence than vanilla End-to-End ASR. My joint CTC/Seq2Seq End-to-End ASR approach has been available publicly [14] and it has used and extended in many other follow up studies [15, 16].

2. **An efficient way to preserve long conversational contexts by creating a context encoder that maps spoken utterance histories to a single context vector** - I serialize the training utterances based on their onset time, then shuffle them at dialog-level. I extract/detach/cache context embeddings similar to a truncated backpropagation through time (BPTT). My proposed method allows to model entire conversations with the computational efficiency and avoiding GPU memory issues.

3. **An effective way to integrate conversational contexts into End-to-End models using a gating mechanism** - I create a gated contextual decoder that can recognize the current utterance conditioning on the context information. I introduce a contextual gating mechanism to integrate multiple types of embeddings: word, speech, and conversational context embeddings effectively. My proposed gated decoder method can decide how to weigh different embeddings and can shape information flow using multiplicative interactions and can achieve better accuracy performance rather than simple concatenation method.

4. **Various methods to encode conversational contexts by using previously spoken utterances and augmenting with world knowledge using external linguistic resources** - I create a conversational context encoder to map the multiple preceding utterance histories into a fixed-length context vector. My proposed context encoding methods can also use either the generated sentences or the true preceding utterance during training similar to a sampling strategy. My methods are augmented *world knowledge* by using the external word (fastText) and/or sentence embeddings (BERT) so that it can represent conversational context better. My methods can also model an interaction between two speakers based on speaker-turn information by using speaker-specific cross-attention that can look at the output of the other speaker side as well as the output of the current speaker.

5. **Analytical methods to demonstrate the effectiveness of my conversational context models** - In addition to simply comparing the Word Error Rate (WER) with standard End-to-End speech recognition systems, I also compare the performance of my model with the

context vector generated from *Oracle* utterances and *Random* utterances. This method can verify that my model's benefit does not come from randomness or regularization. I also create a "conversational similarity" scoring function that shows my context models work well where the historical utterances and the current utterance are similar to each other which confirms the hypothesis of my thesis.

1.4 Thesis Outline

Chapter 2 will review previous work for two main End-to-End speech recognition approaches: Connectionist Temporal Classification (CTC) and Attention-based Encoder-Decoder (Seq2Seq) and context modeling approaches. I will review the strengths and weaknesses of the two approaches. This will give motivation of my proposed End-to-End speech recognition approach, joint CTC/Seq2Seq model in Chapter 3. I show my model provides fast convergence as well as improved accuracy than the previous End-to-End speech recognition system. In Chapter 4, I present my proposed conversational End-to-End automatic speech recognition (ASR) systems that model entire conversations. I show two novel methods: 1) an efficient way to preserve long conversational contexts by creating a context encoder that maps spoken utterance histories to a single context vector, and 2) an effective way to integrate conversational contexts into End-to-End models using a gating mechanism. In Chapter 6, I present various methods to encode conversational contexts by using previous spoken utterances and augmenting with world knowledge using external linguistic resources (e.g. BERT, fastText). I also show a method that can look at the output of the other speaker side as well as the output of the current speaker to learn an interaction between two speakers based on speaker-turn information. In Chapter 7, I show a series of experiments on three different large corpora, Switchboard (300 hours), Fisher (2,000

hours), and Medical conversation (1,700 hours). I will share my analyses to demonstrate the effectiveness of my proposed framework, and in which conditions my context model works well. I will finally close up with a conclusion and future work in Chapter 9.

Chapter 2

Background

This chapter first reviews the conventional architecture for conversational speech recognition systems (Section 2.1). We discuss the limitations of the current systems, and give motivations for the overall thesis topic, End-to-End learning on entire conversations. Second, we review two main End-to-End learning approaches: CTC and Seq2Seq models, and their complementary characteristics (in Section 2.2). These reviews motivate us for proposing my joint CTC/Seq2Seq models in Chapter 3. Then, we review several literatures that attempts to model context information, and discuss their limitations which gives motivation for Chapter 4 and 6. Finally, we review popular methods to integrate different types of input resources effectively: Attention/Gating mechanism (in Section 2.4). This gives motivations to propose my gated contextual decoder described in Chapter 5.

2.1 Conversational ASR

Figure 2.1 shows the multiple steps of the system building process for typical conversational ASR.

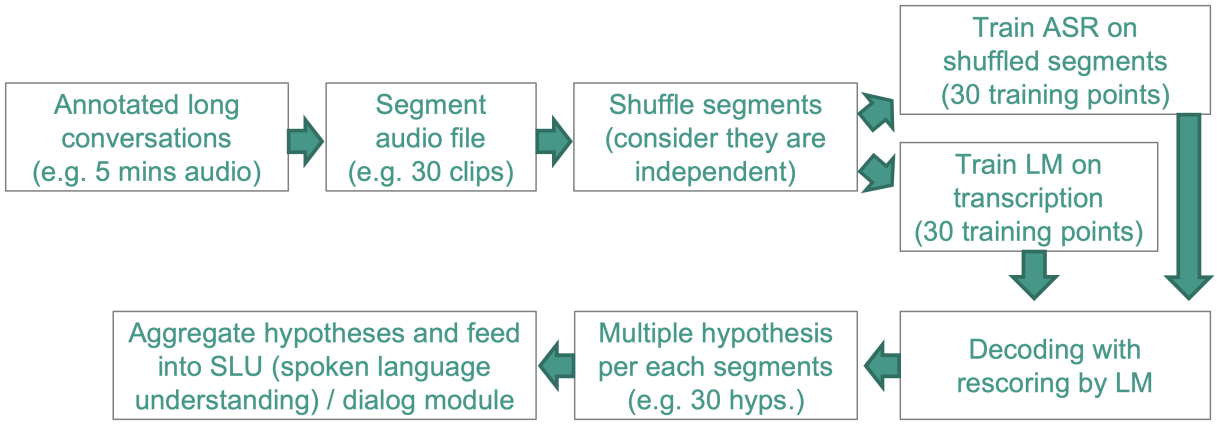


Figure 2.1: Overall system building process of typical conversational ASR.

Segmentation

The long audio file is first split into multiple, small chunk of segments/utterances. The segments are then shuffled and the ASR model is trained on such shuffled, isolated utterances assuming that each utterance is independent of each other. This segmentation/shuffling procedure makes training the speech recognition models computationally feasible. I will show that training the speech recognition models on an entire conversation is computationally infeasible by conducting a simple experiment in Chapter 4. For example, 5 minutes long audio file is split into 30 sequential 10-sec long short audio clips.

Feature computation

Once we obtain multiple audio clips and each audio clip is processed and converted to a sequence of frames of features. For example, log-mel Filterbanks (Fbanks), Mel-Frequency Cepstral Coefficients (MFCCs) [17], Perceptual Linear Prediction (PLP) [18], and Power-normalized Cepstral Coefficients (PNCC) [19] by a signal processing frontend.

Modeling

Figure 2.2 shows the overall architecture of traditional ASR models.

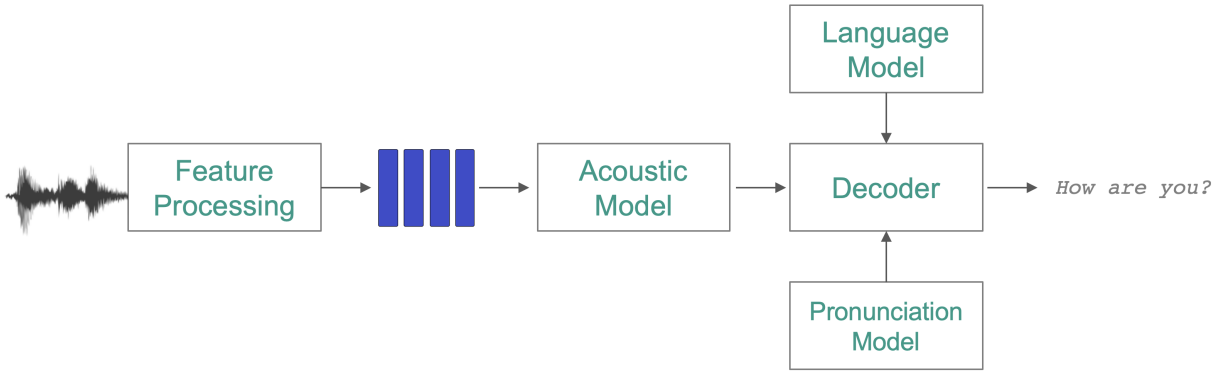


Figure 2.2: The overall architecture of traditional ASR.

Let we have a dialog d, and the dialog is split into N number of segments. Let $x^n = (x_1, ..., x_T)$ is T-length of the sequence of acoustic feature of n-th utterance. Let $w^n = (w_1, ..., w_U)$ is U-length of the sequence of words or characters or sub-words of n-th utterance. The typical ASR system models the probability distributions of a sequence of words w, given a sequence of acoustic feature x, and it is decomposed into two terms using Bayes' rule, an acoustic model $p(x|w)$ and a language model $p(w)$:

$$p(w|x) = \frac{p(x|w)p(w)}{p(w)} \propto p(x|w)p(w) \tag{2.1}$$

The parameter of these models $p(x|w)$ and $p(w)$ trained separately.

Acoustic Models

The acoustic model is using deep neural networks and hidden Markov models (HMM) [20]. Deep Neural Network (DNN) [21, 22, 23, 24, 25], Convolutional Neural Network (CNN) [26, 27, 28, 29], or Long Short-Term Memory (LSTM) which is variants of Recurrent Neural Network (RNN) [30, 31, 32, 33, 34] are used to map an acoustic frame x_t into a phonetic state posterior

q_t at each input time t (in Equation 2.2):

$$p(q_t|x_t) = \texttt{AcousticModel}(x_t) \tag{2.2}$$

Prior to this acoustic modeling procedure, the output targets of the neural network models, the frame-level phonetic state sequence $q_{1:T}$, is generated by HMM and Gaussian Mixture Model (GMM) in an ad-hoc training methods. GMM models an acoustic feature at the frame-level $x_{1:T}$ and the HMM estimates the most probable phonetic state sequence $q_{1:T}$. The acoustic model is optimized with the cross entropy error which is phonetic classification error per frame.

Pronunciation Dictionary

$$w = \texttt{PronunciationDictionary}(q) \tag{2.3}$$

The pronunciation dictionary maps between the phonetic sequence q, which is a minimal unit of sounds, and the word sequence w (in Equation 2.3). For example, English words are represented as 39 different phonemes, or triphones which considering the left and the right phoneme contexts. The pronunciation dictionary enables to find the word from acoustics fast by constraining the search space of decoder, however, it assumes the several limited phonemes can cover all variety of pronunciations of a word (across all different accents, speaking styles, etc). The pronunciation dictionary is generally handcrafted with linguistic expertise and rarely updated, in case of rare words or new words, such pronunciation is not available. Even worse, for second- and third-tier languages, such pronunciation resources may be unavailable or limited.

Language Models

$$p(w_u|w_{<u}) = \texttt{LanguageModel}(w_{<u}) \tag{2.4}$$

The language model $p(w)$ is modeling the most probable word sequences independent of the acoustics (in Equation 2.4 [1]. The RNN or LSTM (typically) is widely used for the architecture of the language models since they can capture long term dependencies rather than traditional n-gram models, which is based on Markovian assumption and limited to conditioning on a specific n-range of word history. The language model is also optimized separately with the different objective, perplexity. This disjoint procedure limits the other components, the acoustic model, cannot fully exploit the linguistic information or linguistic context information.

Decoder

Once all the separate models are optimized, the decoder tries to find the best hypothesis for each utterance that gives the highest probability. The decoder combines all other components the acoustic model, language model, and pronunciation dictionary to find the best single utterance w given acoustic inputs x.

$$\hat{w} = \texttt{argmax}_w p(x|w)p(w) \tag{2.5}$$

Post-processing

The multiple hypotheses $\hat{w}^1, \cdots, \hat{w}^N$ of the dialog d are then passed to the spoken language understanding (SLU) or dialog modules which are also optimized disjointly. This post-processing performed by separately trained models and then the user's specific request finally is completed in such subsequent modules such as language model.

However, with such a system building process of typical conversational ASR, the systems cannot model the context information, which is beyond the utterance-level, because they model a single, isolated utterance (in Equation 2.2 and 2.4). Although modeling dialog-level context information can be helpful in processing a long conversation since there exists global coherence as we discussed in Chapter 1, this context information is only modeled in post-processing models,

such as SLU, dialog module. This limitation motivates us for the overall my thesis topic, and proposing a novel conversational ASR that models entire conversations.

2.2 Previous Approaches: End-to-End ASR

In this section, we review an End-to-End speech recognition approaches that motivate us to optimize the ASR model with the context information jointly. we also review the strengths and weaknesses of two main End-to-End approaches that motivate us for proposing joint CTC/Seq2Seq models 3 which provide better performance and fast convergence.

End-to-end speech recognition is a recently proposed approach that directly transcribes speech to text without requiring predefined alignment between acoustic frames and characters [8, 9, 10, 12, 13, 35, 36, 37]. As we described in the previous section, the traditional hybrid approach, Deep Neural Networks - Hidden Markov Models (DNN-HMM), factorizes the system into several components trained separately (i.e. acoustic model, context-dependent phone model, pronunciation dictionary, and language model) based on conditional independence assumptions (including Markov assumptions) and approximations [21, 38]. Unlike such hybrid approaches, the End-to-End model learns acoustic frames to grapheme mappings in one step towards the final objective of interest, and attempts to rectify the sub-optimal issues that arise from the disjoint training procedure.

Figure 2.3: The End-to-End ASR directly transcribes from a sequence of acoustic features with a single neural network based on the data-driven methods.

Recent work in End-to-End speech recognition can be categorized into two main approaches:

Connectionist Temporal Classificaiton (CTC) [7, 8, 9, 10] and Attention-based Encoder-Decoder [12, 13, 35, 36]. Both methods address the problem of variable-length input and output sequences.

2.2.1 Connectionist Temporal Classification (CTC)

The key idea of CTC [7] is using intermediate label representation π, allowing repetitions of labels and occurrences of a special blank label, $-$, which represents emitting no label. CTC trains the model to maximize $P(y|x)$, the probability distribution over all possible label sequences $\Phi(y')$:

$$P(y|x) = \sum_{\pi \in \Phi(y')} P(\pi|x) \tag{2.6}$$

where y' is given modified label sequence y' (blank symbol inserted between labels) for allowing blanks in the output.

CTC is applied on top of Recurrent Neural Networks (RNNs), which is interpreted as the label distribution including the blank. The probability of label sequence $P(\pi|x)$ is computed by product of the probability of each label based on the conditional independent assumption:

$$P(\pi|x) = \prod_{t=1}^{T} q_t(\pi_t) \tag{2.7}$$

where $q_t(\pi_t)$ denotes the activation of π_t-th unit in RNN output layer q at time t, representing the probability of t-th label of π.

By forward-backward algorithm, CTC can be computed efficiently as in Equation 5.2. $\alpha_t(u)$ is forward variable representing total probability of all possible prefixes($y'_{1:u}$) that end with u-th label, and $\beta_t(u)$ is backward variable ($y'_{u:U}$) vice versa. The network can then be trained with

standard backpropagation by taking the derivative with respect to $q_t(k)$ for any k label as in Equation 5.3. (Let $\text{lab}(y', k) = \{k : y'_u = k\}$)

$$P(y|x) = \sum_{u=1}^{|y'|} \frac{\alpha_t(u)\beta_t(u)}{q_t(y'_u)} \tag{2.8}$$

$$\frac{\partial}{\partial q_t(k)} P(y|x) = \frac{1}{q_t(k)^2} \sum_{u \in \text{lab}(y',k)} \alpha_t(u)\beta_t(u) \tag{2.9}$$

By relying on conditional independence assumption, the probability of prefix α and the probability of postfix β can be calculated in a dynamic programming method and CTC achieves fast convergence speed during the training. This characteristic is also well suited in speech recognition task where the alignment between input and output should be monotonic, rather than other tasks without such restriction, i.e. machine translation task.

However, such conditional independent assumption limited CTC cannot explicitly model the inter-label dependencies. This is not true in real world since the output labels, words, or characters are close related each other. This property limits the CTC to model linguistic information. Consequently, CTC requires to incorporate the lexicon or separately trained language models disjointly in order to alleviate this issue. This is similar to traditional hybrid framework [9, 10].

2.2.2 Attention-based Encoder-Decoder (Seq2Seq)

Unlike CTC approach, Attention model directly predicts each target without requiring intermediate representation or any assumptions, so that it has been shown improvement in CER when no external language model used over CTC [36]. The model emits each label distribution at u conditioning on previous labels as in Equation 2.10. The framework consists of two RNNs: *Encoder* and *AttentionDecoder*, so that it is able to learn two different lengths of sequence based on cross-entropy criterion. *Encoder* transforms x, to high-level representation $h = (h_1, \cdots, h_L)$ as in Equation 2.11, then *AttentionDecoder* produces probability distribution over characters, y_u,

conditioned on h and all the characters seen previously $y_{1:u-1}$ as in Equation 2.12. Here a special start-of-sentence(sos)/end-of-sentence(eos) token is added to target set, so that the decoder completes to generate the hypothesis when (eos) is emitted.

$$P(y|x) = \prod_{u} P(y_u|x, y_{1:u-1}) \tag{2.10}$$

$$h = \text{Encoder}(x) \tag{2.11}$$

$$y_u \sim \text{AttentionDecoder}(h, y_{1:u-1}) \tag{2.12}$$

Attention mechanism helps decoding procedure by integrating all the inputs h into c_u based on attention weight vector $a_u \in \mathbb{R}_+^L$ over input L identifying where to focus at output step u. As following equations 2.13 - 2.16, a_u can be computed by softmax of energy $e_{u,l}$ from two types of attention mechanisms: content-based and location-based [12]. Both are depending on (1) the decoding history, s_{u-1}, and (2) the content in input, h. Location-based attention mechanism additionally uses convolutional features f from (3) the previous attention vector a_{u-1}.

$$e_{u,l} = \begin{cases} \text{content-based:} \\ \qquad w^T \tanh(W s_{u-1} + V h + b) \\ \text{location-based:} \\ \qquad f_u = F * \alpha_{u-1} \\ \qquad w^T \tanh(W s_{u-1} + V h + U f_{u,l} + b) \end{cases} \tag{2.13}$$

$$a_{u,l} = \frac{\exp(\gamma e_{u,l})}{\sum_l \exp(\gamma e_{u,l})} \tag{2.14}$$

$$c_u = \sum_l a_{u,l} h_l \tag{2.15}$$

$$s_u = \text{Reccurency}(s_{u-1}, c_u, y_{u-1}) \tag{2.16}$$

where w, W, V, F, U, b are trainable parameters, γ is sharpening factor [12], and * denotes convolution.

In practice, the approach has two main issues. (1) The model is weak in noisy corpus. Attention model easily affected by additive noise and generate misalignments because the model does not have any constraint that the alignment is monotonic. (2) Another issue is that it is hard to train from scratch on larger input sequences via purely data-driven method. To make training faster, the author [12, 36] constrains the attention mechanism to only consider inputs within narrow range. However, this modification may limit the model's capability to extract useful information from long character sequence.

2.3 Previous Approaches: Context Modeling

2.3.1 Context Modeling in Language Models

In the language model trained only on text data, several recent studies have attempted to use document-level or dialog-level context information to improve language model performance. Recurrent neural network (RNN) based language models [1] have shown success in outperforming conventional n-gram based models due to their ability to capture long-term information.

Based on the success of RNN based language models, recent research has developed a variety of ways to incorporate document-level or dialog-level context information [2, 3, 4, 5]. Mikolov et al. proposed a context-dependent RNN language model [2] using a context vector that is produced by applying latent Dirichelt allocation [39] on the preceding text. Wang et al. pro-

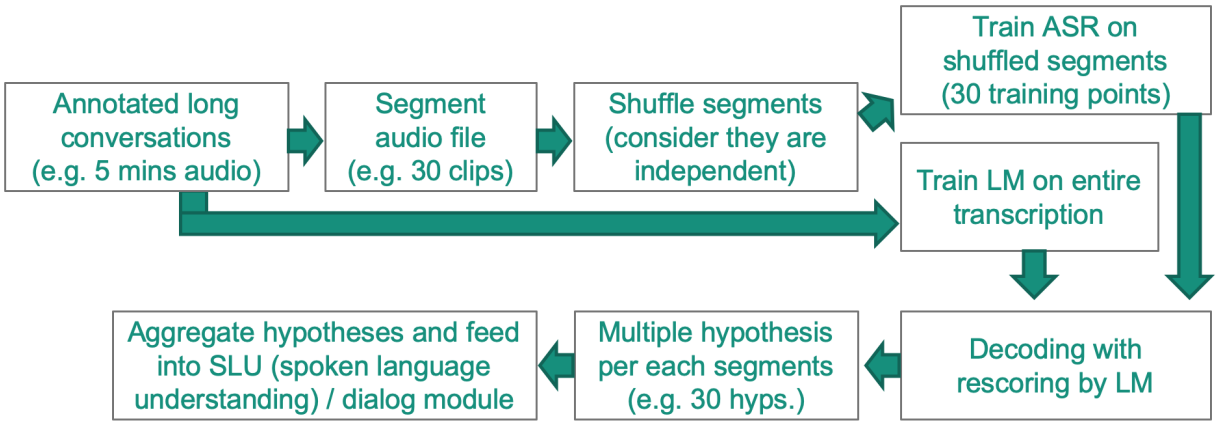


Figure 2.4: Context modeling in language models in conversational ASR pipelines.

posed using a "bag-of-words" to represent the context vector [3], and Ji et al. proposed using the last RNN hidden states from the previous sentence to represent the context vector [4]. Liu et al. proposed using an external RNN to model dialog context between speakers [5]. All of these models have been developed and optimized on text data, and therefore must still be combined with conventional acoustic models, which are optimized separately without any context information beyond sentence-level.

The recent study [40] attempted to integrated such a dialog session-aware language model with acoustic models (as in Figure 2.4). Their proposed session-aware language model explicitly learn dialog-level context information. By re-scoring with session-aware LM, they showed improved WER on SWBD tasks. However, this approach is still disjoint training procedure, so that the context information is only added as a post-fix, and is not fully exploited.

2.3.2 User-specific Context Modeling in End-to-End ASR

Several recent studies have considered incorporating the context information within a sequence-to-sequence based End-to-End speech recognition models [41, 42]. Pundak, et al. proposed Contextual LListen, Attend, and Spell (CLAS) that can be jointly optimized with the context information. They create a list of bias phrases, such as "`talk to`", "`play rihanna music`",

"`call demetri mobile`", "`talk to trivia game`", and use attention mechanism to encode the context. They showed significant WER improvements over the disjoint procedure.

In contrast with my thesis work which focuses on incorporating conversational context information for processing a long conversation, their methods rely on a strong assumption that the prior knowledge to get the list of bias phrases for specific tasks, *contact names, song names, voice search, dictation*, is exist at inference time.

2.4 Previous Approaches: Gating Mechanism

Gating mechanism is widely used to control information flow or integrate different types of information [30, 43].

The `input` gate, `output` gate, and `forget` gate in LSTM model are popular examples of usage of gating mechanism. The gate mechanism generates a value between 0 to 1 by using `sigmoid` function. This generated gate value is multiplied with network hidden representation and control the information flow. Consequently, it alleviates the gradient exploding or vanishing problem in the vanilla RNN models, and it enables to learn long dependencies. Learning Hidden Unit Contribution (LHUC) [44] is another example that is using a gating mechanism to amplitude the hidden units. Swietojanski, et al. proposed LHUC to learn the speaker-specific hidden unit contributions and has been shown the performance improvement in speaker adaptation tasks.

Gating-based approaches have been also used for fusing various types of representations [45, 46, 47]. It has been shown better performance than merely concatenating representation because it can decide how to weigh among the different representations and its increased representation power using multiplicative interactions.

In genre classification task and image search task, [45, 46] attempted to integrate word representation and visual representation by using gating mechanism on over the two different types of representation. [47] proposed to use language-specific gating mechanism for language universal speech recognition models, which enable to recognize multiple languages. The internal repre-

sentation of neural network architecture can be modulated with a gating mechanism conditioned on the language identity in a language-specific way.

Chapter 3

Multi-task learning of CTC/Seq2Seq

In this chapter, I first describe the complementary characteristics of two main End-to-End approaches: CTC and Seq2Seq models (in Section 2.2). I then propose a joint CTC/Seq2Seq End-to-End ASR framework to overcome the shortcomings of the existing two main approaches CTC and Seq2Seq and to improve convergence speed as well as accuracy performance [48, 49].

3.1 Complementary Characteristics of CTC and Seq2Seq

As we detailed discussed in Chapter 2, CTC based End-to-End ASR framework generates the output sequences which has the same length of input frame length by allowing repetition and blank symbols (Figure in 3.1).

CTC based End-to-End ASR framework has been shown strength in fast convergence speed. By relying on conditional independence assumption similar to Hidden Markov Models, the probability of prefix α and the probability of postfix β can be calculated in a dynamic programming method and CTC achieves fast convergence speed during the training. This characteristic is also Ill suited in speech recognition task where the alignment between input and output should be monotonic, rather than other tasks without such restriction, i.e. machine translation task.

However, such conditional independent assumption limited CTC cannot explicitly model the

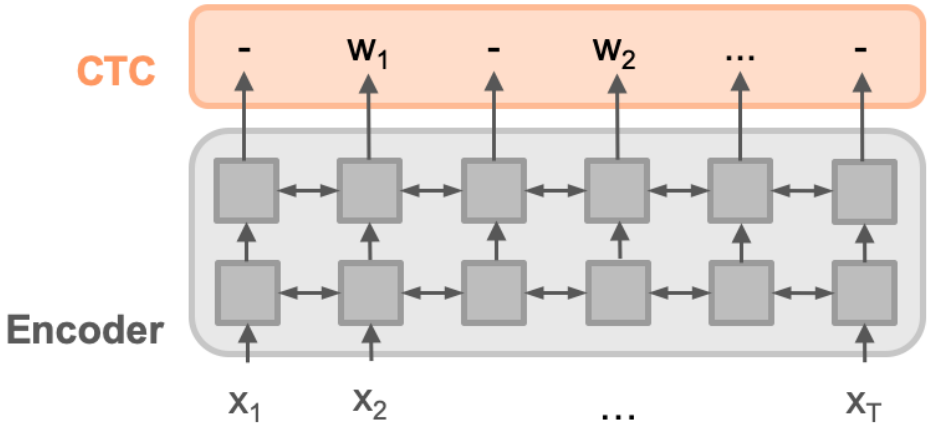


Figure 3.1: Previous CTC based End-to-End framework

inter-label dependencies. This is not true in real-world since the output labels, words, or characters are close related to each other. This property limits the CTC to model linguistic information. Consequently, CTC requires to incorporate the lexicon or separately trained language models disjointly in order to alleviate this issue. This is similar to a traditional hybrid framework [9, 10].

On the other hand, Seq2Seq based End-to-End ASR framework generates the output sequence conditioning on an attended h which is a weighted sum of h (high-level input feature representation) by an attention mechanism (Figure in 3.2.

As we detailed discussed in Chapter 2 as Ill, the biggest strength of SeqSeq based End-to-End ASR framework is the ability to learn label dependencies similar to the language model within a single network and optimized jointly without any conditional independent assumption.

However, Seq2Seq based End-to-End ASR framework has two main issues. (1) The model is weak in noisy speech data. The attention model is easily affected by noises, and generates misalignments because the model does not have any constraint that guides the alignments to be monotonic as in DNN-HMM and CTC. (2) Another issue is that it is hard to learn from scratch on larger input sequences via purely data-driven methods. To make training faster, the authors [11, 12] constrains the attention mechanism to only consider inputs within a narrow range. However, this modification may limit the model's capability to extract useful information

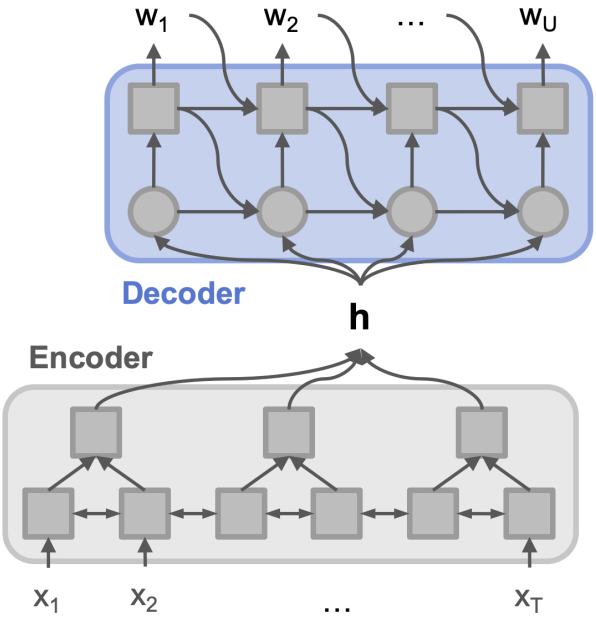


Figure 3.2: Previous Sequence-to-Sequence based End-to-End framework

from long character sequences.

3.2 Joint learning of CTC/Seq2Seq

The main idea of my joint CTC/Seq2Seq ASR model is to use both CTC and Seq2Seq objective function within a multi-task learning framework (MTL). I share the encoder network with both objective functions simultaneously. The shared encoder transforms our input sequence $x_{1:T}$ into high level features $\boldsymbol{h}$, the location-based attention decoder generates the output target sequence $w_{1:U}$.

Figure 3.3 illustrates the overall architecture of my joint CTC/Seq2Seq ASR framework. My proposed joint CTC/Seq2Seq based End-to-End framework: the shared encoder is trained by both CTC and attention model objectives simultaneously.

Unlike the Seq2Seq model, the forward-backward algorithm of CTC can enforce monotonic alignment between speech and label sequences. We therefore expect that my framework is more robust in acquiring appropriate alignments in noisy conditions.

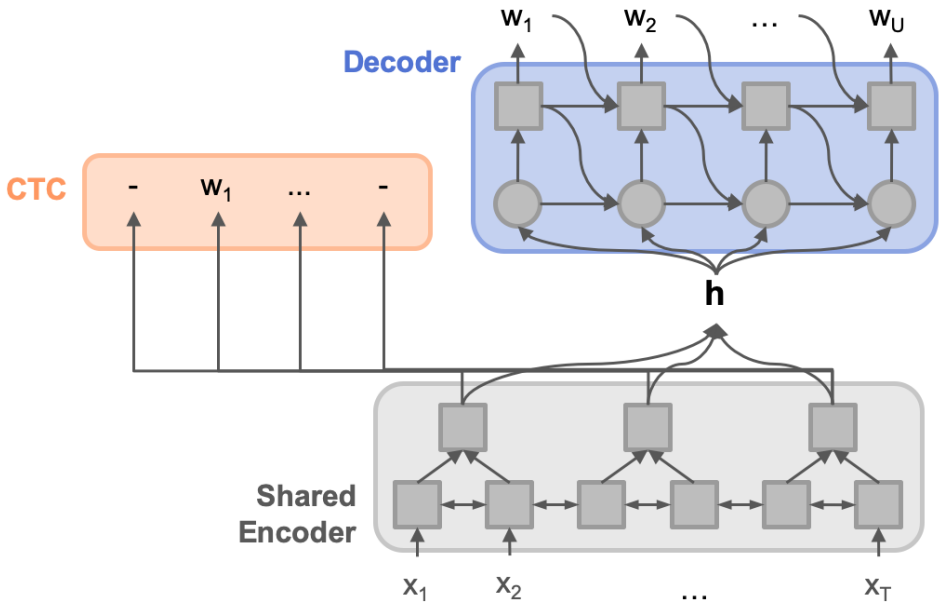


Figure 3.3: My proposed Joint CTC/Seq2Seq based End-to-End framework

Another advantage of using CTC as an auxiliary task is that the network is learned quickly. In my experiments, rather than solely depending on data-driven attention methods to estimate the desired alignments in long sequences, the forward-backward algorithm in CTC helps to speed up the process of estimating the desired alignment without the aid of rough estimates of the alignment which requires manual effort.

Let we have T-length acoustic input feature sequences $\boldsymbol{x}$, and corresponding U-length output label (word or characters) sequences $\boldsymbol{w}$.

The CTC loss to be minimized is defined as the negative log likelihood of the ground truth output sequence $\boldsymbol{w}^*$, i.e.

$$\mathcal{L}_{\text{CTC}} \triangleq -\ln \sum_{\boldsymbol{\pi} \in \Phi(\boldsymbol{w})} p(\boldsymbol{\pi}|\boldsymbol{x}) \tag{3.1}$$

where $\boldsymbol{\pi}$ is the label sequence allowing the presence of the *blank* symbol, Φ is the set of all possible $\boldsymbol{\pi}$ given u-length $\boldsymbol{w}$.

The Seq2Seq loss to be minimized is defined as the negative log likelihood of the ground

truth output sequence $\boldsymbol{w}^*$, i.e.

$$\mathcal{L}_{\text{S2S}} \triangleq -\ln P(\boldsymbol{w}^*|\boldsymbol{x}) = -\sum_u \ln P(w_u^*|\boldsymbol{x}, w_{1:u-1}^*) \tag{3.2}$$

where $w_{1:u-1}^*$ is all the previous labels.

The proposed joint CTC/Seq2Seq objective is minimizing the loss $\mathbf{L_{joint}}$ which is the combination of the CTC loss $\mathbf{L_{CTC}}$ and the Seq2Seq loss $\mathbf{L_{S2S}}$. With a tunable parameter, $\lambda : 0 \leq \lambda \leq 1$, the model regularizes the weight of the loss from CTC objective and cross entropy of Seq2Seq model:

$$\mathcal{L}_{|\langle\rangle\backslash\sqcup} = \lambda\mathcal{L}_{\text{CTC}} + (1-\lambda)\mathcal{L}_{\text{S2S}} \tag{3.3}$$

Both CTC and the attention-based encoder-decoder networks are also used in the inference step. The final hypothesis is a sequence that maximizes a weighted conditional probability of CTC and attention-based encoder-decoder network [15]:

$$\begin{aligned} \boldsymbol{w}* = \text{argmax}\{&\gamma \log p_{CTC}(\boldsymbol{w}|\boldsymbol{x}) \\ &+ (1-\gamma)\log p_{S2S}(\boldsymbol{w}|\boldsymbol{x})\} \end{aligned} \tag{3.4}$$

3.3 Experiments

3.3.1 Datasets

WSJ and CHiME-4

I performed three sets of the experiment: clean speech corpus, WSJ0 (15 hours), WSJ1 (81 hours), [50, 51], and noisy speech corpus, CHiME-4 (18 hours) [52].

The CHiME-4 corpus was recorded using a tablet device in an everyday environment - a

cafe, a street junction, public transport, and a pedestrian area. As input features, I used 40 mel-scale filterbank coefficients, with their first and second order temporal derivatives to obtain a total of 120 feature values per frame. Evaluation was done on (1) "eval92" for WSJ, and (2) "et05_real_isolated_1ch_track" for CHiME-4. Hyperparameter selection was performed on the (1) "dev93" for WSJ, and (2) "dt05_multi_isolated_1ch_track" for CHiME-4. All of my experiments I do not use any language model and lexicon information. For Seq2Seq model, I use only 32 distinct labels: 26 characters, apostrophe, period, dash, space, noise, and sos/eos tokens. CTC model uses the blank instead of sos/eos, and MTL model uses both sos/eos and the blank.

3.3.2 Training and Decoding

My model used 4 layers of 320 Bidirectional Long Short-Term Memory Networks(BLSTM) [30, 31] in the encoder, and 1 layer of 320 LSTM in the decoder. The top two layers read every second of hidden states of the below network, thereby the encoder reduced the utterance length by the factor of 4, $L = T/4$. 10 centered convolution filter of width 100 was used in the location-based attention to extract the feature from the previous step alignment. I use the sharpening factor $\gamma = 2$. Linear projection layer is followed by each BLSTM layer.

The AdaDelta algorithm [53] with gradient clipping [54] was used for optimization. All the weights are initialized with the range [-0.1, 0.1] of uniform distribution. For my MTL, I tested three different task weights, λ: 0.2, 0.5, and 0.8.

For decoding, I used a beam search algorithm similar to [55] with the beam size 20 to minimize the cost. I adjusted the cost by adding length penalty, length(hyp) $* 0.3$ for CHiME-4 and length(hyp) $* 0.1$ for WSJ experiments. Note that I do not use any lexicon or language model. My framework for these experiments is implemented with Chainer library [56].

3.3.3 Results

In Table 3.1 shows the Word Error Rate (WER) results of CTC only, Seq2Seq only, and my proposed joint CTC/Seq2Seq End-to-End model on the clean dataset WSJ0 and WSJ1.

Table 3.1: Comparison of Character Error Rate (% CER): CTC, Seq2Seq, and my Joint CTC/Seq2Seq on WSJ1, and WSJ0 tasks.

Model(train)	CER(valid)	CER(eval)
WSJ-train_si284 (80hrs)	dev93	eval92
CTC	10.06	12.45
Seq2Seq(content-based)	10.85	8.27
Seq2Seq(location-based)	10.88	8.07
MTL($\lambda = 0.2$)	10.86	8.16
MTL($\lambda = 0.5$)	**9.93**	7.43
MTL($\lambda = 0.8$)	10.46	**7.26**
WSJ-train_si84 (15hrs)	dev93	eval92
CTC	31.23	24.69
Seq2Seq(content-based)	31.00	22.10
Seq2Seq(location-based)	24.15	16.32
MTL($\lambda = 0.2$)	23.73	15.69
MTL($\lambda = 0.5$)	**22.45**	**15.12**
MTL($\lambda = 0.8$)	25.77	15.92

The WER results showed that my joint CTC/Seq2Seq End-to-End model significantly outperformed over both CTC or Seq2Seq only models. My joint CTC/Seq2Seq End-to-End model shows 7.0 - 9.5% and 6.6 - 9.9% relative improvements on validation and evaluation set, respectively.

In Table 3.1 shows the Word Error Rate (WER) results of CTC only, Seq2Seq only, and my proposed joint CTC/Seq2Seq End-to-End model on the noisy dataset CHiME-4. My joint CTC/Seq2Seq End-to-End model again significantly outperformed over both CTC or Seq2Seq

Table 3.2: Comparison of Character Error Rate (% CER): CTC, Seq2Seq, and my Joint CTC/Seq2Seq on noisy CHiME-4 tasks.

Model(train)	CER(valid)	CER(eval)
CHiME-4-tr05_multi (18hrs)	dt05_real	et05_real
CTC	37.16	48.84
Seq2Seq(content-based)	45.15	55.80
Seq2Seq(location-based)	36.16	48.31
MTL($\lambda = 0.2$)	35.71	47.95
MTL($\lambda = 0.5$)	33.56	46.85
MTL($\lambda = 0.8$)	**32.71**	**45.13**

only models.

We also observed that the benefit from my joint CTC/Seq2Seq increased in noisy conditions, and when larger weights on CTC loss (i.e. $\lambda = 0.8$) achieved the best performance in CHiME-4, while $\lambda = 0.5$ showed the best performance in clean WSJ0.

One noticeable thing is that my framework outperformed over both CTC and Seq2Seq models even in the clean corpus. One possible reason is that CTC can train the encoder maintaining a balance between acoustics and transcription information because CTC does not explicitly use character inter-dependencies.

Apart from the robustness, my proposed joint CTC/Seq2Seq End-to-End model can be also very helpful for accelerating learning the desired alignment.

Figure 3.4 shows the learning curve of different models, CTC only, Seq2Seq only, and my proposed joint CTC/Seq2Seq End-to-End model with (λ = 0.2, 0.5, 0.8) over training epoch. The character accuracy on the validation set of CHiME-4 is calculated by edit distance. Note that the accuracy of Seq2Seq and my proposed joint CTC/Seq2Seq End-to-End model were obtained with a given gold standard history. As shown in Figure 3.4, the CTC only model

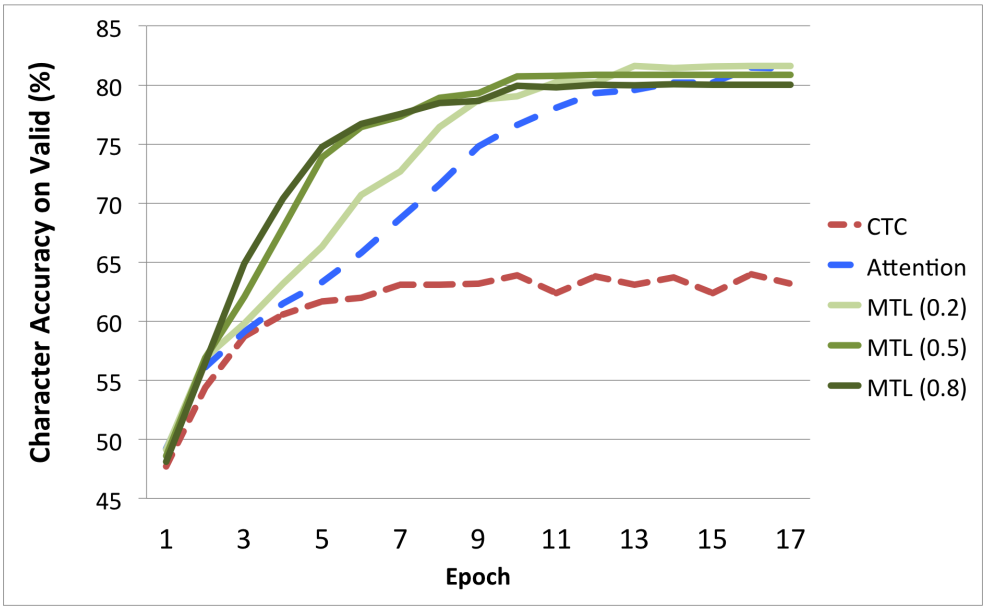


Figure 3.4: Comparison of the learning curve: CTC, Seq2Seq, and my Joint CTC/Seq2Seq.

converged the fast however its character accuracy is worse than the other models, Seq2Seq only, and my proposed joint CTC/Seq2Seq End-to-End model as I expected. I observed that Seq2Seq only model converged the slowest even though the character accuracy is similar to my proposed joint CTC/Seq2Seq End-to-End models. I also observed that when I use large λ giving more weight on CTC loss, the network learns fast and converges early. This result demonstrates the effectiveness of using CTC as an additional objective function to boost the convergence speed as I hypothesized.

In addition to comparing the learning curve between different End-to-End ASR models to demonstrate the effectiveness of using CTC as an additional objective function, I also visualize the learned attention alignments between input and output at different training epoch stages to analyze. From this analysis, I demonstrate why my proposed joint CTC/Seq2Seq End-to-End model outperformed over the Seq2Seq only models in addition to the benefit of convergence speed improvement. Since we are visualizing the actual attention weights, I only show Seq2Seq models and my proposed joint CTC/Seq2Seq End-to-End model.

Figure 3.5 shows the attention alignments between characters (y-axis) and acoustic frames

(x-axis) of Seq2Seq only model across over training epoch (1,3,5,7, and 9). All alignments are for the utterance (F05_440C0207_CAF_REAL) in noisy CHiME-4 evaluation set.

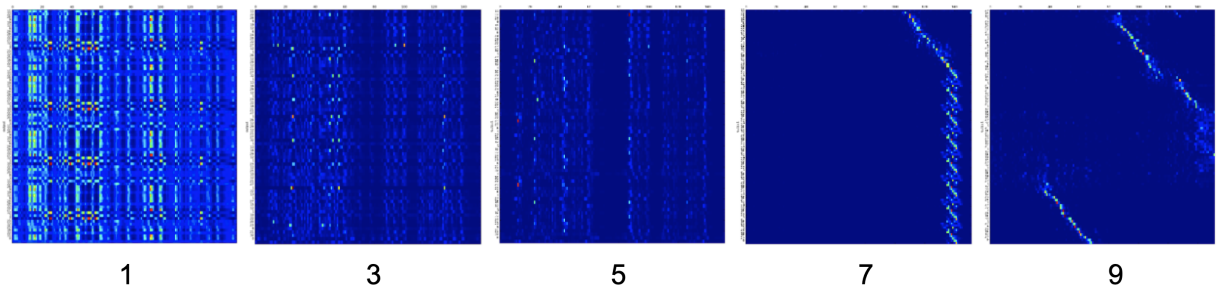


Figure 3.5: Visualization of the alignment between input frames and output characters of Seq2Seq only model.

We first observed that Seq2Seq only model could not learn the alignment properly and showed distorted alignment. I found that such distorted alignments resulted in a negative impact on the overall WER results.

Figure 3.6 shows the attention alignments between characters (y-axis) and acoustic frames (x-axis) of my proposed joint CTC/Seq2Seq End-to-End model across over training epoch (1,3,5,7, and 9). All alignments are for the same utterance (F05_440C0207_CAF_REAL) in noisy CHiME-4 evaluation set.

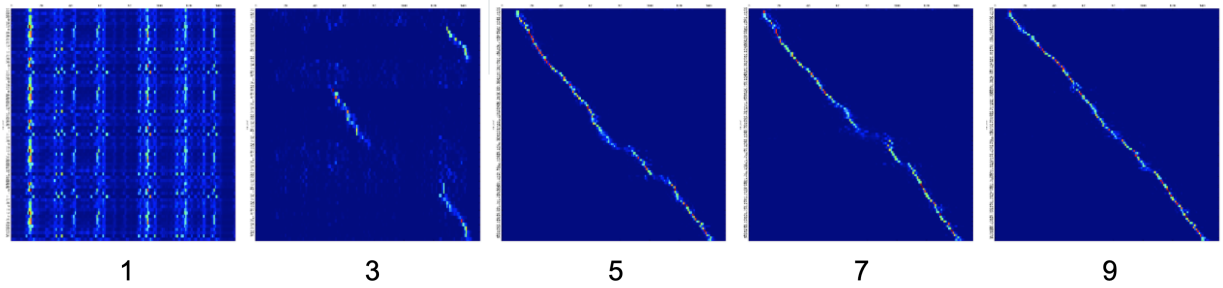


Figure 3.6: Visualization of the alignment between input frames and output characters of my proposed joint CTC/Seq2Seq End-to-End model.

We observed that my proposed joint CTC/Seq2Seq End-to-End model learned the desired, monotonic alignment in relatively early training stage (5th epoch), while Seq2Seq only model could not learn the desired alignment. From this result I found that using CTC as an additional objective function helps the model to learn our desired, monotonic alignment even without any manual techniques, such as windowing restrictions.

3.4 Summary

We have introduced a novel, general method for robust and fast End-to-End speech recognition based on multi-task learning approach with CTC and Seq2Seq models. My method improves robustness via training shared-encoder using auxiliary CTC objective. Moreover, it significantly speeds up learning the desired alignment without any manually restricting the range of inputs even in longer sequences. My method has outperformed both CTC and the Seq2Seq model on a speech recognition task in real-world noisy conditions as well as clean conditions. In the next chapters 4 6, I will show conversational End-to-End ASR based on the Joint CTC/Seq2Seq model.

Chapter 4

Conversational End-to-End ASR

The recent progress in End-to-End speech recognition systems that we discussed in Section 2promises to integrate all available information (e.g. acoustic, language resources) into a single model, which is then jointly optimized. It seems natural that conversational context information should thus also be integrated into the End-to-End models to improve recognition accuracy further.

Current End-to-End ASR solutions, even state-of-the-art systems, are still formulated as an optimization problem over isolated utterances, not entire conversations. These systems are therefore unable to use potentially important contextual information that spans across multiple utterances.

In this chapter, I present a novel conversational End-to-End ASR system that process entire conversations with a technique to preserve long conversational contexts, and present my context-aware End-to-End ASR models.

4.1 Preserving Conversational Context

In this section, I first demonstrate why we cannot treat an entire conversation as an single utterance to preserve conversational context information similar to the language model training. I

then show my technique to preserve such conversational context information efficiently by using `extract`, `detach`, `cache`, the context embeddings on serialized training minibatches.

4.1.1 Naive Solution: Concatenation of Utterances

The simplest solution would be treating an entire conversation as an single utterance to preserve conversational context. Similar to the previous work, context modeling in language models, that we reviewed in Chapter 2, we can simply concatenate multiple utterances and train the model on those data. However, the input sequences of speech frame and the output label sequences are too large to train the model so that it is computationally infeasible. For example, one second of speech input has more than 30 frames and 10 character labels, so 20 minute-long conversation may have more than 36k frames 12k characters which is huge, unable to fit in GPU memory. Even worse, it will result in poor parallelization due to severely variable-length between different dialogs. Therefore, we need to **`extract`** some sort of embeddings as "context" from such huge data. The detailed methods how we extract the context embeddings will be discussed in Chapter 6.

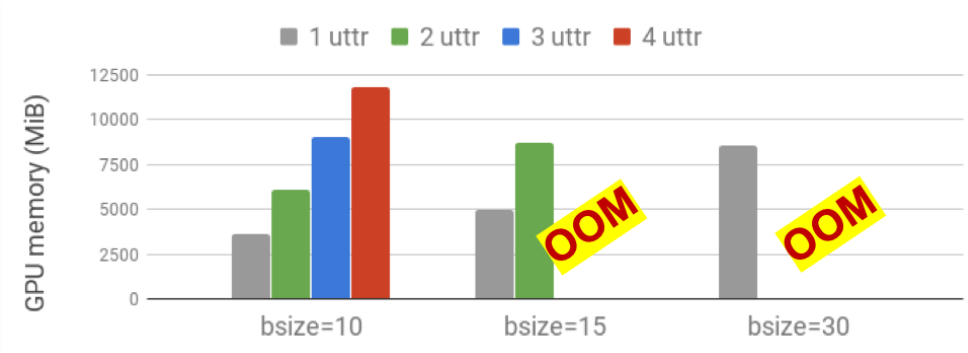


Figure 4.1: BPTT on entire conversation is computationally infeasible.

We conducted a simple experiment to show why back-propagating through time (BPTT) on

entire conversation is computationally infeasible. Figure 4.1 shows the consumption of GPU memory for computing the minibatch that has different number of utterances. Grey bar shows the normal way how we train the model with the minibatch that has a single utterance, and 30 size of minibatch can work well on 11G single GPU. However, when we are concatenating just 3 or 4 utterances, we got out-of-memory issue. Therefore, we need to **`detach`** the computational graph for the context embeddings and **`cache`** the context embeddings until needed. This can be seen a similar process to truncated back-propagation through time (BPTT).

4.1.2 Extract/Detach/Cache Context Embedding on Serialized Dataset

Dataset Serialization

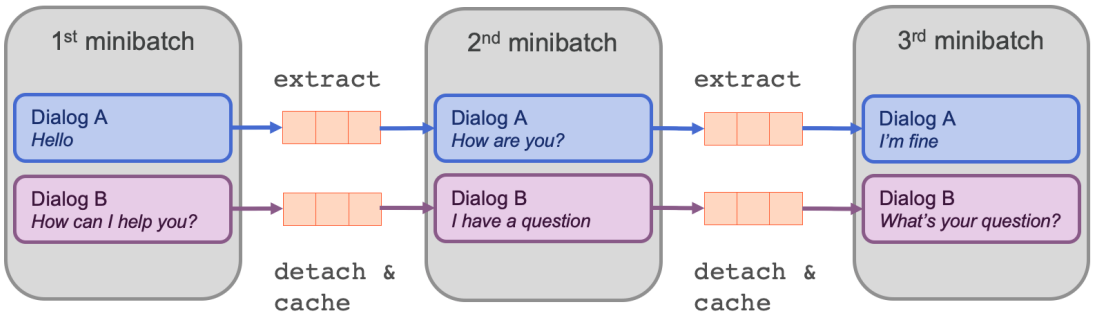


Figure 4.2: Extract/Detach/Cache context embedding on serialized utterances based on their onset time in dialog.

In order to keep track the context embeddings over each minibatch iteration during the training procedure, I create minibatches with serialized utterances based on their start time in dialog. I only apply randomization at dialog level. Unlike a minnibatch for typical ASR contains randomly shuffled utterances, our minibatch contains the utterances from each different dialogs, and the next minibatch contains the next utterances of the same dialogs as the previous minnibatch, and so on.

Figure 4.2 shows a simple example how I create minibatches to keep track the context em-

beddings. For example, let we have two dialogs A and B, and we want to create minibatch size of 2. We then choose the first utterance from dialog A, "*Hello*", and also the first utterance from dialog B, "*How can I help you?*" based on their start time within the dialog to create the first minibatch. For the next minibatch (2nd), we choose the second utterance from dialog A, "*How are you?*" and the second utterance from dialog B "*I have a question*" and put in the second minibatch. In this way, we can keep the order of the original conversation and track the context embeddings even during the training time.

This serialization may hurt the efficient parallelization since each minibatch may have the utterances with severely variable input and output lengths. This is because we should not control the order of utterance to track the context information, unlike in practical we try to use similar or same length of utterances to make a minibatch for reducing the dummy computation of the gap across the utterances. This issue can be alleviated by using some segmentation algorithm that tries to segment the long audio with the similar length of chunks.

We assume that we already have the segmentation information and audio files are already segmented, we may be able to extend to process the entire audio in similar way even without segmentation information by using voice activity detection algorithm [57].

Extract/Detach/Cache Context Embeddings

As I shown in previous section, the backpropagation through time (BPTT) on the entire conversation is computationally infeasible. As my solution, I propose to `extract` the context information in a single, fixed-dimensional vector. I call it as context embeddings in this thesis. Conversational context embedding can be encoded in two categories: acoustic conversational context and linguistic conversational context. In this thesis I focus on linguistic conversational context by using previous spoken utterances. I propose various way to encode linguistic conversational context in Chapter 6 and will discuss it further later.

Another important key technique is `detach` the computational graph for the context. At

every minibatch computation, after I `extract` the context embedding, we need to `detach` the computational graph. We can consider this context embedding for the current minibatch as an additional new input values which contains context information for the next minibatch computation. This is similar process to truncated backpropagation through time.

We then `cache` the context embeddings and pass them to the next minibatch computation. We have a single context embedding per each dialog per minibatch (utterance). The context embedding for each dialog should be passed to the next utterance of the same dialog, in case of the order of utterances are changed within minibatches the utterances as seen in Figure 4.3, I additionally store the dialog identities of the minibatch so that we can pass the correct context embedding to the next minibatch properly.

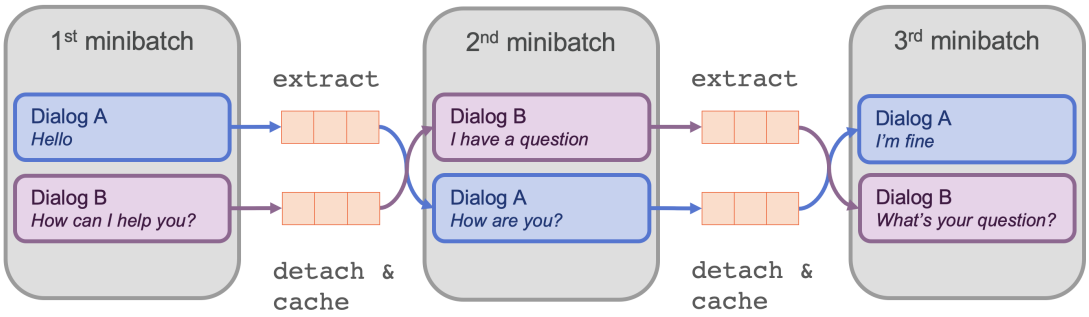


Figure 4.3: In case of the order of utterances are changed within minibatch, we additionally keep track the dialog ID.

Efficient Minibatch Creation

After the minibatch computations of a dialog finished, I release the cached context histories and reset the model states. Since the number of utterance (segmentation) is vary among the different dialogs, the reset time might be different per each dialog. For making training procedure simple, I try to group the dialog that has similar number of utterances and make a series of minibatches using the utterances of those dialogs. In this way, we wait until the series of utterances of the dialog groups are processed, then reset the model states, and release the cached context histories.

Algorithm 1 The overall procedure of creating serialized dataset given size of minibatch m and number of GPU k.

```
1: procedure CREATEDIALOGGROUPS([uttr], m, k)
2:     [sorted_uttr] = sort([uttr]) by utterance's start time and dialog id
3:     [dialogIDs] = sort(dialogIDs) by its number of utterances
4:     initialize [dialog-group]
5:     while [dialogIDs]:
6:         initialize [uttr-batch]
7:         for dialogID in dialogIDs :
8:             uttr = dequeue([sorted_uttr], dialogID)
9:             enqueue(uttr-batch, uttr)
10:        if dialogIDs not in [sorted_uttr]:
11:            enqueue(dialog-group, [uttr-batch])
12:            dialogIDs = dequeue([dialogsID], m × k)
13:    return [dialog-group]
```

For the training the model using multiple GPUs, we should consider data parallelism carefully in my approach. Assume there are k-number of GPUs on a machine and we want to create m-size of minibatch per each GPU. Given the model to be trained, each GPU will maintain a complete set of model parameters independently as well as minibatch data.

The usual way of data parallelism is that: for each iteration, 1) select $(m \times k)$ utterances randomly, 2) divide the utterances in the batch into k portions, and 3) distribute one to each GPU randomly.

However, in my approach, not for each iteration, but for a series of iteration (at dialog level), 1) I first select $(m \times k)$ dialogs that have same or similar number of utterances, 2) divide the dialogs into k portions, 3) distribute one to each GPU, and 4) for each iteration, select m utterances from the m dialogs in each GPU independently with our serialized way (as described in previously).

Overall procedure is described as follows:

Algorithm 2 The overall procedure of our minibatch creation including data parallelism for the Multi-GPU computation

```
1: procedure CREATEUTTRMINIBATCH([dialog-group], m, k)
2:     shuffle [dialog-groups]
3:     if multi-GPU then:
4:         for dgroup in [dialog-groups]:
5:             for dbatch in dgroup:
6:                 split dialogs into k
7:                 distribute one (m-dialogs) to each GPU
8:                 [uttr-batch] += unpack(dbatch)
9:     else:
10:        for dbatch in [dialog-groups]:
11:            [uttr-batch] += unpack(dbatch)
12:
13:    end if
14:    return [uttr-batch]
```

4.2 Vanilla Context-Aware End-to-End ASR

In this section, I describe my proposed vanilla conversational End-to-End ASR that incorporates the context embeddings which is forwarded from previous minibatch (utterance) computation based on serialized dataset (in Section 4.1). As we discussed in previous Chapter 3, the recent progress in End-to-End speech recognition systems promises to integrate all available information (e.g. acoustic, language resources) into a single model, which is then jointly optimized. It seems natural that such conversational context information should thus also be integrated into the End-to-End models to improve recognition accuracy further. I formulate my vanilla conversational End-to-End ASR as follows.

Let we have a dataset consists of D-number of dialogs, $\{d_1, d_2, \cdots, d_D\}$. Let each dialog d_d has N-number of segments which is the pair of acoustic features x and word or character (subword) sequences w, $d_d = ((x, w)^1, \cdots, (x, w)^N)$. We have variable input and output lengths for each segment n. Let x^n is T-length of sequence of acoustic features $x^n = (x_1, \cdots, x_T)$ and let w^n is U-length of sequence of words $w^n = (w_1, \cdots, w_U)$. Let we have a context embedding, c.

I use Seq2Seq framework (in Chapter 2 and 3) as a basis and extend the decoder sub-network.

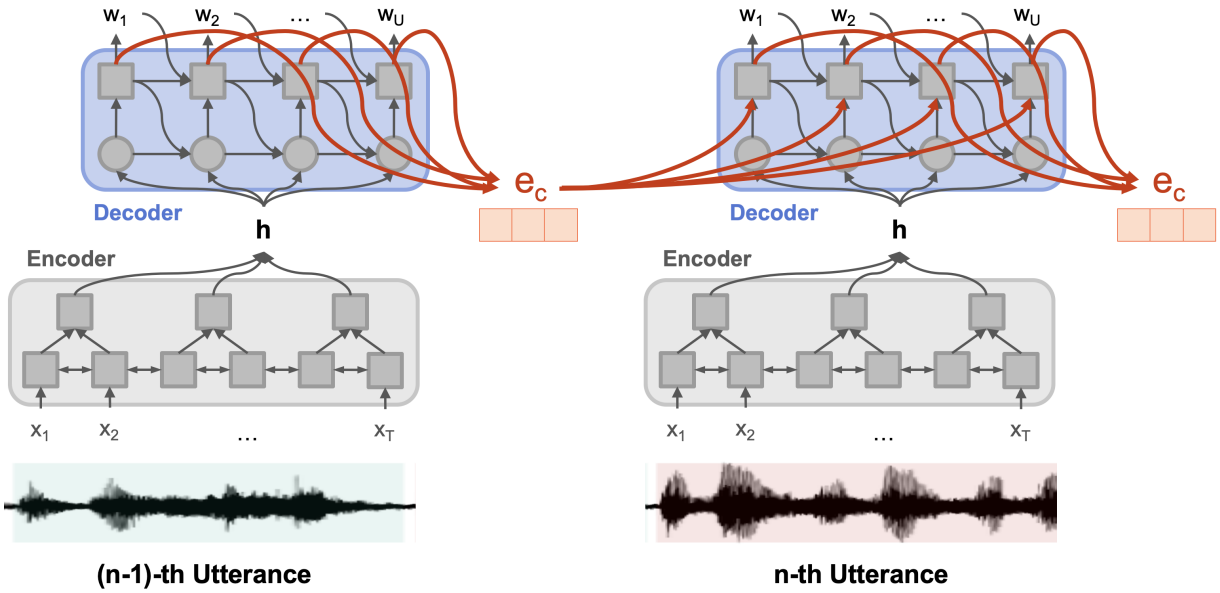


Figure 4.4: Overall architecture of my conversational End-to-End ASR models.

Figure 4.4 shows the standard End-to-End ASR that I use as basis. The standard End-to-End system is modeling only each utterance $(x, w)^n$. The T-length acoustic input $x_{1:T}$ is forwarded and mapped to $h_{1:\hat{T}}$, high-level input features in `Encoder`, and usually we subsample original length from T to $\hat{T}$ by a factor of 4. The `Decoder` network takes h and and generates w with attention mechanism. The whole network parameters are optimized towards maximizing probability of w^n given x^n:

$$\theta = \max_\theta \sum_U \log P(w_u^n | x_{1:T}^n, \hat{w}_{<u}^n; \theta) \tag{4.1}$$

We now have an additional input, c, context embedding which is generated from previous utterance. In my vanilla conversational End-to-End model, I simply use a single preceding utter-

ance, $(x, w)^{n-1}$, to obtain the context embedding c^n for n-th utterance modeling.

$$e_c^n = \texttt{ContextEncoder}((x, w)^{n-1}) \tag{4.2}$$

My proposed conversational end-to-ennd ASR is modeling each utterance conditioning on the context embedding c^n, and the whole network parameters are optimized towards maximizing probability of w^n given x^n and c^n:

$$\theta = \max_\theta \sum_U \log P(w_u^n | x_{1:T}^n, \hat{w}_{<u}^n, e_c^n; \theta) \tag{4.3}$$

4.3 Experiments

4.3.1 Dataset

300 hours of Switchboard task (SWBD)

I investigated the accuracy performance of my proposed conversational end-to-end ASR models on the Switchboard LDC corpus (97S62) which has a 300 hours training set. Note that in this experiment I did not use the Fisher dataset. I split the Switchboard data into two groups, then used 285 hours of data (192 thousand sentences) for model training and 5 hours of data (4 thousand sentences) for hyper-parameter tuning. Evaluation was carried out on the HUB5 Eval 2000LDC corpora (LDC2002S09, LDC2002T43), which have 3.8 hours of data (4.4 thousand sentences), and I show separate results for the Callhome English (CH) and Switchboard (SWB) evaluation sets. I denote train_nodup, train_dev, SWB, and CH as our training, development, and two evaluation datasets for CH and SWB, respectively. Table 4.1 shows the number of dialogs per each dataset.

I sampled all audio data at 16kHz, and extracted 80-dimensional log-mel filterbank coefficients with 3-dimensional pitch features, from 25 ms frames with a 10ms frame shift. I used

Table 4.1: Experimental dataset description: SWBD datasets.

	training	validation	evaluation	
	SWBD	SWBD	SWBD	CallHm
dialogs	2,402	34	20	20
utters./dialog	80	118	92	131

83-dimensional feature vectors to input to the network in total. I used 49 distinct labels: 26 characters, 10 digits, apostrophe, period, dash, underscore, slash, ampersand, noise, vocalized-noise, laughter, unknown, space, start-of-speech/end-of-speech, and blank tokens.

Note that no pronunciation lexicon was used in any of the experiments.

4.3.2 Training and decoding

Model

In this experiment, I used joint CTC/Seq2Seq End-to-End speech recognition architecture [48, 49] with ESPnet toolkit [14] as I described in previous Chapter 3. I used a CNN-BLSTM encoder as suggested in [15, 58]. I followed the same six-layer CNN architecture as the prior study, except I used one input channel instead of three, since I did not use delta or delta delta features. Input speech features were downsampled to (1/4 x 1/4) along with the time-frequency axis. Then, the 4-layer BLSTM with 320 cells was followed by the CNN. I used a location-based attention mechanism [12], where 10 centered convolution filters of width 100 were used to extract the convolutional features.

The decoder network of both my proposed models and the baseline models was a one-layer LSTM with 300 cells. My dialog-context aware models additionally requires one-layer with 300 hidden units for incorporating the context vector with decoder states, and attention network with 2402-dimensional output layer to generate the context vector. I also built a character-level RNNLM (Char-RNNLM) on the the same Switchboard text dataset. The Char-RNNLM network

was a two-layer LSTM with 650 cells, trained separately only on the training transcription. This network was used only for decoding. Note that I did not use any extra text data other than the training transcription.

The AdaDelta algorithm [53] with gradient clipping [59] was used for optimization. I used $\lambda = 0.5$ for joint CTC/Attention training. I bootstrap the training my proposed dialog-context aware End-to-End models from the baseline End-to-End models. For decoding of the models, I used joint decoder which combines the output label scores from the AttentionDecoder, CTC, and Char-RNNLM by using shallow fusion [15]:

$$\begin{aligned} \boldsymbol{y}* = \text{argmax}\{ & \log p_{att}(\boldsymbol{y}|\boldsymbol{x}) \\ & + \alpha \log p_{att}(\boldsymbol{y}|\boldsymbol{x}) \\ & + \beta \log p_{rnnlm}(\boldsymbol{y})\} \end{aligned} \tag{4.4}$$

The scaling factor of CTC, and RNNLM scores were $\alpha = 0.3$, and $\beta = 0.3$, respectively. I used a beam search algorithm similar to [55] with the beam size 20 to reduce the computation cost. I adjusted the score by adding a length penalty, since the model has a small bias for shorter utterances. The final score $s(\boldsymbol{y}|\boldsymbol{x})$ is normalized with a length penalty 0.1.

The models were implemented by using the Chainer deep learning library [60], and ESPnet toolkit [14, 48, 49].

4.3.3 Results

I evaluated both the End-to-End speech recognition model which was built on sentence level data (***sentence-level end2end***) and my proposed conversational End-to-End ASR models which leveraged conversational context information within and beyond the utterance level (***conversational-context aware end2end***).

Table 4.2 shows the WER of my baseline, my conversational End-to-End ASR models, and several other published results those were using character level output units and only trained on

Table 4.2: Word Error Rate (% WER) of baseline and my conversational End-to-End ASR models (vanilla) on the Switchboard dataset.

Models	CH WER (%)	SWB WER (%)
sentence-level end2end		
Seq2Seq A2C [61]	40.6	28.1
CTC A2C [9]	31.8	20.0
CTC A2C [62]	32.1	19.8
sentence-level end2end		
My baseline (CTC/Seq2Seq)	34.4	19.0
dialog-context aware end2end		
My method (a)	34.1	**18.2**
My method (b)	**33.2**	18.6

300 hours Switchboard training data.

As shown in Table 4.2, I obtained a performance gain over my baseline *sentence-level end2end* by using the conversational context information. My proposed method (a) performed best on SWB evaluation set showing 4.2% relative improvement over my baseline. My method (b) performed best on CH evaluation set showing 3.4% relative improvement over my baseline.

Table 4.3: Substitution rate (% Sub), Deletion rate (% Del), and Insertion rate (% Ins) for the baseline and my proposed model.

Model	Test	Sub %	Del %	Ins %	WER %
Baseline	CH	23.9	**5.8**	4.7	34.4
Proposed model(a)	CH	23.9	5.9	4.3	34.1
Proposed model(b)	CH	**22.8**	6.3	**4.1**	**33.2**
Baseline	SWB	13.1	3.4	2.5	19.0
Proposed model(a)	SWB	**12.5**	3.4	**2.2**	**18.2**
Proposed model(b)	SWB	12.6	3.6	2.4	18.6

I also analyze the WER results by decomposing it into the insertion, deletion, and substitution

rates. In Table 4.3 shows the insertion, deletion, and substitution rates. I observed that the largest factor of WER improvement was from the substitution rates rather than deletion or insertion.

In addition to the WER results, I show the three example utterances which are adjacent and manually chosen from evaluation dataset. In Table 4.4, second column is the preceding utterance, and the last column shows the current utterance. In the first example, the groundtruth word is the bolded word "sauna" which is relatively rare term than "saw". The baseline model cannot predict it, however, my conversational End-to-End ASR model can predict correctly. When we take a look at the preceding sentence, there exist semantically related words were appeared, and my proposed model might benefit from the context information this past utterance. In the second example, the groundtruth word is the bolded word "comfortable". The baseline model cannot predict it again, however, my conversational End-to-End ASR model can predict correctly. In the previous utterance exists semantically related words were appeared, and my proposed model might benefit from the context information this past utterance.

Table 4.4: Comparison of reference transcription, and two hypotheses of the baseline and my proposed conversational end-to-ennd ASR models.

Model	previous sentence	current sentence
REF	*yes it is so hot in the building have you ever been in it*	*it is like a* ***sauna***
base	*yeah if when he said that is like just so hot in ours belly have ever been but*	*it is like i* ***saw***
Ours	*yeah if when he is in this like it is so hot in the belief I have never been but*	*it is like a* ***sauna***
REF	*if we go we like check into a to a to a hotel but*	*i know but it is much more* ***comfortable***
base	*if we go we like check until the law that you know to do*	*i know that is much more* ***comes of one***
ours	*if we go we like check into a law if it does a job hotel*	*i know that is much more* ***comfortable***

4.4 Summary

In this chapter, I described my proposed efficient way to preserve a long, conversational context information over the entire conversations based on `extract/detach/cache` context embeddings on serialized minibatch sets by utterance start time in dialog. I also described my proposed conversational End-to-End ASR models that explicitly use conversational context embeddings

which is beyond utterance-level information and that can be optimized in an End-to-End manner towards minimizing the current utterance prediction error. My conversational End-to-End ASR model was shown to outperform previous End-to-End models trained on utterance-level data, even when I just use a single previous utterance. In the next chapters, we will discuss a more effective way to integrate the context embeddings within our conversational End-to-End ASR model in Chapter 5, and various ways to encode context embedding better in Chapter 6.

Chapter 5

Gated Contextual Decoder

My vanilla conversational context End-to-End models have been shown promising results as seen in Chapter 4 by jointly learning conversational context information based on my End-to-End ASR architecture (in Chapter 3). In this chapter, I explore the way to integrate conversational context embeddings e_c more effectively by using gating mechanism. In Section 5.1, I show the example of usage of gating mechanism that was motivated us to use in my conversational End-to-End ASR models. In Section 5.2, I will describe both the naive way - concatenation and gated contextual decoder to fuse context embedding in to the decoder network.

5.1 Integrating Different Types of Representation

As we already discussed in Chapter 2, gating mechanism is widely used to integrate different types of information [30, 43]. In this section, I show my language-specific gating mechanism [47] as an example of gating mechanism to integrate different types of representation. This previous work motivate us to propose gated contextual decoder.

The way how to use gating mechanism in [47] is as follows: The outputs of each hidden layer in ASR models are processed by a series of language-dependent gates before being passed to the next layer in the model. Specifically, one-hot language indicator vector d_l for each language l

is created. Then the gate value based on the language indicator vector d_l and the current output values of h_i, the ith hidden layer were computed, as

$$g(h_i, d_l) = \sigma(Uh_i + Vd_l + b) \tag{5.1}$$

where U, V, and b are trainable parameters. The gated hidden activations are then calculated as

$$\hat{h_i} = g(h_i, d_l) \odot h_i \tag{5.2}$$

Finally, $\hat{h_i}$ and d_l are concatenated and input to the next layer.

$$\tilde{h_i} = [\hat{h_i} : d_l] \tag{5.3}$$

If the dimensions of h_i and d_l are n and m, respectively, each gating layer requires $(n + m)$ x n additional parameters.

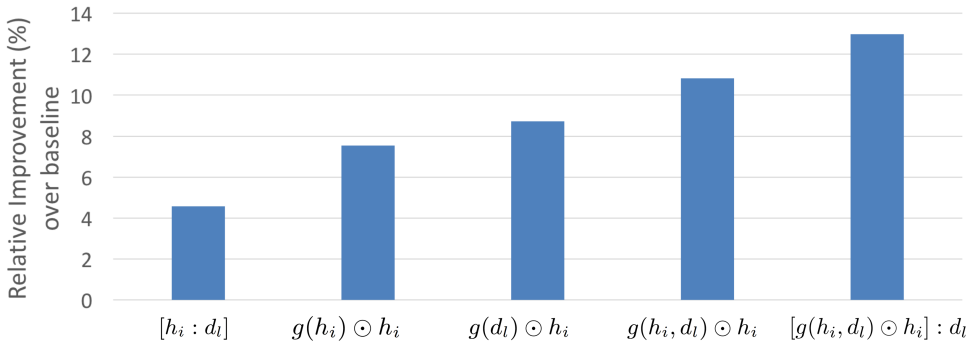


Figure 5.1: Comparison of the % WER performance improvement (relative) over different types of gating mechanism.

The Figure 5.1 shows the comparison of the WER performance improvement over different types of gating mechanism. The leftmost bar shows the result of simply concatenation with the one-hot language vector d_l. The next three bars show the performance of different gating functions driven by the current hidden state, the language identity, or both, respectively. Finally, the rightmost bar shows the approach shown in Equations 5.2-5.3. The Figure 5.1 shows that the

best performance was obtained when gating is applied to every layer (not shown).

The gating mechanism enable to effectively modulate the internal representation. This finding motivates us to use gating mechanism to fuse conversational context embeddings. I will describe the details in the next Section 5.2.

5.2 Conversational Context Fusion in Decoder

In this Section, I show how to integrate the conversational context embeddings which is forwarded from previous minibatch as discussed in Section 4.1 into my model and trained jointly in an End-to-End manner.

5.2.1 Naive Solution: Concatenation of Context/Word/Speech Embeddings

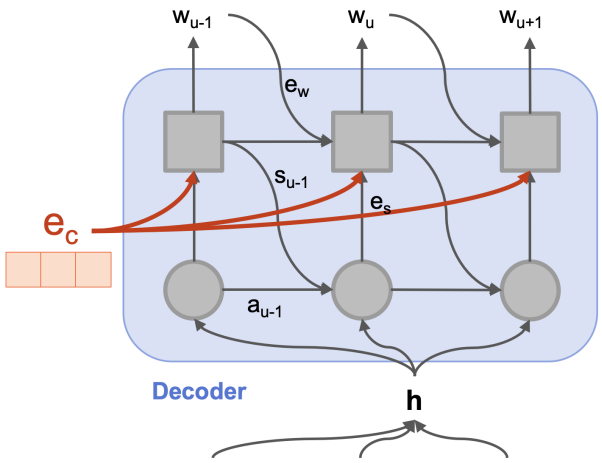


Figure 5.2: Decoder network of End-to-End ASR that takes conversational context embeddings.

If we look at the detail of decoder part, as in Figure 5.2, it has attention mechanism to identify which input is more focused on at each output prediction. This attention mechanism takes h, high-level input features, and previous attention weights a_{u-1}, and previous decoder states s_{u-1}. Then the attention mechanism generates attended h, which is generally called as context vector, but to avoid confusion with my conversational context embedding, I call it as speech embedding.

The LSTM decoder, takes two different types of embeddings, 1) speech embeddings e_s, the attended input, and 2) word embeddings e_w, from previous output token. I extend this decoder, since we now have "context embedding" e_c from previous spoken utterances (in Equation 5.4. So, my LSTM decoder takes this embedding as an additional input. we will discuss how we encode a single, fixed dimensional, context embeddings in Chapter 6. The network parameters are optimized towards maximizing probability of w given x and e_c (in Equation 5.5).

$$e_c = \texttt{ContextEncoder}((x, w)^{n-1}) \tag{5.4}$$

$$\theta = \max_\theta \sum_U \log P(w_u | x_{1:T}, \hat{w}_{<u}, e_c; \theta) \tag{5.5}$$

The simplest way to use the additional input, context embeddings, would be concatenation with other embeddings, speech embeddings, previous word embedding, as illustrated in Figure 5.3.

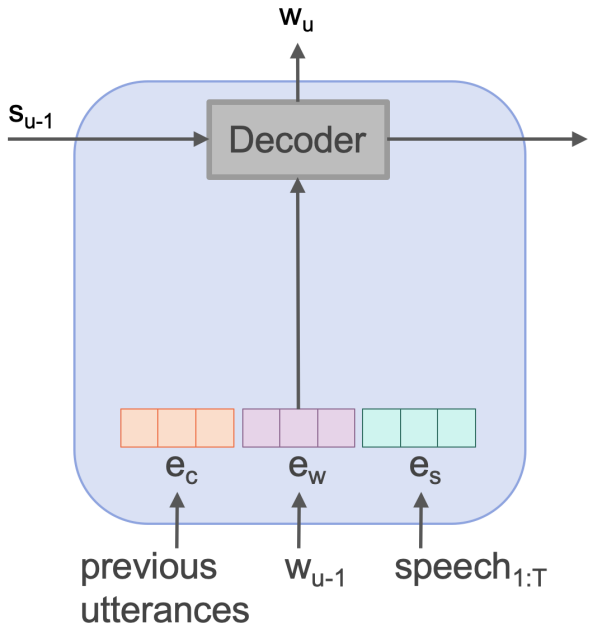


Figure 5.3: Concatenation of context/word/speech embeddings

5.2.2 Gated Contextual Decoder

Rather than simple concatenation methods as discussed in previous section, I propose to use gating mechanism to integrate three different types of embeddings: context, word, speech. The gating mechanism has been successfully used for fusing different types of representations, i.e. word and visual representation in genre classification task or image search task [45, 46], and for learning different langauges in speech recognition task [47]. The gating mechanism decides how to weigh different types of embeddings, and we can shape information flow using multiplicative interactions. I first concatenate three embeddings, then take sigmoid to get gating value between 0 to 1 (in Equation 5.6).

$$g = \sigma(e_c, e_w, e_s) \tag{5.6}$$

$$e = g \odot (e_c, e_w, e_s) \tag{5.7}$$

$$s_u = \texttt{LSTM-Decoder}(e, s_{u-1}) \tag{5.8}$$

This gating value g is then product with original embeddings, and generates the final gated embeddings (in Equation 5.7). This new embeddings e are forwarded into LSTM decoder (in Equation 5.8).

Figure 5.4 illustrates my proposed contextual gating mechanism.

5.3 Experiments

In this section, I evaluate my proposed conversational End-to-End ASR with gated contextual decoder on 300 hours of Switchboard (SWBD) task. The detailed dataset description is already discussed in 4.3.1.

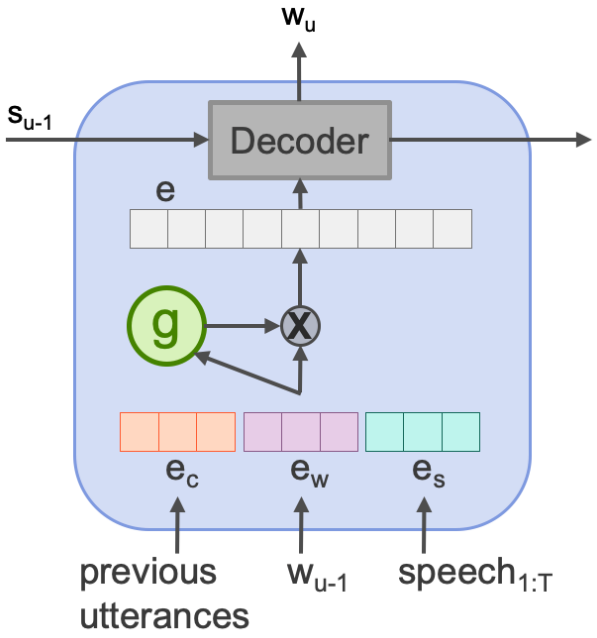


Figure 5.4: My contextual gating mechanism in decoder network of End-to-End ASR to integrate context/word/speech embeddings

5.3.1 Output Units

In this study, I use word+character (`WordChar`), rather than character output units.

Word+Character units

I first explore the use of word and character for the output units of my model. Direct acoustics-to-word (A2W) models train a single neural network to directly recognize words from speech without any sub-word units, pronunciation model, which significantly simplifies the training and decoding process [63, 64, 65, 66, 67]. In addition, A2W models can learn more semantically meaningful conversational context representations rather than a single character unit. Also it allows us to exploit the pre-trained external resources like word/sentence embeddings where the unit of representation is generally words. While these benefits, A2W models require more training data compared to conventional sub-word models and additional efforts are needed to handle out-of-vocabulary(OOV) words. In order to mitigate these issues, I first restrict the vocabulary to

10k frequently occurring words. I then additionally use character units and start-of-OOV (sunk), end-of-OOV (eunk) tokens to make my model generate a character by decomposing the OOV word into a character sequence. For example, the OOV word, *rainstorm*, is decomposed into (sunk) *r a i n s t o r m* (eunk) and the model tries to learn such a character sequence rather than generate the OOV token. The `WordChar` contains roughly 10k units (10,034) units including the words (10,000) and the characters (34). By using this output units, we were able to obtain better performed A2W baseline over standard models which use word only.

5.3.2 Architecture

The model architecture is also same as my vanilla End-to-End ASR models (described in Section 4.2), except that we have additional gating mechanism to incorporate the context embedding more effectively. This gating mechanism requires additional parameters to be trained. The amount of parameters are depend on the dimension of context embeddings, previous output token embeddigns, and speech embeddings, in this experiment requires additional 2 million trainable parameters.

For the architecture of the End-to-End speech recognition, I used joint CTC/Seq2Seq End-to-End ASR [48, 49] as I proposed in Chapter 3. As suggested in [15, 58], the input feature images are reduced to $(1/4 \times 1/4)$ images along with the time-frequency axis within the two max-pooling layers in CNN. Then, the 6-layer BLSTM with 320 cells is followed by the CNN layer. For the attention mechanism, I used a location-based method [12]. For the decoder network, I used a 2-layer LSTM with 300 cells.

The training procedure and decoding procedure is same as in the previous experiment setup 4.3.2.

5.3.3 Results

In Table 7.7 summarizes the Word Error Rate (% WER) results of Switchboard (300hr) with `WordChar` output units. The * mark denotes our estimate for the number of parameters used in the other systems.

Table 5.1: Comparison of word error rates (% WER) of baseline and my proposed conversational End-to-End ASR with gated contextual decoder on SWBD task.

Model		#params.	LM	SWB	CH
Other A2W End-to-End systems					
CTC [65]	Word output, phone pretrain.	n/a	✗	14.6	23.6
My End-to-End ASR systems					
My baseline	Char output	23M	✗	19.0	34.4
My baseline	`WordChar`	32M	✗	17.9	30.6
My Context models					
vanilla	`WordChar`	33M	✗	17.3	30.3
vanilla + gate	`WordChar`	35M	✗	17.2	29.8

I first present the other systems, my baseline system that use the word level output units, and then my conversational systems. Unlike other systems, note that my system does not use any phonetic information [65].

I first observed that my new baseline with my proposed output unit set `WordChar` performed better than the old baseline that using character level outputs. As shown in Table 5.1, the new baseline with `WordChar` obtained 5.7% and 11.0% relative WER improvements on the evaluation set, SWB and CH, respectively.

Secondly I observed that my vanilla conversational End-to-End ASR model obatained 3.3% and 1.0% further, relative WER improvement over my new improved baseline. This result again confirmed the finding that incorporating context embedding generated from previous utterance helps the current utterance prediction.

I also observed that using gated contextual decoder to incorporate the context embeddings

shows the improved WER results. As shown in this Table 5.1, my proposed conversational End-to-End ASR with gated contextual decoder obtained slightly further WER improvement on the evaluation set.

5.4 Summary

In this chapter, I described my proposed effective way to integrate the context embeddings and my proposed `WordChar` output sets. By using my proposed gated contextual decoder and my proposed `WordChar` output sets, I obtained further WER improvements over the previous vanilla conversational End-to-End ASR models that we discussed in the previous Chapter 4. In the next chapters, I will propose a more effective and better way to encode the context embeddings in Chapter 6.

Chapter 6

Conversational Context Encoder

We have discussed how we can preserve conversational context during training without GPU memory issue and integrate context embedding into the ASR models in an End-to-End manner. In this Section, we will discuss how to encode conversational context embedding from previous spoken utterances.

6.1 Context Encoder

As Figure in 6.1, I create an additional sub-network in my End-to-End ASR that generates conversational context embedding, e_c, from previous spoken utterances. The conversational context embedding, e_c, is a single, fixed-length vector that encodes conversational context information.

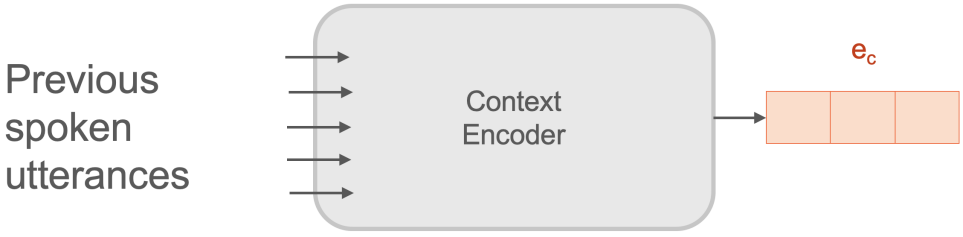


Figure 6.1: My context encoder generates conversational context embedding from previous spoken utterances

I propose various types of context encoder, based on four kinds of criteria:

- Selection of history of unit and its representation: we can select either word-level or sentence-level as an each input of context encoder.
- Augmentation with "world knowledge": we can augment with "world knowledge" by using external word/sentence embeddings which is trained on massive amount of text data.
- Aggregation of multiple history unit: context encoder may take multiple past utterances, not just a single preceding utterance. We can use various ways i.e. mean-pooling/LSTM/Attention in the context encoder to aggregate these histories.
- Sampling strategy: similar to sampling strategy used in previous word embedding, we can choose either model outputs or groundtruth for the histories to avoid overfitting.

In the following subsections, I will describe details of various types of context encoder.

6.1.1 Utterance History Unit and Representations

We can select either word-level or sentence-level as an each input of context encoder. Each history unit can be represented as one-hot vector simply, or output token level (word/subword/character) multinomial distributions from the model output or embedding representation by re-using the embedding layer in decoder network.

6.1.2 Aggregation of Multiple Utterance History

I also explore the way to encode context embeddings by using multliple utterance history, not just use a single utterance like vanilla model (in Chapter 4 and Chapter 5.

Since my context encoder may take multiple past utterances, not just a single preceding utterance, I consider various ways i.e. mean-pooling/LSTM/Attention to aggregate these histories. Figure 6.2 shows the simplest way to aggregate multiple word-level inputs by using mean-pooling.

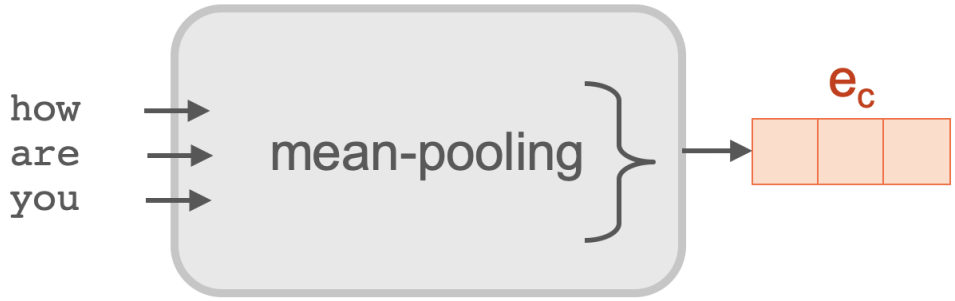


Figure 6.2: My vanilla context encoder with word-level input and mean-pooling method

I first investigate the effect of the number of utterance history being encoded. I tried different $N = [1, 5, 9]$ number of utterance histories to learn the conversational-context embeddings. Figure 6.3 shows the relative improvements in the accuracy on the Dev set (7.2.3) over the baseline "non-conversational" model.

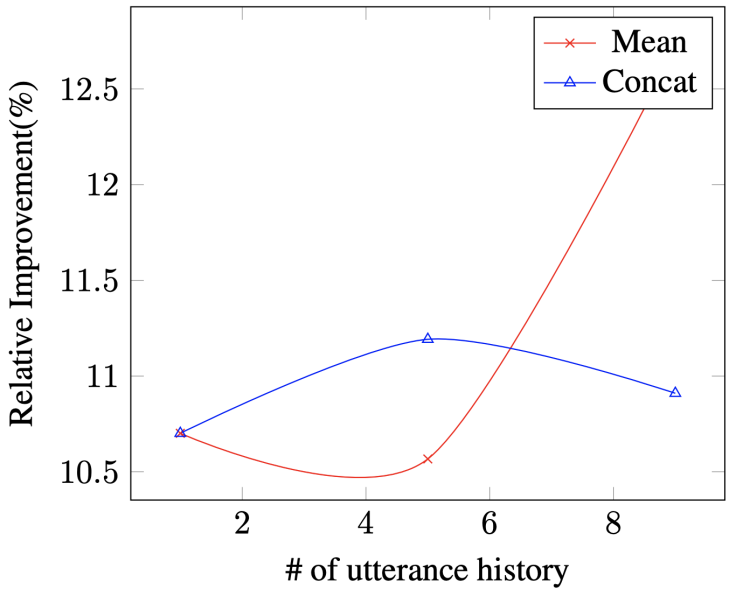


Figure 6.3: The relative WER improvement on validation set with different number of utterance history.

First I observed that more history helps to improve performance generally. However, as I

increase the number of history, I observe the benefit diminishes when we are using concatenation of multiple context embedding. One possible explanation is the number of parameters to be trained increased and it makes the model hard to be learned properly. In the rest of my experiments, I therefore decided not to use concatenation method to integrate the multiple histories since it is not scalable. I also decided to use 10 historical utterances in the rest of my experiments because I found that it performed best when we are using additional attention mechanism instead of mean-pooling. The detailed results will be described in Chapter 7.

6.1.3 Sampling Strategy

Similar to sampling strategy [68] which is widely used in Seq2Seq model for generating previous word embedding, I also consider an utterance level sampling strategy. Since the model does not have access to groundtruth utterance history at inference time and the model predictions itself are always used to encode context, the model may suffer from these mismatch between training and inference. To reduce this mismatch, we choose previous utterances from the groundtruth or from the model outputs for input of context encoder at training time.

I conducted experiments with various sampling ratio, $[0.0, 0.2, 0.5, 1.0]$ to choose model outputs. Figure 6.4 shows the relative WER improvement on validation set with the sampling strategy with various ratio $[0.0, 0.2, 0.5, 1.0]$ to choose model outputs.. I observed that a sampling rate of 0.1 or 0.2 performed best slightly empirically. In the rest of my experiments, I therefore decided to use 0.1.

6.2 Augmentation with "World Knowledge"

Learning better representation of conversational context is the key to achieve better processing of long conversations and it requires massive amount of training data. However, the annotated conversational speech corpus is more expensive to obtain, in contrast with the textual corpus,

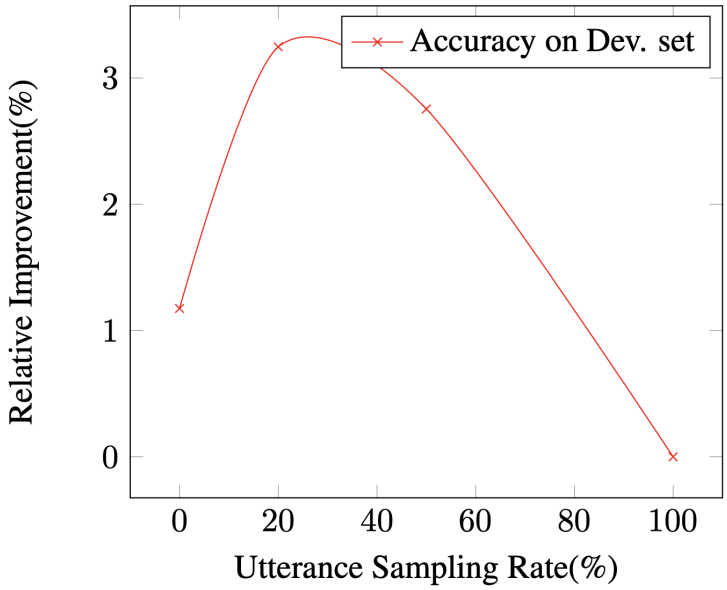


Figure 6.4: The relative WER improvement on validation set with sampling strategy with various ratio $[0.0, 0.2, 0.5, 1.0]$ to choose model outputs.

we typically use thousands of hours annotated corpus even in the industry [69]. For example, in case of the text corpus which is widely used in building language model, BookCorpus [70] and English Wikipedia have 800 million and 2,500 million words, while in case of annotated speech corpus which is widely used in building ASR, Fisher and Switchboard has 2,000 hours of speech recordings and 20 million words in transcriptions. The context embeddings trained only on the transcription is therefore limited to learn rich context information.

In order to address such issues, I propose to use external word or sentence embeddings that are pre-trained on large textual corpora to augment "world knowledge". As the input of my context encoder described in conversational End-to-End speech recognition framework (in previous sections), I use publicly available, external word or sentence embeddings rather than training for the embeddings from scratch with only on limited amount of the transcription.

Another advantage of using pre-trained embedding models is that we do not need to back-propagate the gradients across contexts, making it easier and faster to update the parameters for

learning a conversational context representation.

There exist many word or sentence embeddings which are publicly available. The neural word/sentence embeddings can be broadly classify into two categories: (1) non-contextual word/sentence embeddings, and (2) contextual word/sentence embeddings.

Non-contextual word embeddings, such as Word2Vec [2], GloVe [71], fastText [72], maps each word independently on the context of the sentence where the word occur in. Although it is easy to use, it assumes that each word represents a single meaning which is not true in real-word.

Contextualized word embeddings [73, 74, 75], sentence embeddings, such as deep contextualized word representations [74], Pre-training of Deep Bidirectional Transformers for Language Understanding (BERT) [73], encode the complex characteristics and meanings of words in various context. Such word/sentence embeddings learned on large text corpora, including BooksCorpus (800 million words) and English Wikipedia (2,500 million words). Especially, the BERT model has been used in the form of transfer learning and it has been shown the state-of-the-art performance in a variety of downstream tasks (11 NLU tasks), such as sentence pairs in paraphrasing, hypothesis-premise pairs in entailment, question-passage pairs in question answering, text classification, sequence tagging, sentimental analysis.

6.2.1 External Word Embeddings: fastText

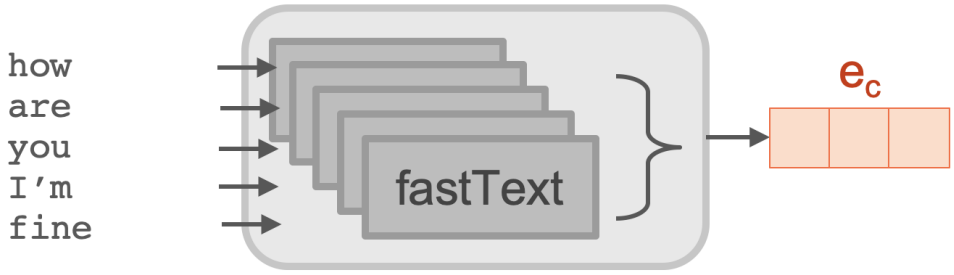


Figure 6.5: My context encoder with external word embeddings (fastText)

I explore both types of embeddings to learn conversational context embeddings as illustrated

in Figure 6.5 and Figure 6.6. The first method is to use word embeddings, fastText, to generate 300-dimensional embeddings from the one-hot vector of preceding utterance histories or distribution over output tokens as we discussed in previous subsection.

6.2.2 External Sentence Embeddings: BERT

Figure 6.6: My context encoder with external sentence embeddings (BERT)

The second method is to use the sentence embeddings, BERT. It is used to a generate single 786-dimensional sentence embedding from the preceding utterances and then merge into a single context vector.

Since my model uses a restricted vocabulary which is different from the external embedding models, we need to handle out-of-vocabulary words. For fastText, words that are missing in the pretrained embeddings we map them to a random multivariate normal distribution with the mean as the sample mean and variance as the sample variance of the known words. For BERT, I use their provided tokenizer to generates byte pair encodings to handle OOV words.

Using this approach, we can obtain a more dense, informative, fixed-length vectors to encode conversational context information, e_c to be used in next utterance prediction.

6.3 Speaker-specific Cross-Attention

If we have access of the speaker identity information for each utterance, then we can consider turn-change information or interaction between two-speakers to process the two-party conversa-

tions better. For example, we can track and learn the interaction of the two speakers based on the history of *what other speaker said* and the history of *what current speaker said*. The overall architecture of my proposed model is described in Figure 6.7. Specifically, my model works as follows.

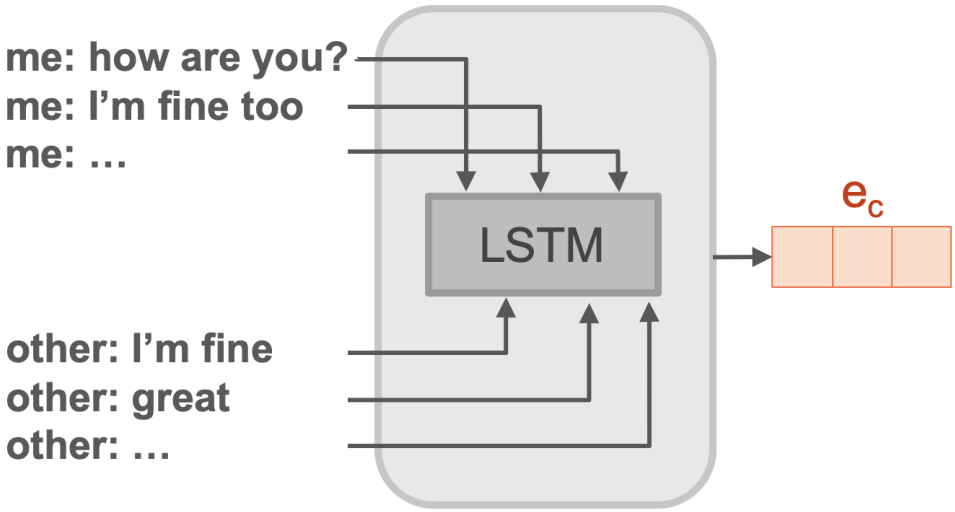


Figure 6.7: My context encoder with LSTM and Attention mechanism for learning the interaction of two-speaker conversations

I create a queue for each speaker to store the history of utterance embeddings so that the utterance embeddings can be stored separately. I then use an attention mechanism to generate speaker-specific conversational context embedding given the history of *what other speaker said* and the history of *what current speaker said*. Note that, based on what current speaker is, I swap the queues properly. I propose two methods to generate the attended context embeddings.

6.3.1 Attention Over Each Speaker's Utterance History

First method is simply using an additional attention mechanism over the utterance embeddings. Given the N-size of the utterance history for speaker A, $e^{u-A}_{k-N}, \cdots, e^{u-A}_{k-1}$, the conversational-

embedding, $\texttt{att\_e}_k^A$, is generated as follows:

$$\mathbf{G}_k^A = \texttt{tanh}(\mathbf{W}e_{k-N:k-1}^A + \mathbf{b}) \tag{6.1}$$

$$\alpha_k^A = \texttt{softmax}(\mathbf{w}^T\mathbf{G}_k^A + \mathbf{b}) \tag{6.2}$$

$$\texttt{att\_e}_k^A = \sum_N a_k^A \odot e_{k-N:k-1}^A \tag{6.3}$$

where $\mathbf{W}$, $\mathbf{b}$ are trainable parameters. e_k^B is generated in the same way.

6.3.2 Cross-attention Between Two Speakers' Utterance History

Second method is using `LSTM` with an attention mechanism. Inspired by the `matchLSTM` model which has been widely used in question answering tasks and natural languge inferance (NLI) [76, 77], we consider to track the interaction between two speakers sequentially, by attending *what other speaker said* at each utterance timestamp. The idea of the `matchLSTM` is to attempt to take the question (premise) and the passage (hypothesis) along with an answer pointer [78] pointing to the start and the end of the answer to make predictions. The `matchLSTM` tries to obtain a question-aware representation of the passage, by attending over the representations of the question tokens for each token in passage.

The key difference in my work is that the question (premise) is a sequence of utterance-embedding from other speaker (*what other speaker said*), and passage (hypothesis) is a sequence of utterance-embedding from current speaker (*what current speaker said*). The embedding of *what current speaker said* takes into consideration the alignment between the *what current speaker said* and *what other speaker said*.

By using `matchLSTM` over the first simple attention method, there are two benefits -

- First, the model is able to handle a longer utterance-history
- Second, the model can learn the interaction between the two speakers, as the `matchLSTM` can potentially track the flow of the conversations.

Specifically, the attended conversational embedding at i-th utterance-history step is generated as follows:

$$\mathbf{G}^A_{k_i} = \texttt{tanh}(\mathbf{W}e^B_{k-N:k-1} + \mathbf{V}e^A_i + \mathbf{U}\mathbf{h}^A_{i-1} + \mathbf{b})$$
$$\alpha^A_{k_i} = \texttt{softmax}(\mathbf{w}^T\mathbf{G}^A_{k_i} + b)$$

where $\mathbf{W}, \mathbf{V}, \mathbf{U}, \in \mathbb{R}^{h\times h}$, $\mathbf{b} \in \mathbb{R}^h$, $b \in \mathbb{R}$ are trainable parameters. Each hidden state $\mathbf{h}_{i-1} \in \mathbb{R}^h$ comes from the output of the `matchLSTM` that is fed the following $\mathbf{z}_i$ as input.

$$\mathbf{z}^A_i = [e^A_i, e^B_{k-N:k-1} \odot \alpha^A_{k_i}]$$
$$\mathbf{h}^A_i = \texttt{matchLSTM}(\mathbf{z}^A_i, \mathbf{h}^A_{i-1})$$

Using a LSTM, there are N such h-dimensional hidden states, and I take the final hidden states for my attended conversational context embedding:

$$\texttt{att\_e}^A_k = \mathbf{h}^A_{k-1}$$

6.4 Training of Context Encoder

My proposed context encoder can be trained towards minimizing the current utterance prediction error in an End-to-End manner. Figure 6.8 shows the training process of my context encoder. The acoustics and words of current utterance, and contexts from past utterances are forwarded into my End-to-End ASR, the loss for the current utterance prediction is then calculated. This loss back-propagated through over entire history of the utterance (or utterance embeddings). The current method is only using linguistic contexts, it can be easily extended to use acoustic contexts as well. We will discuss this in Chapter 9, as future work.

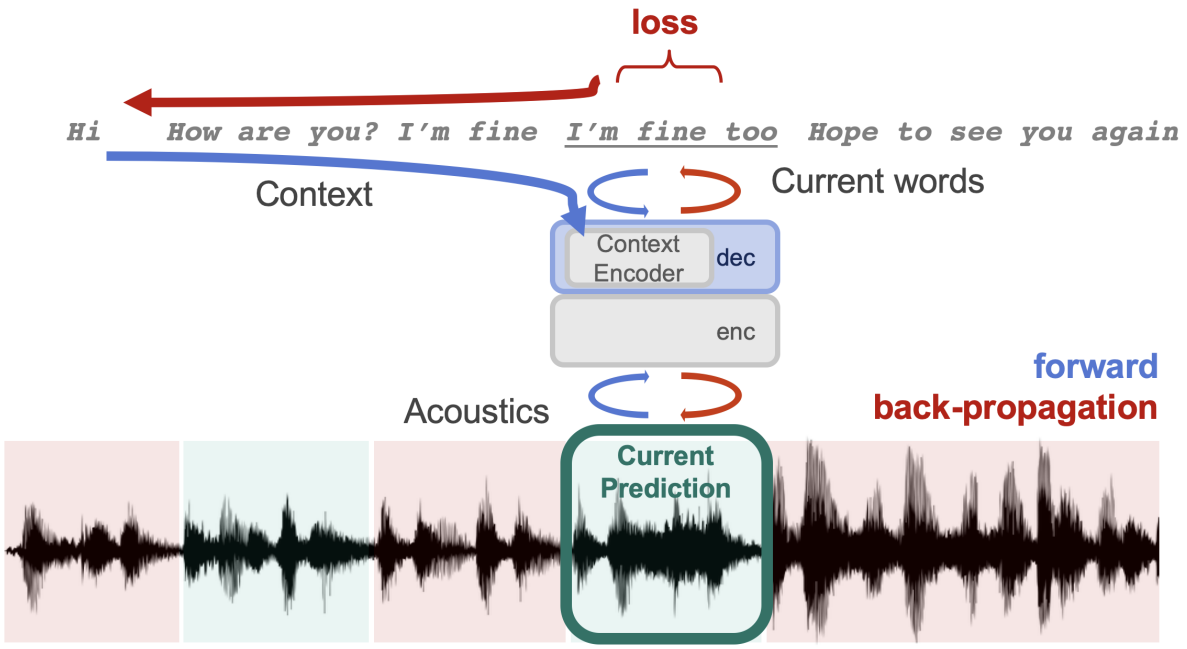


Figure 6.8: My context encoder is designed to be trained over all (or window) of past utterances

6.5 Summary

In this chapter, I proposed various types of my context encoder to encode context embedding better. I also describe how my context encoder is trained over all (or window) over the utterance histories in an End-to-End manne. In the next Chapter 7, I will investigate the WER results across the different types of context encoder in details.

Chapter 7

Experiments to Three Large-Scale Conversational Speech Recognition Tasks

We have discussed how we can preserve and integrate the conversational context information and how to encode conversational context embedding from previously spoken utterances. In this chapter, I will show my conversational End-to-End ASR models improve accuracy on three large scale conversational speech recognition tasks and share the analyses to demonstrate the effectiveness of my conversational End-to-End ASR models.

7.1 Datasets

I evaluate my conversational End-to-End ASR models on three different large scale conversational speech recognition tasks: Switchboard (300h), Fisher (2,000h), and Medical task (1,700h). In Table 7.1, 7.2, 7.3, I describe statistics of training, validation, evaluation sets of each task, the number of utterances, the number of dialogs, the number of words, the average number of utterances per a dialog, the average number of words in an utterance.

7.1.1 Switchboard

I first use the Switchboard (SWBD) LDC corpus (97S62) task [79]. Switchboard is a corpus of recorded telephone conversations. The task involves transcribing conversations between strangers discussing topics such as sports and politics. The total training dataset has 300 hours of recording, so I split 300 hours of training data into two sets: 285 hours of data for the model training, and 5 hours of data for the hyper-parameter tuning. I evaluate the model performance on the HUB5 Eval2000 which consists of the Callhome English (CH) and Switchboard (SWBD) (LDC2002S09, LDC2002T43).

Table 7.1: The description of the dataset for the Switchboard task.

	training	validation	evaluation	
	SWBD	SWBD	SWBD	CallHm
# of words	3,007,098	49,204	18,418	20,847
# of utterances	192,656	4,000	1,831	2,627
# of dialogs	2,402	34	20	20
utters/dialog	80	118	92	131
words/utter	15.6	12.3	10.1	7.9

7.1.2 Fisher

I also use the Fisher dataset, which is also recorded telephone conversations [80]. Compared to the Switchboard task, the Fisher task has a larger number of participants and they made calls of speaking to other participants, whom they typically do not know, about assigned topics. The Fisher dataset has around 2,000 hours of training data. The Fisher data includes 285 hours of SWBD training set as well. I evaluate the model on the same evaluation sets as the Switchboard task: the Callhome (CH) and Switchboard (SWBD).

Table 7.2: The description of the dataset for the Fisher task.

	training	validation	evaluation	
	Fisher	SWBD	SWBD	CallHm
# of words	20,409,563	49,204	18,418	20,847
# of utterances	1,505,869	4,000	1,831	2,627
# of dialogs	11,698	34	20	20
utters/dialog	129	118	92	131
words/utter	13.6	12.3	10.1	7.9

7.1.3 Medical task

Table 7.3: The description of the dataset for the Medical task.

	training	validation	evaluation
	Medical	Medical	Medical
# of words	16,423,024	54,868	127673
# of utterances	2,069,594	6,726	15,111
# of dialogs	13,701	45	100
utters/dialog	151	149	151
words/utter	7.9	8.2	8.4

In addition to the above well-known speech corpora, I also evaluate my models on a medical conversational task. The task involves transcribing conversations between a doctor and a patient in a real and noisy environment in the hospital. Unlike the Switchboard or Fisher tasks, there are no assigned topics, the conversation is recorded under their real diagnosis. This dataset is from UPMC, Pittsburgh hospital, which is unique, not publicly available. The Medical task is more challenging than the Switchboard/ Fisher, because the transcriptions, segments, and alignments

are noisy. This dataset has never been used for speech recognition experiments before, so there are no other benchmark results. The Medical dataset has 1,700 hours of the training dataset.

7.2 Models

In this section, I will describe the architecture and the input/ output of my models for the experiments. The models are implemented using the PyTorch deep learning library [81], and ESPnet toolkit [14, 48, 49]. To build my baseline, I follow the recipe of ESPnet toolkit, except our decoder network has two layers instead of a single layer.

7.2.1 Input Features

All of my experiments in this study, I use an 83-dimensional feature vector for each input frame. From the audio data sampled at 16kHz, the feature vector consists of 80-dimensional log-Mel filterbank coefficients and 3-dimensional pitch features as suggested in [82].

7.2.2 Output Units

In this study, I use a word+character (`WordChar`). I explore the use of word and character for the output units of my model. Direct acoustics-to-word (A2W) models train a single neural network to directly recognize words from speech without any sub-word units, pronunciation model, which significantly simplifies the training and decoding process [63, 64, 65, 66, 67]. In addition, A2W models can learn more semantically meaningful conversational context representations rather than a single character unit. Also, it allows us to exploit the pre-trained external resources like word/sentence embeddings where the unit of representation is generally words. While these benefits, A2W models require more training data compared to conventional sub-word models and additional efforts are needed to handle out-of-vocabulary(OOV) words. In order to mitigate these issues, I first restrict the vocabulary to 10k frequently occurring words. I then addition-

ally use character units and start-of-OOV (sunk), end-of-OOV (eunk) tokens to make my model generate a character by decomposing the OOV word into a character sequence. For example, the OOV word, *rainstorm*, is decomposed into (sunk) *r a i n s t o r m* (eunk) and the model tries to learn such a character sequence rather than generate the OOV token. The `WordChar` contains roughly 10k units (10,034) units including the words (10,000) and the characters (34). By using these output units, we were able to obtain better performed A2W baseline over standard models that use the word only.

7.2.3 Architecture

For the architecture of the End-to-End speech recognition, I used joint CTC/Seq2Seq End-to-End ASR [48, 49] as I proposed in Chapter 3. As suggested in [15, 58], the input feature images are reduced to $(1/4 \times 1/4)$ images along with the time-frequency axis within the two max-pooling layers in CNN. Then, the 6-layer BLSTM with 320 cells is followed by the CNN layer. For the attention mechanism, I used a location-based method [12]. For the decoder network, I used a 2-layer LSTM with 300 cells. In addition to the standard decoder network, my proposed models additionally require extra parameters for gating layers in order to fuse conversational context embedding to the decoder network compared to baseline. I denote the total number of trainable parameters in Table 7.7.

7.3 Training and Decoding

For the optimization method, I use AdaDelta [53] with gradient clipping [59]. I used $\lambda = 0.5$ for joint CTC/Seq2Seq training (in Eq. 3.3) and $\gamma = 0.3$ for joint CTC/Seq2Seq decoding (in Eq.3.4). I bootstrap the training of my proposed conversational End-to-End models from the baseline End-to-End models. To decide the best models for testing, I monitor the development accuracy where I always use the model prediction in order to simulate the testing scenario. At

inference, I used a left-right beam search method [55] with the beam size 10 for reducing the computational cost. I adjusted the final score, $s(\boldsymbol{y}|\boldsymbol{x})$, with the length penalty 0.5.

7.4 The Word Error Rate results

7.4.1 The Word Error Rate (% WER) results of my End-to-End ASR baselines

Switchboard task

In Table 7.4 summarizes the Word Error Rate (% WER) results of the Switchboard (300hr) with `WordChar` output units. The * mark denotes my estimate for the number of parameters used in the other systems.

Table 7.4: Comparison of word error rates (% WER) of Switchboard task of my End-to-End ASR baseline systems and other End-to-End ASR systems.

Model		#params.	LM	SWB	CH
Other End-to-End ASR systems					
CTC [62]	Char output	53M	✓	19.8	32.1
CTC [65]	Word output, phone pretrain	n/a	✗	14.6	23.6
Seq2Seq [83]	BPE-1k, layer-wise pretrain	*150M	✗	13.1	26.1
LF-MMI [84]	Char output, data augment	26M	✓	13.0	23.6
Seq2Seq [85]	BPE-1k, data augment	360M	✗	7.2	14.6
My End-to-End ASR systems					
My baseline	Char output	23M	✗	19.0	34.4
My baseline	Wordchar output	32M	✗	17.9	30.6

I first show the WER of the other systems, and then my baseline systems. Unlike other systems, my systems have relatively smaller model parameters and do not use any phonetic information [65, 86], any external language model [62, 65, 86, 87], or any data augmentation [85],

or any additional, complex training procedure (i.e. layer-wise pretraining [83] or hierarchical multi-task learning [87] or MBR training [88]).

Fisher task

Table 7.5 shows the WER results of the Fisher (2,000hr) task with `WordChar` output units.

Table 7.5: Comparison of word error rates (% WER) of the Fisher task of my End-to-End ASR baseline systems and other End-to-End ASR systems.

Model		#params.	LM	SWB	CH
Other End-to-End ASR systems					
CTC [62]	Char output	n/a	✓	10.2	17.7
CTC [65]	Word output, phone pretrain	n/a	✗	8.8	13.9
LF-MMI [84]	Char output, data augment	26M	✓	12.0	21.9
Seq2Seq [89]	Char output	120M	✗	8.6	17.8
Seq2Seq [88]	Char output, MBR	n/a	✗	8.3	15.5
My End-to-End ASR systems					
My baseline	wordchar output	32M	✗	14.4	21.9

Unlike other systems, my systems again have relatively smaller model parameters and do not use any phonetic information [65], any external language model [62, 84], or any data augmentation [84], or MBR training [88]).

Medical task

Table 7.6 shows the WER results of the Medical (1,700hr) task with `WordChar` output units. This dataset is from UPMC, Pittsburgh hospital, which is unique, not publicly available, so there are no other benchmark results.

Table 7.6: The word error rates (% WER) of the Medical task of my End-to-End ASR baseline systems.

Model		#params.	LM	Medical
My End-to-End ASR systems				
My baseline	Wordchar output	32M	✗	22.6

7.4.2 The WER results of my conversational End-to-End ASR models

In Table 7.7 summarizes the Word Error Rate (% WER) results of Switchboard (300hr) with `Wordchar` output units.

Table 7.7: Comparison of word error rates (% WER) of my baseline and my proposed conversational End-to-End ASR on the SWBD task.

Model	#params.	LM	SWB	CH
my baseline	32M	✗	17.9	30.6
My Context End-to-End ASR				
Vanilla Context	33M	✗	17.3	30.3
+ Gate	35M	✗	17.2	29.8
+ BERT Context	34M	✗	**15.5**	29.0
+ LSTM-Attn for 2spk	34M	✗	15.6	**28.5**

I show the improved WER results by adding each of the individual components that we discussed in previous sections. The `vanilla` and `vanilla + gate` models use one-hot vectors for encoding conversational context, with standard decoder and with the gated decoder, respectively. The `+ BERT` model uses BERT external language model for encoding conversational context with the gated decoder. The `+ LSTM-Attention` model uses 2 speaker information as described in Chapter 6 in addition to use BERT external language model for encoding conversational context with gated decoder. As shown in this Table 7.7, my best model gets around 12.8% relative improvement on the Switchboard (SWBD) evaluation set and 6.9% relative im-

provement on the CallHome (CH) evaluation set. I observed that the simple vanilla model only obtained 2.6 - 3.9% relative improvement, however, when I augmented world knowledge by using an external language model to encode conversational context, my model obtained significant relative improvements (12.8%, and 6.9%).

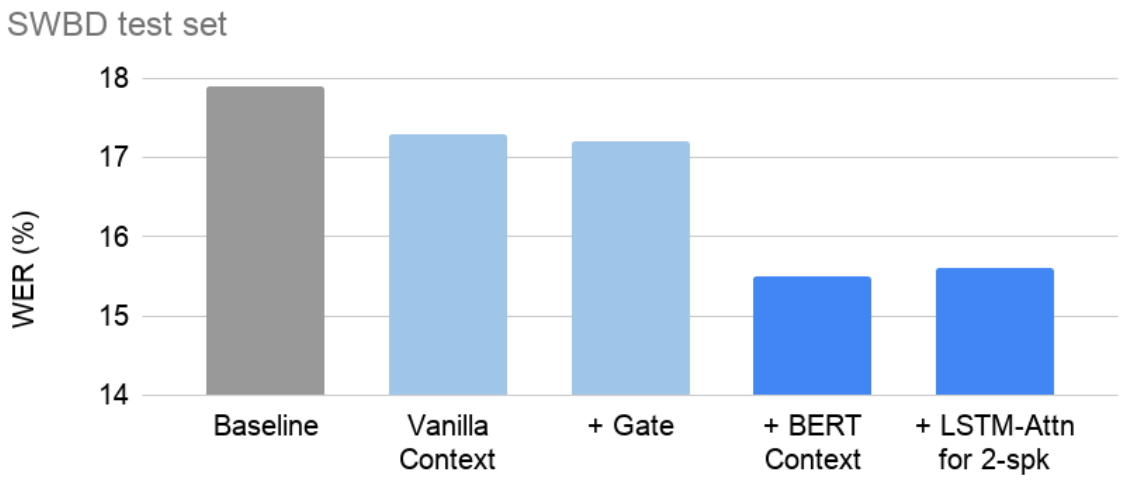


Figure 7.1: % WER on SWBD evaluation set over different proposed context encoder methods

I also investigate the effect of the number of utterance history being encoded. I tried different $N = [1, 5, 10]$ number of utterance histories to learn the conversational context embeddings. Typically increasing the number of utterance histories improves accuracy. However, I also observed that the improvement diminished when I used 9-utterance history in case of using the concatenation. It is possible that the increased number of trainable parameters of the concatenate model makes it harder for the model to train. I use 10-utterance histories to encode conversational context embedding in the rest of the experiments. The WER over different types of proposed context encoder is also illustrated in Figure 7.2.

Table 7.8 and 7.9 show the WER results on a larger dataset, Fisher (2,000 hours), and Medical (1,700 hours). As seen in these Tables, my proposed approach outperformed over baseline in

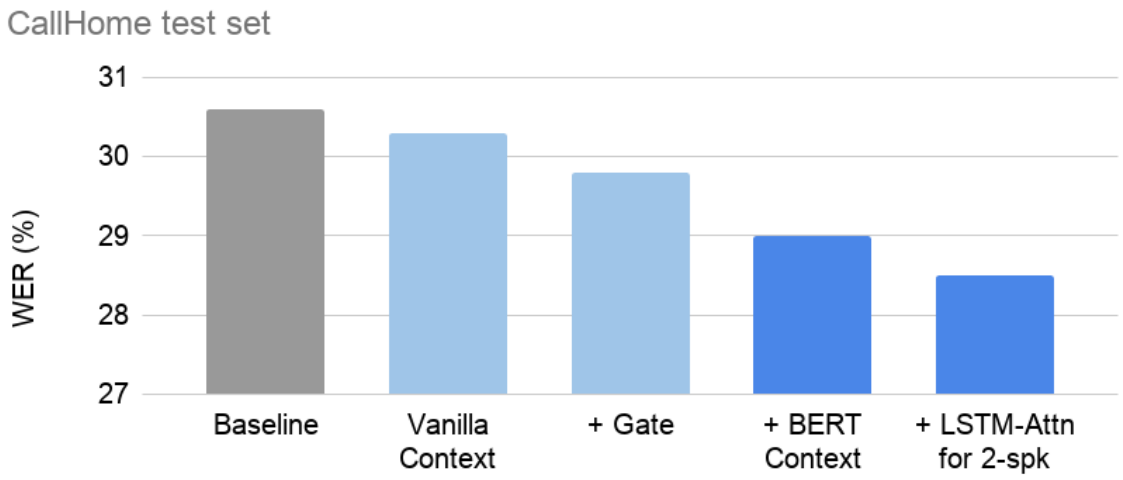


Figure 7.2: % WER on CallHome evaluation set over different proposed context encoder methods

Table 7.8: Comparison of word error rates (% WER) of baseline and my proposed conversational End-to-End ASR on the Fisher dataset.

Model		#params.	LM	SWBD	CH
My End-to-End ASR systems					
My baseline	Wordchar output	32M	✗	14.4	21.9
My context End-to-End ASR	Wordchar output	34M	✗	13.2	21.5

large dataset consistently.

[90] is a conversational speech recognition model that relies on an extensive system combination for the acoustic model at senone and word levels, followed by N-best rescoring with multiple language models in a confusion network, and adding backchannel penalties. It uses twice the number of parameters as my baseline, and multiple acoustic models and language models to produce the best score on this task thus far, to the best of my knowledge. My baseline does not use an external LM like [62], or Minimum Bayes Risk Training like [88].

Table 7.9: Comparison of word error rates (% WER) of baseline and my proposed conversational End-to-End ASR on Medical dataset.

Model		#params.	LM	Medical
My End-to-End ASR systems				
My baseline	Wordchar output	32M	✗	22.6
My context End-to-End ASR	Wordchar output	34M	✗	21.7

7.5 Statistical Significance

To understand the "statistical significance" of WER improvement using my conversational End-to-End ASR models, I performed a significant test on the WER results of three tasks: SWBD, Fisher, Medical tasks. I used a `compute-wer-bootci` tool, bootstrap method for statistical significance test proposed in [91] which is provided in the Kaldi library [92]. This tool calculates the confidence intervals that provide a range of reliability defining how reliable it is an observed improvement from my conversational models. Based on the boostrap technique that extracts WER for a number of replications (10^4) of the same size of test sets, they also provide the "probability of improvement", which is the relative number of bootstrap sample WERs which favor my models.

The formulation of ΔW, the difference in the number of errors of two systems: baseline and conversational model, on the same test sets is as follows:

$$\Delta W = W^{base} - W^{conv} \tag{7.1}$$

$$= \frac{\sum_i (e_i^{base} - e_i^{conv})}{\sum_i n_i} \tag{7.2}$$

where $e_i^{base} e_i^{conv}$ is the word error count of utterance i with two systems, baseline and conversational model, n_i is a total number of the word in utterance i.

Based on this ΔW, the probability of improvement (poi) is then computed as follows:

$$poi = P(\Delta W < 0) \tag{7.3}$$

$$= \frac{1}{B} \sum_{b=1}^{B} \Theta(-\Delta W^b) \tag{7.4}$$

where $\Theta(x)$ is the step function, which is one for $x > 0$ and zero otherwise.

Table 7.10: Probability of improvement of my conversational models over baselines based on 95% confidence intervals.

Task	SWBD	Fisher	Medical
Training set size (hours)	300	2,000	1,700
# dialogs in Training set	2.4k	11.7k	13.7k
# utters in Test set	4k	4k	15k
# dialogs in Test set	40	40	98
WER of Baseline (%)	24.28	18.17	22.60
WER of Context model (%)	22.08	17.37	21.70
Relative Improvement (%)	**9.8**	**5.1**	**4.0**
Probability of Improvement (%)	**100.00**	**100.00**	**100.00**

Table 7.10 shows the WER results of baselines and my context models on three tasks: SWBD,

Fisher, and Medical tasks, and with bootstrap statistical significant test results. I found that the empirically observed differences in WER performance between baseline and my context model are due to a genuine advantage of my system over the baseline, not just an effect of chance. I also observed that the probability of improvement is 100.00 % in all my context models. In this test, I use 95% confidence intervals based on standard error and bootstrap $B = 10^4$ [91].

7.6 Summary

In this chapter, I showed my conversational context End-to-End ASR models improves WER on three different conversational speech recognition tasks: SWBD, Fisher, Medical tasks. I observed that my conversational End-to-End ASR model outperforms over baseline consistently and statistically significant. In the next Chapter 8, I share several in-depth analyses results to understand my models better.

Chapter 8

Analytic Methods for Conversational Context Models

I have shown my thesis idea that including conversational context into End-to-End ASR can help to improve WER on three large scale conversational speech recognition tasks and it is statistically significant. In this chapter, I will share in-depth analyses to understand my conversational End-to-End ASR models better. I conducted several in-depth analyses to answer the following questions: *"How my conversational End-to-End ASR models work?"*, *"In which condition my models work well or not?"*, *"Which historical utterances my models attend to more?"*, *"How does the strength of the AM affect the impact of my context models?"*, and *"How do large & small training datasets affect the impact of my context models?"*. I will also show and compare several examples of reference and the hypotheses transcriptions from the baseline and my context models. For the rest of my experiments of the in-depth analyses, I used my best performed conversational End-to-End ASR model which is using the 10 historical utterances with the speaker-specific attention mechanism and using the BERT pre-trained LM for encoding the conversational context to augment "world knowledge". The models used for the in-depth analyses were trained on 300 hours of the training dataset. Note that the "context model" in the rest of the sections refers to my best performed conversational End-to-End ASR model. I also used

"context LM" in the several in-depth analyses and checked the perplexity relative improvement. The "context LM" has the same model architecture as "context model" but the encoder network is completely removed. These ablation studies are for better understanding our linguistic "context model" by ruling out the AM effect.

8.1 Analysis of Context Models

In this section, I analyze in which conditions my context model work well. In the first two subsections 8.1.2, and 8.1.3, I will show that my context model works well when 1) the historical utterances are similar to the current utterance and 2) the historical utterances are more informative. In subsections 8.1.4, and 8.1.5, I will show my context models tend to attend to a longer utterance. In 8.1.6, and 8.1.7, I will show that the improvement provided by my linguistic context model is reduced as the acoustic model part gets stronger due to large training datasets. In 8.1.9, I will show the substitution, deletion, and insertion rate in WER.

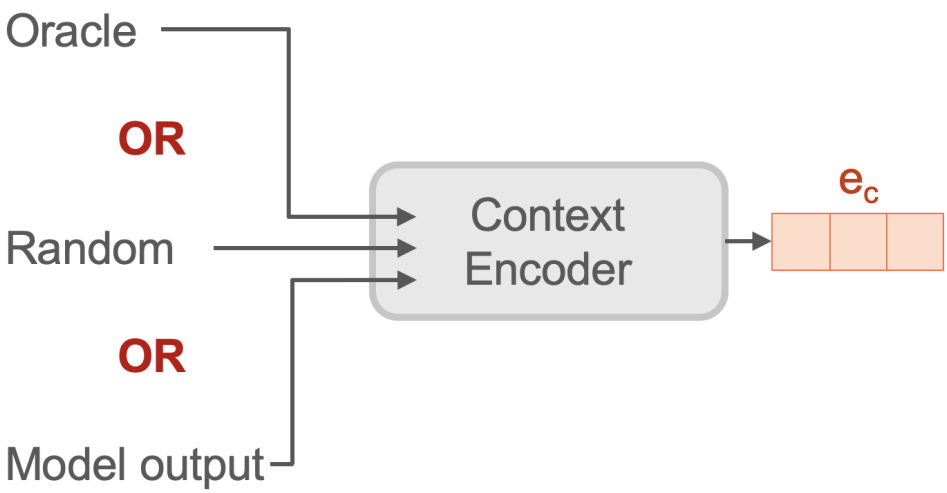
8.1.1 Comparison of Oracle and Random Context Embeddings

In addition to simply comparing WER performance, I show my analysis to demonstrate the effectiveness of my conversational End-to-End ASR first. I validate the effect of my context embeddings on performance improvement.

To do so, I generated two context embeddings: `Oracle` and `Random`. In general, my model always uses previously spoken utterances from the model output at inference time. For `Oracle` context embeddings, I used the ground-truth transcriptions as an input of context encoder instead of `Model output`. For `Random` context embedding, I generated random output tokens and forwarded it to the context encoder as illustrated in Figure 8.1. I then compared WER of my conversational End-to-End ASR and checked how WER changes as different context embeddings.

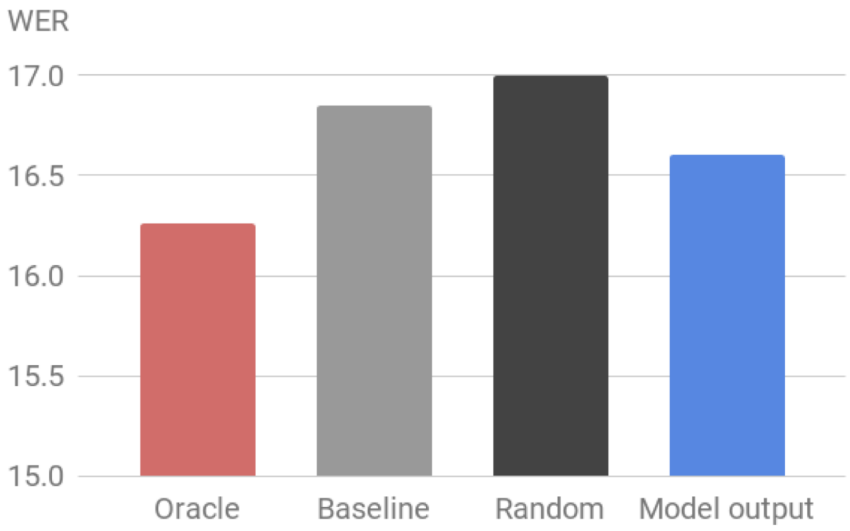
As in Figure 8.1, the `Oracle` case performed best as I expected, while the `Random` case was

Figure 8.1: Context Encoder with different input at inference time: oracle/ random/ model outputs



even worse than the baseline which does not use any context information. The `Model output` case, our general usage, outperformed over the baseline and the `Random` case. This implies that the accuracy benefit of my proposed method is coming from actually learning the conversational context, not just from randomness or regularization.

Figure 8.2: Comparison of % WER of different inputs of my context encoder at inference time: oracle/random/ model outputs



8.1.2 Analysis of Conversational Similarity of Historical Utterances

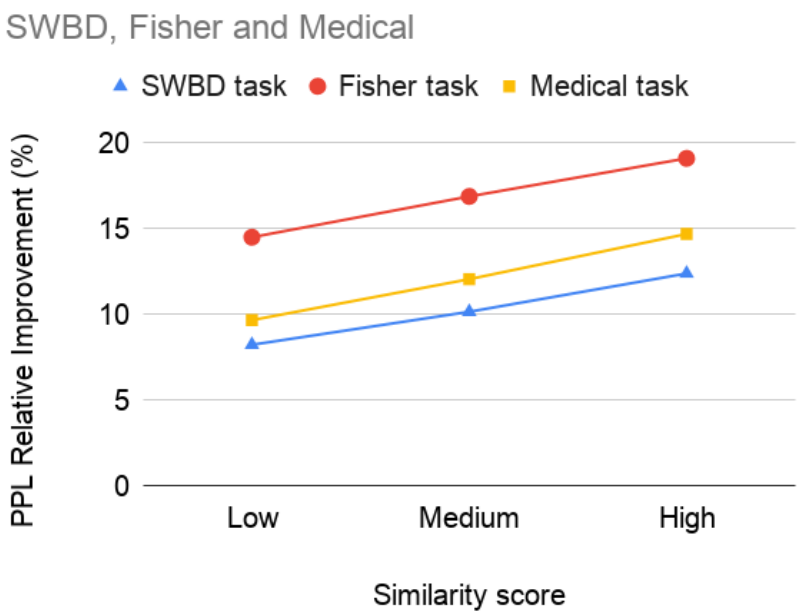
I first analyze whether the similarity between historical utterances and currently predicted utterance affects the impact of my context model. I develop a conversational similarity score function, $s(i)$ that measures how similar utterance i with historical utterances in the same dialog. I used the [1 - 10] historical utterances and calculate the Cosine similarity between the current utterances. I then used an average of 10 cosine similarities as my conversational similarity score for utterance i, $s(i)$. Before calculating the Cosine similarity, I mapped each utterance to a single vector by using BERT sentence embedding. I followed a standard way to generate the sentence embedding from the pre-trained BERT model by using an average of each output tokens from the last layer. Mathematically it can be written as,

$$s(i) = \frac{1}{n}\sum_{j=1}^{n}\frac{e_i \cdot e_{i-j}}{||e_i||||e_{i-j}||}$$

where, e_i, e_{i-j} is BERT sentence embeddings of current utterance and historical utterances, n is the number of historical utterances the model uses to predict the utterance i.

Figure 8.3 shows the relationship between conversational similarity score and perplexity relative improvement of my context model over baseline. I used the transcriptions of the evaluation sets and split utterances into three chunks in its similarity score $s(i)$: Low, Medium, and High. Each chunk represents a third of the utterances in the test data, roughly. Then, I checked the relative improvement in the perplexity of my context LM over baseline LM for each chunk of utterances. The mean values of conversational similarity of the Low, Medium, and High were 0.4, 0.5, and 0.6, respectively. As shown in Figure 8.3, I observed that my context model performs better when historical utterances and current, predicted utterances are similar. I also observed that my context LM which is trained on 300 hours of Fisher dataset shows the largest relative improvement over baseline LM, the red line (Fisher) is above the other lines (SWBD and Medical). However, the effect of the conversational similarity score shows similar trends across all

Figure 8.3: Comparison of performance improvement of my context model with different conversational similarity score in three tasks: SWBD, Fisher, and Medical



the tasks.

8.1.3 Analysis of Informativeness of Historical Utterances Based on TF-IDF

I also analyze whether an average of the informativeness score of historical utterances affects the impact of my context model. I first develop the informativeness score for each utterance, $info(u)$. The informativeness score function measures how many rare words contain in the utterance u. This is based on 'TermFrequency-InverseDocumentFrequency' (`TF-IDF` [93]) which is one of the popular term-weighting schemes in searches of information retrieval, text mining, and user modeling [94].

Mathematically, the informativeness score for each utterance is an average log inverse frequencies of all the words that appear in the utterance. The frequency of each word measures how frequently a word occurs in the training transcripts. The info score function based on `TF-IDF`

for the utterance u is defined as follows:

$$\texttt{TF-IDF}_{uttr}(w_i) = \text{frequency}_u(w_i) / \log(\text{frequency}_{train}(w_i)) \tag{8.1}$$

$$\text{info}(u) = \frac{1}{k} \sum_{i=1}^{k} \texttt{TF-IDF}_{uttr}(w_i) \tag{8.2}$$

where $\text{freq}(w_i)$ is a frequency of word w_i, $\text{count}(w_i)$ is the number of times word w_i appears in the training transcriptions, $\text{info}(u)$ is an informativeness score of utterance u, and k is a number of the word in utterance u. Note that the score is normalized for the utterance length. I then analyze the average of the informativeness score of [1 - 10] historical utterances in the same dialog.

Figure 8.4: Comparison of performance improvement of my context model with different TF-IDF scores in three tasks: SWBD, Fisher, and Medical

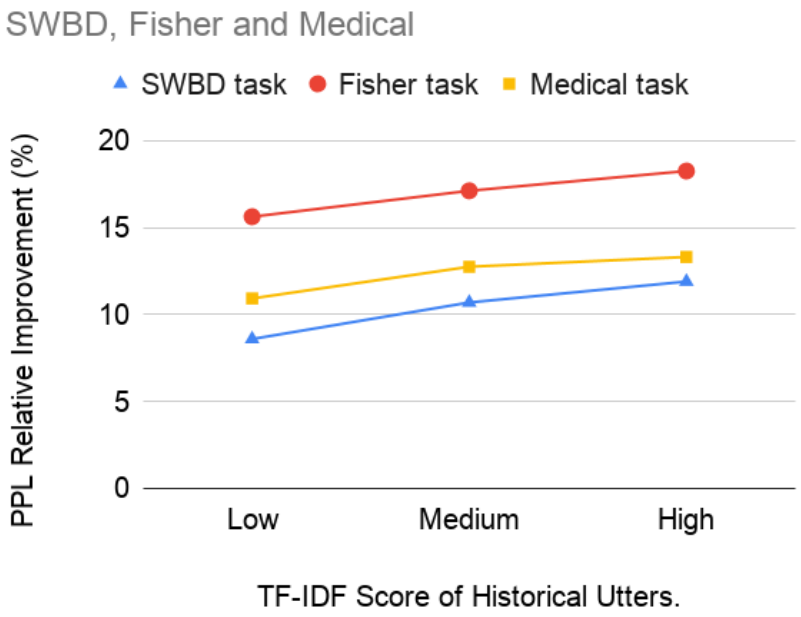


Figure 8.4 shows the relationship between the average of the informativeness score of historical utterances and the perplexity relative improvement of my context model over baseline. I used the transcriptions of the evaluation sets and split utterances into three chunks in its infor-

mativeness score: Low, Medium, and High. Each chunk represents a third of the utterances in the evaluation set, roughly. Then, I checked the relative improvement in the perplexity of my context LM over baseline LM for each chunk of utterances. The mean values of the informativeness score of each of low, medium, and high were [6.9, 8.3, 9.2] (SWBD task), [7.4, 8.7, 10.1] (Fisher task), [7.8, 8.7, 9.6] (Medical task), respectively. As shown in Figure 8.4, I observed that my context model performs better when historical utterances have more rare words.

8.1.4 Analysis of Attention Weights of Context Models

I next analyze how the model's attention work and where the model tends to attend. I used the utterance length and split utterances of the evaluation set into three chunks: Short, Medium, and Long. Each chunk represents a third of the utterances in the evaluation set, roughly. Then, I checked the average of the attention weights of each chunk. The mean values of the utterance lengths of each of the Short, Medium, and Long were [2.4, 8.8, 25.5] (Eval2000), and [2.5, 7.6, 22.9] (Medical task), respectively. In this analysis, I used the attention weight over the 10 historical utterances. As shown in Figure 8.5 my context model tends to attend longer utterances.

I also analyze how the recency of historical utterances affects on models' attention weight. I grouped historical utterances based on its recency from 1 to 10 (higher is older), and checked the average of attention weights of each group. As shown in Figur 8.6, I found that the recency of historical utterances is not strongly related to attention weights.

These results from the analytic methods confirm our previous hypothesis that the context models tend to attend more a longer historical utterance. However, I found that there is no clear correlation between the recency of the historical utterances and the model's attention weights [95].

Figure 8.5: Comparison of models' attention weights on different lengths of historical utterances in three tasks: SWBD, Fisher, and Medical

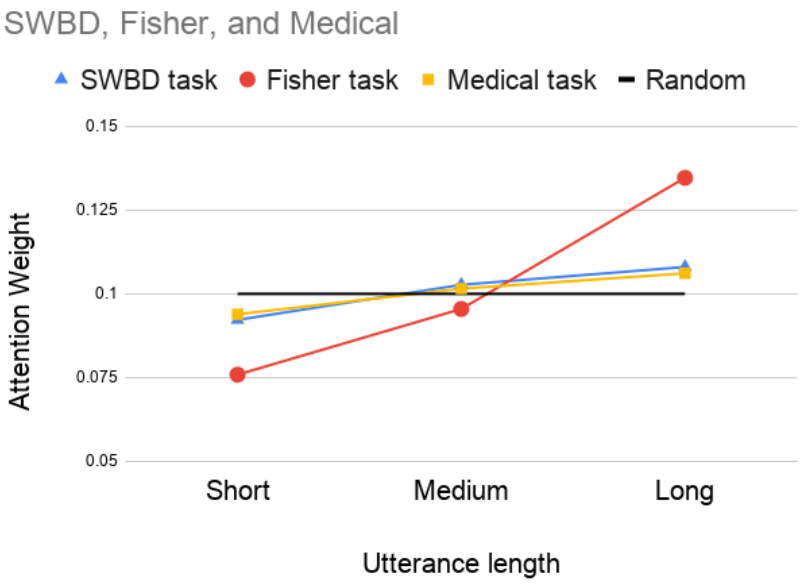
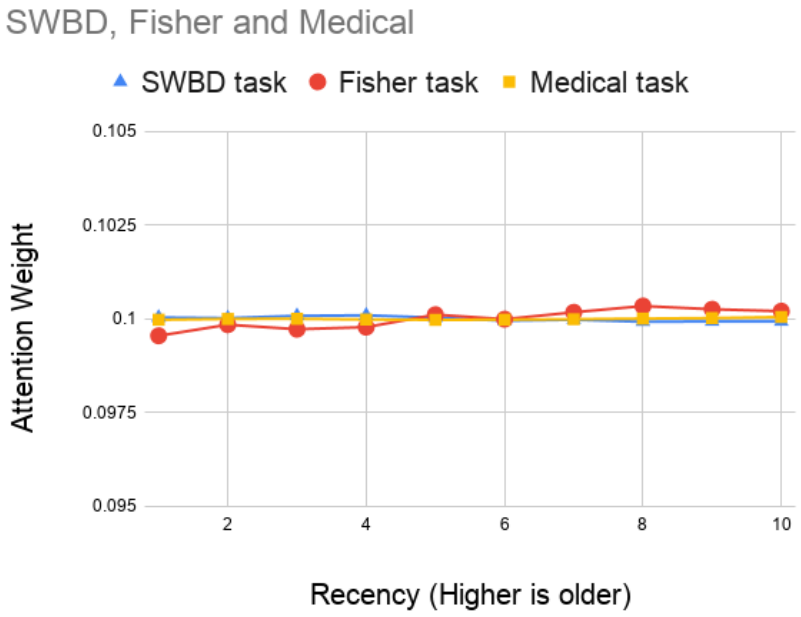


Figure 8.6: Comparison of models' attention weights on the recency of historical utterances in three tasks: SWBD, Fisher, and Medical tasks



8.1.5 Visualization of Attention Weights Over Historical Utterances with Examples

Figure 8.7, and 8.8 visualize the attention weights over the historical utterances and corresponding transcript. The example (in 8.7 is selected from the eval2000 when the model predicts the utterance (*en_4574-a_042366-042850 deliver the papers to a guy that is in a certain location in the immigration area and then*). The example (in 8.8 is selected from the Medical evaluation set when the model predicts the utterance (*98164_pt-052172-052323 uh naprosyn can mask*).

The dark color represents higher attention weight. The results consistently show that my model tends to focus more on a long utterance, rather than a short utterance, i.e. *jeez*, *oh*, *well maybe* etc, in all three tasks.

Figure 8.7: The visualization of the attention weights over [1-10] historical utterances. The example is selected from the eval2000 set.

Historical utterances	*Attention weight*	*Text*
en_4574-B_038830-038870	0.06	oh
en_4574-A_038906-039299	0.22	you carry somebody else's luggage or something else for somebody
en_4574-B_039203-039380	0.07	drugs
en_4574-A_039373-039442	0.06	well maybe
en_4574-B_039396-039896	0.07	do not carry anything illegal and if you are going to do not discuss it now on the taped conversation
en_4574-A_039846-040541	0.16	really well it is really like being a spy you should see these instructions i have you know it <sunk> s a y ' s <eunk> like you have to get to the airport and
en_4574-A_040572-041103	0.07	look for a guy in a blue jacket with a certain kind of s- <sunk> i n s i g n i a <eunk> on it and he will give you
en_4574-A_041130-041790	0.14	papers and take you to the luggage place and check you in and then the luggage will go and then you have one carry on piece
en_4574-A_041824-041955	0.06	that you get to take and then
en_4574-A_042041-042321	0.09	and then when you get to japan you have to

8.1.6 Analysis of Impact of the Strength of AM (Encoder Network) on Improvements of Context Models

In this subsection, I analyze how does the strength of the acoustic model part (encoder subnetwork) affects the impact of my linguistic context model. As shown in Table 7.10 in the

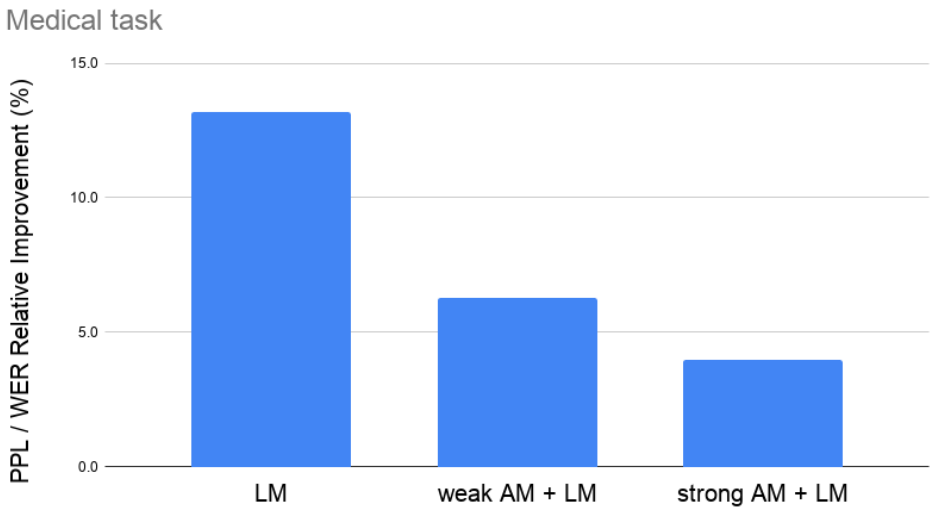
Figure 8.8: The visualization of the attention weights over [1-10] historical utterances. The example is selected from the Medical evaluation set.

Historical utterances	*Attention weight*	*Text*
98164_PT-048714-048839	0.09	fevers do
98164_PT-048839-049367	0.11	do you not recently i uh i i could have i think i had a fever
98164_PT-049367-049615	0.11	the tuesday last week that's when i went home
98164_DR-049807-049877	0.10	how high was
98164_PT-049877-049922	0.10	it do you know
98164_PT-049929-051036	0.12	uh no i don't but i had the chills and i was like feeling really bad and so i took um i took <sunk> n y q u i l <eunk> that night before i went to sleep
98164_DR-051233-051275	0.08	all day
98164_PT-051590-051829	0.10	had a fever then i don't think i had a fever by the time i
98164_DR-051829-051867	0.07	came
98164_DR-051867-052167	0.12	in on thursday but [inaudible] motrin sometimes

previous chapter 7, my context models significantly improve the recognition accuracy in three large-scale conversational speech recognition tasks: SWBD, Fisher, and Medical tasks, however, I observed that the WER relative improvement provided by my context models slightly reduced in Fisher and Medical tasks compared to the SWBD task. I hypothesized that the large training data set produces a strong AM affect the impact of my linguistic context model in Fisher/Medical tasks. To validate this hypothesis, I built three sets of baseline and context models with three different strengths of AM part and checked the relative performance improvement provided by my context models. First, I built LM and context LM which is completely ruled out the AM and checked the perplexity relative improvement. Second, I built the baseline ASR model and context End-to-End ASR model with weak AM by reducing the encoder layers from 6 to 1 and checked the WER relative improvement. Third, I showed the WER relative improvement of the standard baseline and standard context model with the strong AM (6 encoder layers).

Figure 8.9 shows the results of three relative improvements: 1) in the perplexity of context LM which is ruled out AM completely, 2) in WER of context ASR with weak AM, and 3) in WER of context ASR with strong AM. The results showed that the benefit of my linguistic context model gets smaller as the AM gets stronger.

Figure 8.9: The effect of the strength of AM on my linguistic context model



8.1.7 Analysis of Impact of Large & Small Training Datasets on Improvements of Context Models

To confirm the training data size affects the improvement of my context models, I made the Fisher/Medical datasets in a size similar to the SWBD and performed the same experiments. Figure 8.10 shows the effect of the size of the training dataset on my context model. I observed improved benefit of my context LM and context ASR (LM+AM) in small datasets (blue bars).

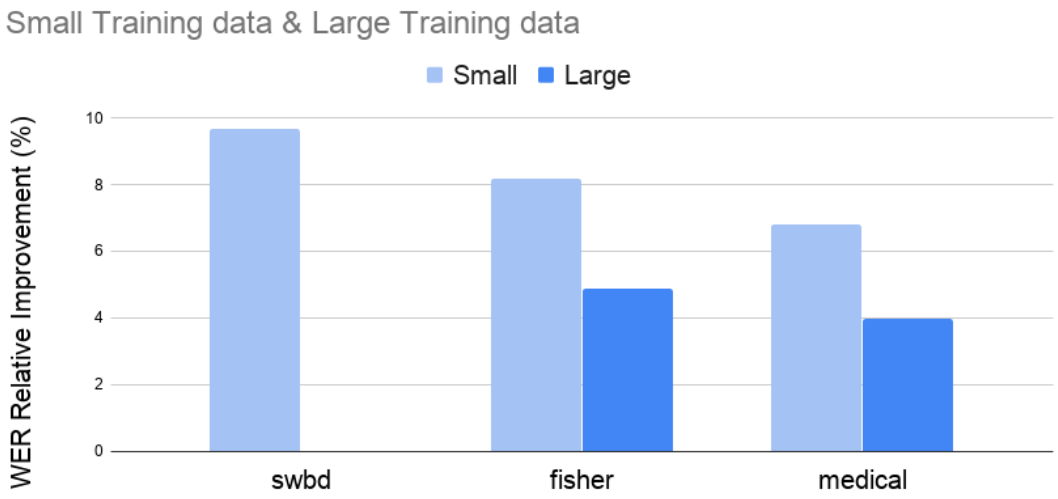
Overall, my linguistic context model performs less effectively with a large training dataset due to the strength of the AM from the large training dataset.

8.1.8 Analysis of Domain-Specific Task (Medical Task) with Speaker Identity

In this subsection, I performed several additional analyses on Medical task to understand how my context model works in domain-specific tasks.

I analyzed the gains of my context models with respect to speaker identity information. Figure 8.11 shows that the relative WER improvement of my context models with the different

Figure 8.10: The effect of the size of the training dataset on my context model



major speaker (doctor and patient) of the historical utterances in Medical tasks. I split utterances of the evaluation set into two chunks: DR, PT. The DR group represents the case that there are more historical utterances spoken by the doctor, and the PT group represents the case where more historical utterances spoken by the patient. I observed that my context model performed better when the historical utterances were spoken by the doctor. This result may indicate that my context model can help predict answers to questions by the doctor. Figure 8.12 shows that the relative WER improvement of my context models with the different speaker (doctor and patient) of the current utterance. I observed that my context model performed better at predicting the utterances spoken by doctor.

I also compared the WERs of two models: one is trained on Fisher (2,000 hours) + Medical training datasets (1,700 hours) and the other is trained on only Medical training datasets (1,700 hours). I observed that the models trained only on Medical training data outperformed by 2.3% relative improvement over the models trained on Fisher (2,000 hours) + Medical training datasets (1,700 hours).

Figure 8.11: Comparison of performance improvement of my context model with different major speaker identity (doctor and patient) of the historical utterances.

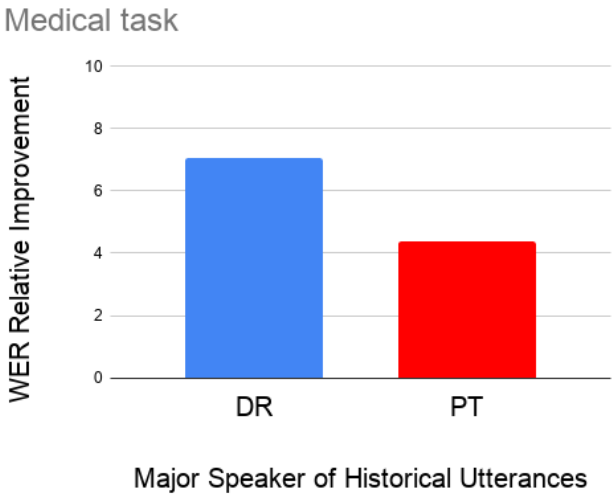
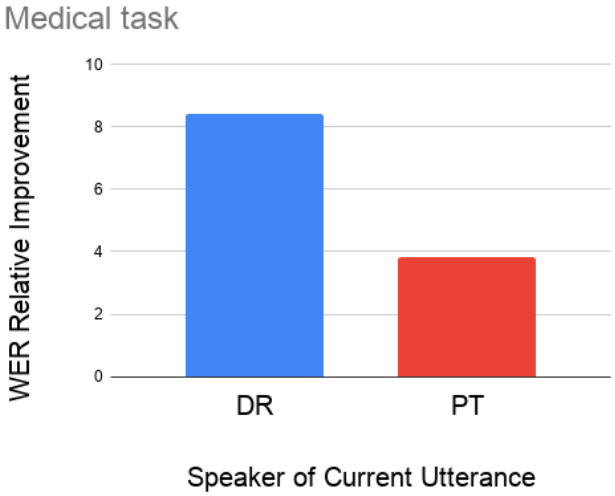


Figure 8.12: Comparison of performance improvement of my context model with different speaker identity (doctor and patient) of the current utterance.



8.1.9 Error Analysis: Substitutions, Deletions, and Insertions error rates

Table 8.1 shows the substitution, deletion, and insertion rates in WER result on SWBD, Fisher, Medical tasks. I observed that the largest factor of WER improvement was from the substitution rates rather than deletion or insertion.

Table 8.1: Substitution (% Sub), Deletion (% Del), and Insertion (% Ins) error rates of baseline and my context models in three different tasks: SWBD, Fisher, and Medical tasks.

Task	WER	Sub	Del	Ins
SWBD task - SWBD				
Baseline	17.9	12.3	3.9	1.7
My Context model	15.6	10.6	3.3	1.7
SWBD task - CallHome				
Baseline	30.6	20.7	6.5	3.3
My Context model	28.5	19.5	5.9	3.2
Fisher task - SWBD				
Baseline	14.4	9.6	3.1	1.6
My Context model	13.2	8.7	2.8	1.7
Fisher task - CallHome				
Baseline	21.9	14.8	4.6	2.6
My Context model	21.5	14.1	4.7	2.6
Medical task				
Baseline	22.6	12.8	5.6	4.2
My Context model	21.7	12.2	5.5	4.0

Decoding with the External Language Model

I analyze how do joint training and disjoint training with conversational context affect the impact of my context models. I first built the conversational context language models separately by using the same context encoder and contextual gated decoder on each training transcript: SWBD,

Fisher, Medical. I then decoded my baseline End-to-End ASR models with the conversational context language models. During the beam-search, I performed the shallow fusion of the language model. Table 8.2 shows the WER results of my baseline End-to-End ASR models decoded with the external language model. I only observed that the improvement by the shallow fusion of external LM is minimal (0.6 - 2.0 % relative improvement in the SWBD task). These results indicate that joint training with the conversational context is important to obtain a significant improvement by conversational context information. This result also suggests that the accuracy gain of my conversational End-to-End ASR is coming from avoiding errors due to acoustics rather than errors due to language model history by joint training with the conversational context.

Table 8.2: The WER results of baseline models with the external language model.

Task	ext. CLM	SWBD (SWBD/CH)	Fisher (SWBD/CH)	Medical Medical
Baseline	✗	17.9/30.6	14.4/21.9	22.6
Baseline	✓	17.8/30.0	14.3/22.1	22.6
Contex model	✗	15.6/28.5	13.2/21.5	21.7

8.2 Examples

Figure 8.13, and 8.14, show examples from SWBD tasks, and Figure 8.15, and 8.16, show examples from the Medical dataset. Each column shows reference utterances, the hypothesis of the baseline model, and the hypothesis of my conversational model in chronological order. The final row shows the corresponding conversational similarity score as I described in the previous section 8.1.2. The higher similarity score represents that the historical utterances and the current, predicted utterance are more similar.

Figure 8.13 shows the first example which was selected from `eval2000` evaluation set. The

Reference	*Baseline*	*Context E2E*
oh boy you guys been all over ***you guys*** *been all over*	*oh boy you guys been all* ***he goes*** *been all over*	*oh boy you guys been all* ***you guys*** *have been all over*
0.882	*0.828*	*0.831*

Figure 8.13: Examples of reference, the hypothesis from baseline, and my conversational End-to-End ASR selected from `eval2000` evaluation set.

blue words, "you guys", in the reference column represent targets. The baseline model could not predict it ("he goes"), while my context model could predict it correctly. As seen in the preceding utterance "oh boy you guys been all", the current prediction may benefit from the repeated word in utterance history. I also observed that the similarity score of the hypothesis of my context model is slightly higher than the one of the baseline.

the the name does not really matter *okay we- my point is the experience* *uh-huh* *you know and specialization* *if if i have* ***specialization*** *in three areas*	*than the name was really not can't* *well my point is the uh experience* *uh-huh* *and special* *if i have* ***fushaliziation*** *in three air yes*	*then the name was really natural okay* *well my point is is the uh experience* *uh* *and specializat* *if if i have* ***specialization*** *in three areas*
0.894	*0.889*	*0.893*

Figure 8.14: Examples of reference, the hypothesis from baseline, and my conversational End-to-End ASR selected from `eval2000` evaluation set.

Figure 8.14 shows the second example which was selected from `eval2000` evaluation set. The blue word in the reference column is "specialization". The baseline could not predict it ("fushaliziation"), while my model could predict it correctly. Even though the preceding hypothesis was partially correct ("specializat"), the current prediction of my context model may benefit from these partial words in the historical utterances.

Figure 8.15 shows the third example which was selected from `eval2000` evaluation set. The blue word in the reference column is "joints" and baseline could not predict it ("john"), while my model could predict it well. As seen in the preceding utterances, the current prediction may benefit from the semantically related words in utterance history, i.e. doctor, muscle.

Figure 8.16 shows another example that was selected from the medical task. The blue word in the reference column is "uh naprosyn". The "naprosyn" is a medicine used to relieve pain from

Reference	Baseline	Context E2E
*i mean he was not proficient at it like doctor clausen is so he just put it in the muscle and figured it will i- it will get somewhere near the joints but it is not the same as when you put it in the **joints***	*i mean he wasn't professional at it like doctor classines so he just put it in the model and figured it will it it'll get somewhere near the georgia but it's not the same as when you put it in the **john***	*i mean he wasn't proficient at it like doctor closson is so he just put it in the muscle and figured it'll it it'll get somewhere near the joints but it is not the same as than when you put it in the **joint***
0.870	*0.858*	*0.862*

Figure 8.15: Examples of reference, the hypothesis from baseline, and my conversational End-to-End ASR selected from `eval2000` evaluation set.

Reference	Baseline	Context E2E
*all day had a fever then i don't think i had a fever by the time i came in on thursday but motrin sometimes **uh naprosyn** can mask*	*all the had a fever but i don't think i had a fever by the time i it came and on thursday but motrin sometimes **an approsyn** can mass*	*all the had a fever then i don't think i had a fever by the time i it came in on thursday but motrin sometimes **a naprosyn** can mass*
0.890	*0.839*	*0.859*

Figure 8.16: Examples of reference, the hypothesis from baseline, and my conversational End-to-End ASR selected from the Medical task.

various conditions such as headache, muscle aches, and dental pain. The "naprosyn" appeared 155 times in training transcriptions which has 16,423,024 words in total. Even though it is hard to model such rare, and medical terms, my context model could predict it well. As seen in the preceding utterances, the current prediction of my context model may benefit from the historical utterances (i.e. had a fever, and motrin) with the "world knowledge".

Reference	Baseline	Context E2E
*it's the uh it comes in a little tube it's like a cream like a white uh white cream. uh do you ever get improvement with the ultraviolet light at all uh not really like summer time the **psoriasis** doesn't do great if you're outside*	*it's the it comes in a little tube it's like a cream like a white uh. um do you ever get improvement with ultraviolet lights at all uh not really like summertime **with the rise it** doesn't do grade if you're outside*	*it's the it comes in a little tube it's like a cream like a white uh um do you ever get improvement with ultraviolet lights at all uh not really like summertime the **psoriasis** doesn't do great if you're outside*
0.902	*0.872*	*0.885*

Figure 8.17: Examples of reference, the hypothesis from baseline, and my conversational End-to-End ASR selected from the Medical task.

Figure 8.17 shows the example which was selected from the medical task. The blue word in the reference column is "psoriasis" which is a skin condition in which skin cells build up and

form scales and itchy, dry patches. The p of “psoriasis” is silence syllable and “psoriasis” only appeared 1,629 times in training transcriptions which has 16,423,024 words in total so that it is hard to model the word. The baseline could not predict it (“with the rise it”), while my context model could predict it well. As seen in the preceding utterances, the current prediction may benefit from the related words (“ultraviolet lights”).

Reference	*Baseline*	*Context E2E*
*better to help more more control okay because again i think your brain looks pretty .. they're more the same than they are different but i kind have the idea that more severe disease we should use one more ..once a week avonex now **plegridy** is what is kind of preferred*	*better to help them more control okay because again i think your your brain looks pretty .. they're more the same than there are different but i'm kind of out the idea that more severe disease you should use one more ..once a week avonex now **plaguery** is what is kind of preferred*	*better to help them more control okay because again i think your your brain looks pretty .. they're more the same than there are different but uh kind of at the idea that more severe disease you use one more ..once a week avonex now **plegridy** is what is kind of preferred*
0.880	*0.837*	*0.843*

Figure 8.18: Examples of reference, hypothesis from baseline, and my conversational End-to-End ASR selected from the Medical task.

As in Figure 8.18, the blue word in the reference column is “plegridy” which is a treatment name related to a central nervous system. The “plegridy” appeared 42 times in training transcriptions which has 16,423,024 words in total. Even though it is hard to model such extremely rare, and medical terms, my context model could predict it well. As seen in the preceding utterances, the current prediction may benefit from the related words (“brain”) in utterance history with “world knowledge”.

8.3 Summary

In this chapter, I provided various analytic methods to demonstrate the effectiveness of my conversational End-to-End ASR models, in addition to the improved WER. I performed the experiments and analyses on three different tasks: SWBD, Fisher, Medical tasks and I observed that my model performs well when the historical utterances and current, predicted utterance are similar and when the historical utterances have more informative rare words. I also found that my model

tends to attend a longer historical utterance. Although the improvement provided by my context model reduces with a large training dataset ($>$ 1,700 hours) due to the strength of the acoustic model from the large training dataset, it is still statistically significant and meaningful. In the next Chapter 9, I will conclude the results and discuss future work.

Chapter 9

Conclusions

In this chapter, I will conclude my thesis and discuss future directions of the conversational speech recognition systems.

9.1 Thesis Conclusions

In this thesis, I have shown that modeling entire conversations, rather than isolated utterances, in End-to-End manner helps to process a long conversation. To prove my thesis idea, I identified challenges that arise in modeling entire conversations and motivation of my thesis (in chapter 1). I first presented my joint CTC/Seq2Seq End-to-End ASR models that achieved better recognition accuracy and faster convergence than vanilla End-to-End ASR by jointly training the model with CTC and Seq2Seq objectives (in chapter 3). I then showed an efficient way to preserve long conversational contexts while training the model without having a GPU memory issue by extracting/detaching/caching conversational context embeddings on serialized utterance-based minibatches (in chapter 4). Next, I showed the effective way to integrate conversational context embeddings into End-to-End ASR by using the gating mechanism (in chapter 5). I also showed various methods to encode conversational context embeddings by using previous spoken utterances and augmenting with world knowledge using external linguistic resources (i.e. BERT) (in

chapter 6). I showed that my conversational End-to-End ASR models improve recognition accuracy on three different large-scale conversational speech recognition tasks, Switchboard (300 hours), Fisher (2,000 hours), and Medical conversation (1,700 hours) (in chapter 7). I also shared several in-depth analyses to demonstrate the effectiveness of my conversational End-to-End ASR models (in chapter 8).

To the best of my knowledge, this is the first work to model entire conversations, rather than isolated utterance, by combining acoustic and language information in End-to-End manner. My thesis shows that modeling entire conversations is feasible and useful for better processing of long conversations.

9.2 Future Work

In this final section, I highlight a few promising areas to advance this thesis work. I hope this thesis work can inspire researchers in the direction of conversational speech recognition and spoken language understanding and pioneer a new class of End-to-End learning systems of conversations.

9.2.1 Acoustic Conversational Context

My approach can be easily extended to use "acoustic" conversational context, such as emotions, speaking style, background noise, music, and other non-verbal cues, in addition to the current "linguistic" conversational context. As an input of my context encoder, we can use the previous encoder outputs or even raw acoustic features instead of using the previous decoder output tokens. The context encoder should be able to learn an acoustic context embedding as special intents which is beneficial to improve overall accuracy performance.

9.2.2 From Audio to Semantics

My approach can potentially be applied to any task, from audio to semantic understanding (i.e. spoken language understanding, audio/video summarization, and question answering) and optimize the model in an End-to-End manner. We should leverage on a long, acoustic and linguistic context in these tasks. Conventional spoken language understanding systems consist of two main components: an automatic speech recognition module and a natural language understanding module. Rather than optimizing these two components independently, we should optimize jointly so that the system can focus more on errors that matter for the understanding tasks, not errors of filler words. Evidence has recently been shown that optimizing directly for the semantic understanding task may yield better results [96, 97].

Bibliography

[1] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *Interspeech*, 2010. 1.1, 2.1, 2.3.1

[2] Tomas Mikolov and Geoffrey Zweig, "Context dependent recurrent neural network language model.," in *SLT*. IEEE, 2012. 1.1, 2.3.1, 6.2

[3] Tian Wang and Kyunghyun Cho, "Larger-context language modelling," in *ACL*, 2016. 1.1, 2.3.1

[4] Yangfeng Ji, Trevor Cohn, Lingpeng Kong, Chris Dyer, and Jacob Eisenstein, "Document context language models," in *ICLR workshop*, 2016. 1.1, 2.3.1

[5] Bing Liu and Ian Lane, "Dialog context language modeling with recurrent neural networks," in *ICASSP*. IEEE, 2017. 1.1, 2.3.1

[6] Wayne Xiong, Lingfeng Wu, Jun Zhang, and Andreas Stolcke, "Session-level language modeling for conversational speech," in *EMNLP*, 2018. 1.1

[7] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *ICML*, 2006. 1.1, 2.2, 2.2.1

[8] Alex Graves and Navdeep Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *ICML*, 2014. 1.1, 2.2, 2.2

[9] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan

Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al., "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014. 1.1, 2.2, 2.2, 2.2.1, 3.1, 4.2

[10] Yajie Miao, Mohammad Gowayyed, and Florian Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," in *ASRU*. IEEE, 2015. 1.1, 2.2, 2.2, 2.2.1, 3.1

[11] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015. 1.1, 3.1

[12] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, "Attention-based models for speech recognition," in *NeurIPS*, 2015. 1.1, 2.2, 2.2, 2.2.2, 2.2.2, 3.1, 4.3.2, 5.3.2, 7.2.3

[13] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *ICASSP*. IEEE, 2016. 1.1, 2.2, 2.2

[14] Shinji Watanabe, 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, "Espnet: End-to-end speech processing toolkit," in *Interspeech*, 2018. 1, 4.3.2, 4.3.2, 7.2

[15] Takaaki Hori, Shinji Watanabe, Yu Zhang, and William Chan, "Advances in joint ctc-attention based end-to-end speech recognition with a deep cnn encoder and rnn-lm," in *Interspeech*, 2017. 1, 3.2, 4.3.2, 5.3.2, 7.2.3

[16] Siddharth Dalmia, Abdelrahman Mohamed, Mike Lewis, Florian Metze, and Luke Zettlemoyer, "Enforcing encoder-decoder modularity in sequence-to-sequence models," *arXiv preprint arXiv:1911.03782*, 2019. 1

[17] Steven Davis and Paul Mermelstein, "Comparison of parametric representations for mono-

syllabic word recognition in continuously spoken sentences," in *ICASSP*. 1980, IEEE. 2.1

[18] Hynek Hermansky, "Perceptual linear predictive (plp) analysis of speech," *Acoustical Society of America*, 1990. 2.1

[19] Chanwoo Kim and Richard M Stern, "Power-normalized cepstral coefficients (pncc) for robust speech recognition," *TASLP*, 2016. 2.1

[20] Leonard E Baum and Ted Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The annals of mathematical statistics*, 1966. 2.1

[21] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine*, 2012. 2.1, 2.2

[22] Nelson Morgan and Herve Bourlard, "Continuous speech recognition using multilayer perceptrons with hidden markov models," in *ICASSP*. IEEE, 1990. 2.1

[23] George E Dahl, Dong Yu, Li Deng, and Alex Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *TASLP*, 2011. 2.1

[24] Navdeep Jaitly, Patrick Nguyen, Andrew Senior, and Vincent Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition," in *Interspeech*, 2012. 2.1

[25] Li Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Michael Seltzer, Geoff Zweig, Xiaodong He, Jason Williams, et al., "Recent advances in deep learning for speech research at microsoft," in *ICASSP*. IEEE, 2013. 2.1

[26] Frank Seide, Gang Li, and Dong Yu, "Conversational speech transcription using context-dependent deep neural networks.," in *Interspeech*, 2011. 2.1

[27] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, and Gerald Penn, "Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition,"

in *ICASSP*. IEEE, 2012. 2.1

[28] Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran, "Deep convolutional neural networks for lvcsr," in *ICASSP*. IEEE, 2013. 2.1

[29] Tara N Sainath, Brian Kingsbury, Abdel-rahman Mohamed, George E Dahl, George Saon, Hagen Soltau, Tomas Beran, Aleksandr Y Aravkin, and Bhuvana Ramabhadran, "Improvements to deep convolutional neural networks for lvcsr," in *ASRU*. IEEE, 2013. 2.1

[30] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, 1997. 2.1, 2.4, 3.3.2, 5.1

[31] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *ASRU*. IEEE, 2013. 2.1, 3.3.2

[32] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *ICASSP*. IEEE, 2013. 2.1

[33] Haşim Sak, Andrew Senior, and Françoise Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *arXiv preprint arXiv:1402.1128*, 2014. 2.1

[34] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *ICASSP*. IEEE, 2015. 2.1

[35] Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "End-to-end continuous speech recognition using attention-based recurrent NN: First results," *arXiv preprint arXiv:1412.1602*, 2014. 2.2, 2.2

[36] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio, "End-to-end attention-based large vocabulary speech recognition," in *ICASSP*. IEEE, 2016. 2.2, 2.2, 2.2.2, 2.2.2

[37] Liang Lu, Xingxing Zhang, and Steve Renals, "On training the recurrent neural network encoder-decoder for large vocabulary end-to-end speech recognition," in *ICASSP*. IEEE,

2016. 2.2

[38] Abdel-rahman Mohamed, George E Dahl, and Geoffrey Hinton, "Acoustic modeling using deep belief networks," *TASLP*, 2012. 2.2

[39] David M Blei, Andrew Y Ng, and Michael I Jordan, "Latent dirichlet allocation," *JMLR*, 2003. 2.3.1

[40] Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig, "The microsoft 2016 conversational speech recognition system," in *ICASSP*. IEEE, 2017. 2.3.1

[41] Golan Pundak, Tara N Sainath, Rohit Prabhavalkar, Anjuli Kannan, and Ding Zhao, "Deep context: end-to-end contextual speech recognition," in *SLT*. IEEE, 2018. 2.3.2

[42] Uri Alon, Golan Pundak, and Tara N Sainath, "Contextual speech recognition with difficult negative training examples," in *ICASSP*. IEEE, 2019. 2.3.2

[43] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014. 2.4, 5.1

[44] Pawel Swietojanski and Steve Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," in *SLT*. IEEE, 2014. 2.4

[45] John Arevalo, Thamar Solorio, Manuel Montes-y Gómez, and Fabio A González, "Gated multimodal units for information fusion," *arXiv preprint arXiv:1702.01992*, 2017. 2.4, 5.2.2

[46] Jamie Kiros, William Chan, and Geoffrey Hinton, "Illustrative language understanding: Large-scale visual grounding with image search," in *ACL*, 2018. 2.4, 5.2.2

[47] Suyoun Kim and Michael L Seltzer, "Towards language-universal end-to-end speech recognition," in *ICASSP*. IEEE, 2018. 2.4, 5.1, 5.2.2

[48] Suyoun Kim, Takaaki Hori, and Shinji Watanabe, "Joint ctc-attention based end-to-end

speech recognition using multi-task learning," in *ICASSP*. IEEE, 2017. 3, 4.3.2, 4.3.2, 5.3.2, 7.2, 7.2.3

[49] Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R Hershey, and Tomoki Hayashi, "Hybrid ctc/attention architecture for end-to-end speech recognition," *JSTSP*, 2017. 3, 4.3.2, 4.3.2, 5.3.2, 7.2, 7.2.3

[50] Linguistic Data Consortium, "CSR-II (wsj1) complete," *Linguistic Data Consortium*, vol. LDC94S13A, 1994. 3.3.1

[51] John Garofalo, David Graff, Doug Paul, and David Pallett, "CSR-I (wsj0) complete," *Linguistic Data Consortium*, vol. LDC93S6A, 2007. 3.3.1

[52] Emmanuel Vincent, Shinji Watanabe, Aditya Arie Nugraha, Jon Barker, and Ricard Marxer, "An analysis of environment, microphone and data simulation mismatches in robust speech recognition," *Computer Speech and Language*. 3.3.1

[53] Matthew D Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012. 3.3.2, 4.3.2, 7.3

[54] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, "On the difficulty of training recurrent neural networks," *arXiv preprint arXiv:1211.5063*, 2012. 3.3.2

[55] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le, "Sequence to sequence learning with neural networks," in *NeurIPS*, 2014. 3.3.2, 4.3.2, 7.3

[56] Preferred Networks, "Chainer," in *"http://chainer.org/"*. 3.3.2

[57] Jongseo Sohn, Nam Soo Kim, and Wonyong Sung, "A statistical model-based voice activity detection," *signal processing letters*, 1999. 4.1.2

[58] Yu Zhang, William Chan, and Navdeep Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in *ICASSP*. IEEE, 2017. 4.3.2, 5.3.2, 7.2.3

[59] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, "On the difficulty of training recurrent neural networks," in *ICML*, 2013. 4.3.2, 7.3

[60] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton, "Chainer: a next-generation open source framework for deep learning," in *NeurIPS workshop*, 2015. 4.3.2

[61] Thomas Zenkel, Ramon Sanabria, Florian Metze, Jan Niehues, Matthias Sperber, Sebastian Stüker, and Alex Waibel, "Comparison of decoding strategies for ctc acoustic models," in *Interspeech*, 2017. 4.2

[62] Geoffrey Zweig, Chengzhu Yu, Jasha Droppo, and Andreas Stolcke, "Advances in all-neural speech recognition," in *ICASSP*. IEEE, 2017. 4.2, 7.4, 7.4.1, 7.5, 7.4.1, 7.4.2

[63] Hagen Soltau, Hank Liao, and Hasim Sak, "Neural speech recognizer: Acoustic-to-word lstm model for large vocabulary speech recognition," in *Interspeech*, 2017. 5.3.1, 7.2.2

[64] Kartik Audhkhasi, Bhuvana Ramabhadran, George Saon, Michael Picheny, and David Nahamoo, "Direct acoustics-to-word models for english conversational speech recognition," in *Interspeech*, 2017. 5.3.1, 7.2.2

[65] Kartik Audhkhasi, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Michael Picheny, "Building competitive direct acoustics-to-word models for english conversational speech recognition," in *ICASSP*. IEEE, 2018. 5.3.1, 5.1, 5.3.3, 7.2.2, 7.4, 7.4.1, 7.5, 7.4.1

[66] Jinyu Li, Guoli Ye, Amit Das, Rui Zhao, and Yifan Gong, "Advancing acoustic-to-word ctc model," in *ICASSP*. IEEE, 2018. 5.3.1, 7.2.2

[67] Shruti Palaskar and Florian Metze, "Acoustic-to-word recognition with sequence-to-sequence models," in *SLT*. IEEE, 2018. 5.3.1, 7.2.2

[68] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *NeurIPS*, 2015. 6.1.3

[69] Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig, "Achieving human parity in conversational speech recognition," *arXiv preprint arXiv:1610.05256*, 2016. 6.2

[70] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Tor-

ralba, and Sanja Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *ICCV*, 2015. 6.2

[71] Jeffrey Pennington, Richard Socher, and Christopher Manning, "Glove: Global vectors for word representation," in *EMNLP*, 2014. 6.2

[72] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov, "Enriching word vectors with subword information," *TACL*, 2017. 6.2

[73] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, 2019. 6.2

[74] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer, "Deep contextualized word representations," in *NAACL*, 2018. 6.2

[75] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi, "Bidirectional attention flow for machine comprehension," in *CoRR*, 2017. 6.2

[76] Shuohang Wang and Jing Jiang, "Learning natural language inference with lstm," *arXiv preprint arXiv:1512.08849*, 2015. 6.3.2

[77] Shuohang Wang and Jing Jiang, "Machine comprehension using match-lstm and answer pointer," *arXiv preprint arXiv:1608.07905*, 2016. 6.3.2

[78] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly, "Pointer networks," in *NeurIPS*, 2015. 6.3.2

[79] John Godfrey and Edward Holliman, "Switchboard-1 release 2 ldc97s62," *Linguistic Data Consortium*, vol. LDC97S62, 1993. 7.1.1

[80] Christopher Cieri, David Miller, and Kevin Walker, "The fisher corpus: a resource for the next generations of speech-to-text.," in *LREC*, 2004. 7.1.2

[81] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differ-

entiation in pytorch," in *NeurIPS workshop*, 2017. 7.2

[82] Yajie Miao, Mohammad Gowayyed, Xingyu Na, Tom Ko, Florian Metze, and Alexander Waibel, "An empirical exploration of ctc acoustic models," in *ICASSP*. IEEE, 2016. 7.2.1

[83] Albert Zeyer, Kazuki Irie, Ralf Schlüter, and Hermann Ney, "Improved training of end-to-end attention models for speech recognition," in *Interspeech*, 2018. 7.4, 7.4.1

[84] Hossein Hadian, Daniel Povey, Hossein Sameti, Jan Trmal, and Sanjeev Khudanpur, "Improving lf-mmi using unconstrained supervisions for asr," in *SLT*. IEEE, 2018. 7.4, 7.5, 7.4.1

[85] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019. 7.4, 7.4.1

[86] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur, "Purely sequence-trained neural networks for asr based on lattice-free mmi.," in *Interspeech*, 2016. 7.4.1

[87] Ramon Sanabria and Florian Metze, "Hierarchical multitask learning with ctc," in *SLT*. IEEE, 2018. 7.4.1

[88] Chao Weng, Jia Cui, Guangsen Wang, Jun Wang, Chengzhu Yu, Dan Su, and Dong Yu, "Improving attention based sequence-to-sequence models for end-to-end english conversational speech recognition," in *Interspeech*, 2018. 7.4.1, 7.5, 7.4.1, 7.4.2

[89] Eric Battenberg, Jitong Chen, Rewon Child, Adam Coates, Yashesh Gaur Yi Li, Hairong Liu, Sanjeev Satheesh, Anuroop Sriram, and Zhenyao Zhu, "Exploring neural transducers for end-to-end speech recognition," in *ASRU*. IEEE, 2017. 7.5

[90] Wayne Xiong, Lingfeng Wu, Fil Alleva, Jasha Droppo, Xuedong Huang, and Andreas Stolcke, "The microsoft 2017 conversational speech recognition system," in *ICASSP*. IEEE, 2018. 7.4.2

[91] Maximilian Bisani and Hermann Ney, "Bootstrap estimates for confidence intervals in asr performance evaluation," in *ICASSP*. IEEE, 2004. 7.5, 7.5

[92] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., "The kaldi speech recognition toolkit," in *ASRU*. IEEE, 2011. 7.5

[93] Gerard Salton and Michael J McGill, "Introduction to modern information retrieval," 1986. 8.1.3

[94] Joeran Beel, Stefan Langer, Marcel Genzmehr, Bela Gipp, Corinna Breitinger, and Andreas Nürnberger, "Research paper recommender system evaluation: a quantitative literature survey," in *RepSys workshop*. ACM, 2013. 8.1.3

[95] Suyoun Kim, Siddharth Dalmia, and Florian Metze, "Cross-attention end-to-end asr for two-party conversations," in *Interspeech*, 2019. 8.1.4

[96] Parisa Haghani, Arun Narayanan, Michiel Bacchiani, Galen Chuang, Neeraj Gaur, Pedro Moreno, Rohit Prabhavalkar, Zhongdi Qu, and Austin Waters, "From audio to semantics: Approaches to end-to-end spoken language understanding," in *SLT*. IEEE, 2018. 9.2.2

[97] Dmitriy Serdyuk, Yongqiang Wang, Christian Fuegen, Anuj Kumar, Baiyang Liu, and Yoshua Bengio, "Towards end-to-end spoken language understanding," in *ICASSP*. IEEE, 2018. 9.2.2