End-to-End Speech Recognition on Conversations

Suyoun Kim

November, 2019

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in

Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Prof. Florian Metze (Carnegie Mellon University)
Prof. Richard M. Stern (Carnegie Mellon University)
Prof. Bhiksha Raj (Carnegie Mellon University)
Dr. Michael L. Seltzer (Facebook)
Prof. Shinji Watanabe (Johns Hopkins University)

Acknowledgments

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, Shinji, and Mike 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 senior 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, Jaejun, Jay-yoon, Eunjin, Euiwoong, Hyeju, Yongsuk, Jihoon, Joohyung, Jooli, Hyokyung, Jiyeon Lee, Philgoo, Jinhee. Specially, 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 previous 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 our approach. This thesis will provide insight into designing conversational speech recognition system and spoken language understanding systems, which are becoming increasingly important as voice-driven device interfaces become mainstream.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

As voice-driven interfaces to devices become mainstream, many real-world applications that can recognize and understand long conversations are becoming increasingly important. For example, spoken dialog systems, AI assistants, including teleconferencing, audio/video recorded meeting summarization and analysis, these applications are everywhere in our everyday life and complete their tasks from the user's voice.

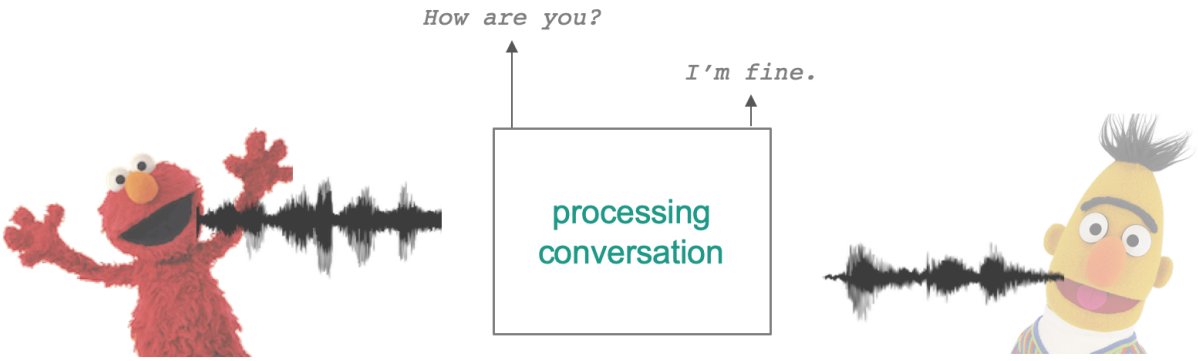


Figure 1.1: Processing of conversations of spoken dialog systems.

Conversational automatic speech recognition (ASR) that takes a spoken audio input and gen-

erates transcription is a core technique in such applications (in Figure 1.1). Comparing to a general speech recognition, conversational speech recognition techniques require to recognize not just a single isolated utterance, but multiple utterances in 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 lexical entrainment [2], the phenomenon of 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, 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 in conversational speech recognition tasks.

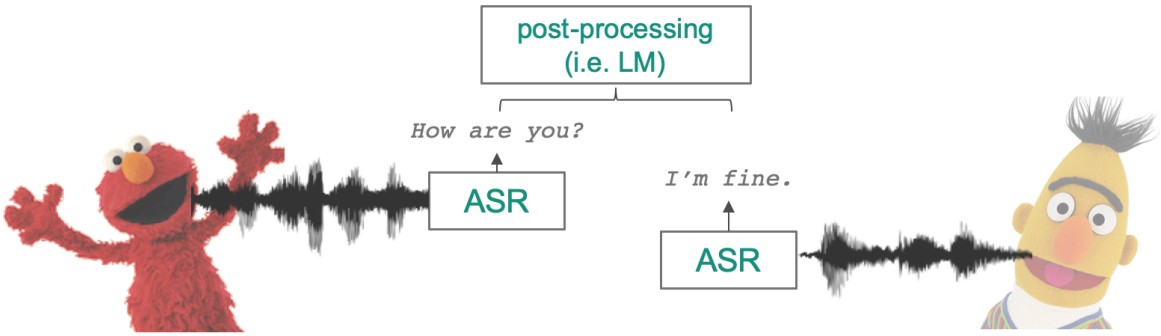


Figure 1.2: Current ASR solution in processing conversations.

However, current automatic speech recognition (ASR) solutions, even state-of-the-art systems, are modeling fragments, not entire conversations. As Figure 1.2 shows, in order to make building systems computationally feasible, long conversations are typically split into shorter blocks with utterance-level audio and systems are built at such individual, isolated utterance level. This modeling process makes the model lose important conversational context informa-

tion, which can help improve speech recognition performance.

There have been many studies that have attempted to use global, conversational level context information [3, 4, 5, 6, 7, 8] 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 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 [9, 10, 11, 12, 13, 14, 15] 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, we address following three main research questions:

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

The thesis also show accuracy improvements with three different large corpora, Switchboard (300 hours), Fisher (2,000 hours), and Medical conversation (3,700 hours), and present our analysis to demonstrate the effectiveness of our proposed framework.

1.3 Thesis Contributions

To the best of our knowledge, this is the first work to model entire conversations in an end-to-end manner.

The technical contributions of this thesis are the followings:

1. **An efficient way to preserve long conversational contexts by creating a context encoder that maps spoken utterance histories to a single context vector** - We serialize the training utterances based on their onset time, then shuffle them at dialog-level. We extract/detach/cache context embeddings for the computational efficiency and avoiding GPU memory issues. This is similar to truncated backpropagation through time (BPTT).
2. **An effective way to integrate conversational contexts into end-to-end models using a gating mechanism** - We also create a gated contextual decoder that can recognize the current utterance conditioning on the context information. We also introduce a contextual gating mechanism to integrate multiple types of embeddings: word, speech, and conversational context embeddings effectively. Rather than simply concatenating three embed-

dings, gating mechanism can decide how to weigh different embeddings and can shape information flow using multiplicative interactions.

3. **Various methods to encode conversational contexts by using previous spoken utterances and augmenting with world knowledge using external linguistic resources** - We create a conversational context encoder to map the multiple preceding utterance histories into a fixed-length context vector. We perform an extensive analysis of ways to represent the conversational-context in terms of the number of utterance history, and sampling strategy considering to use the generated sentences or the true preceding utterance. In addition, we exploit the external word (fastText) and/or sentence embeddings (BERT) for augmenting *world knowledge* and representing conversational context better. We also propose a speaker-specific cross-attention 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.

4. **An analysis methods to demonstrate the effectiveness of conversational context** - In addition to simply comparing the Word Error Rate (WER) with standard end-to-end speech recognition systems, we also compare the performance of our model with the context vector generated from *Oracle* utterances and *Random* utterances. This method can verify that our model's benefit is not come from randomness or regularization. We also propose a scoring function that measure the average distance of all possible pairs of adjacent utterances in the evaluation sets. We then compare the distance score of true transcription, hypothesis of baseline model, and hypothesis of our conversational model to show that our model can preserve the "conversational consistency".

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. We will review the strengths and weaknesses of two approaches. This will give motivation of our proposed end-to-end speech recognition approach, joint CTC/Seq2Seq model in Chapter 3. We show our model provides fast convergence as well as improved accuracy than previous end-to-end speech recognition system. In Chapter 4, we present our proposed conversational end-to-end automatic speech recognition (ASR) systems that model entire conversations. We 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, we 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). We 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, we show a series of experiments on three different large corpora, Switchboard (300 hours), Fisher (2,000 hours), and Medical conversation (3,700 hours), and present our analysis to demonstrate the effectiveness of our proposed framework. We will finally close up with a conclusion and future work in Chapter 8.

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 my 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). This reviews motivate us for proposing our joint CTC/Seq2Seq models in Chapter 3. Then, we review several literature 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 our 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.

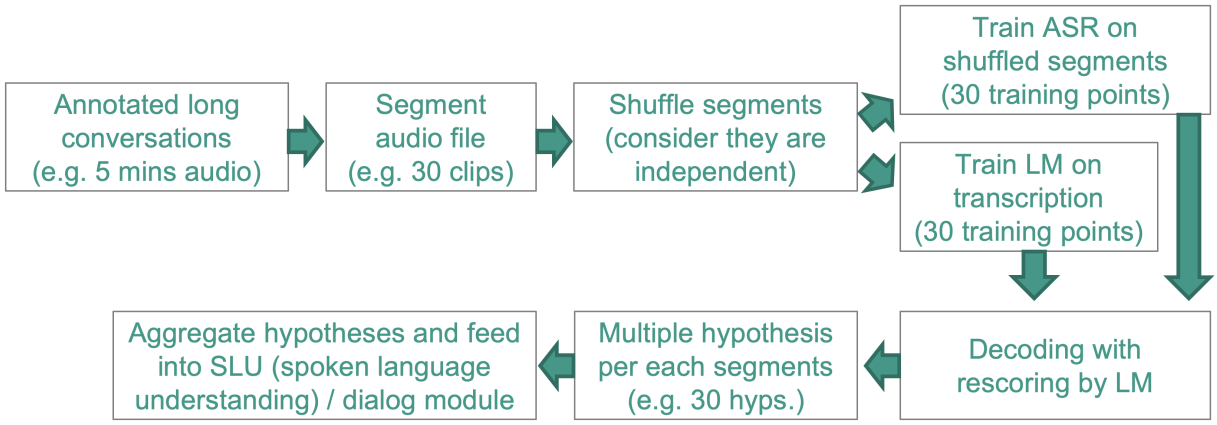


Figure 2.1: Overall system building process of typical conversational ASR.

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 each other. This segmentation/shuffling procedure makes training the speech recognition models computationally feasible. We will show that training the speech recognition models on an entire conversation is computationally infeasible by conducting 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) [16], Perceptual Linear Prediction (PLP) [17], and Power-normalized Cepstral Coefficients (PNCC) [18] by a signal processing frontend.

Modeling

Figure 2.2 shows the overall architecture of traditional ASR models.

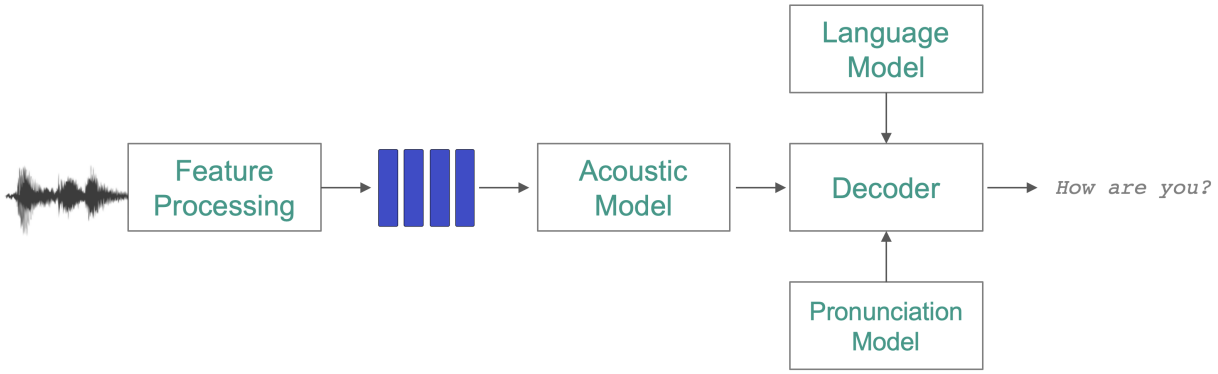


Figure 2.2: The overall architecture of traditional ASR.

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 sequence of acoustic feature of n-th utterance. Let $w^n = (w_1, ..., w_U)$ is U-length of 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) [19]. Deep Neural Network (DNN) [20, 21, 22, 23, 24], Convolutional Neural Network (CNN) [25, 26, 27, 28], or Long Short-Term Memory (LSTM) which is variants of Recurrent Neural Network (RNN) [29, 30, 31, 32, 33] 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 models 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 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 enable to find the word from acoustics fast by constraining 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 [3]. 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, 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 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. These post-processing performed by separately trained models and then user's specific request finally is completed in such subsequent modules such as language model.

However, with such 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, and lexical entrainment 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 provides better performance and fast convergence.

End-to-end speech recognition is a recent proposed approach that directly transcribes speech to text without requiring predefined alignment between acoustic frames and characters [10, 11, 12, 14, 15, 34, 35, 36]. As we described in 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 [20, 37]. 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) [9, 10, 11, 12] and Attention-based Encoder-Decoder

[14, 15, 34, 35]. Both methods address the problem of variable-length input and output sequences.

2.2.1 Connectionist Temporal Classification (CTC)

The key idea of CTC [9] 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 [11, 12].

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 [35]. 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 spe-

cial 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 [14]. 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 [14], 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 [14, 35] 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 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 [3] 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 [4, 5, 6, 7]. Mikolov et al. proposed a context-dependent RNN language model [4] using a context vector which is produced by applying latent Dirichelt allocation [38] on the preceding text. Wang et al. pro-

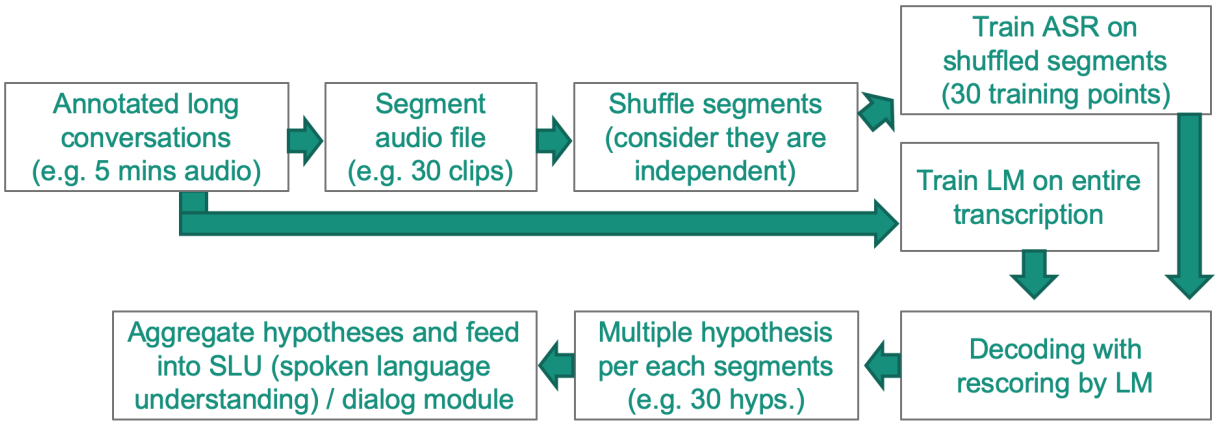


Figure 2.4: Context modeling in language models in conversational ASR pipelines.

posed using a "bag-of-words" to represent the context vector [5], and Ji et al. proposed using the last RNN hidden states from the previous sentence to represent the context vector [6]. Liu et al. proposed using an external RNN to model dialog context between speakers [7]. 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 [39] attempted to integrated such 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 to incorporate the context information within a sequence-to-sequence based end-to-end speech recognition models [40, 41]. 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 a conversational context information for processing a long conversation, their methods relies 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 [29, 42].

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) [43] is another example that is using 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 [44, 45, 46]. 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, [44, 45] attempted to integrate word representation and visual representation by using gating mechanism on over the two different types of representation. [46] 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 gating mechanism conditioned on the language identity in a language-specific way.

Chapter 3

Multi-task learning of CTC/Seq2Seq

In this chapter, we first describe the complementary characteristics of two main end-to-end approaches: CTC and Seq2Seq models (in Section **??**). We then propose a joint CTC/Seq2Seq end-to-end ASR framework to overcome the shortcomings of existing two main approaches CTC and Seq2Seq and to improve convergence speed as well as accuracy performance [47, 48]. We will show

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 a 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 a 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 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.

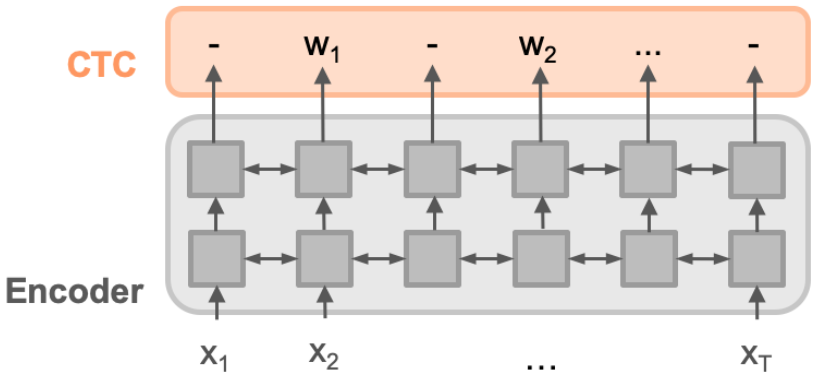


Figure 3.1: Previous CTC based end-to-end framework

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 [11, 12].

On the other hand, Seq2Seq based end-to-end ASR framework generates the output sequence conditioning on a attended h which is 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 well, the biggest strength of SeqSeq based end-to-end ASR framework is the ability of learning 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 on 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 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 [14**?**] 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 from long character

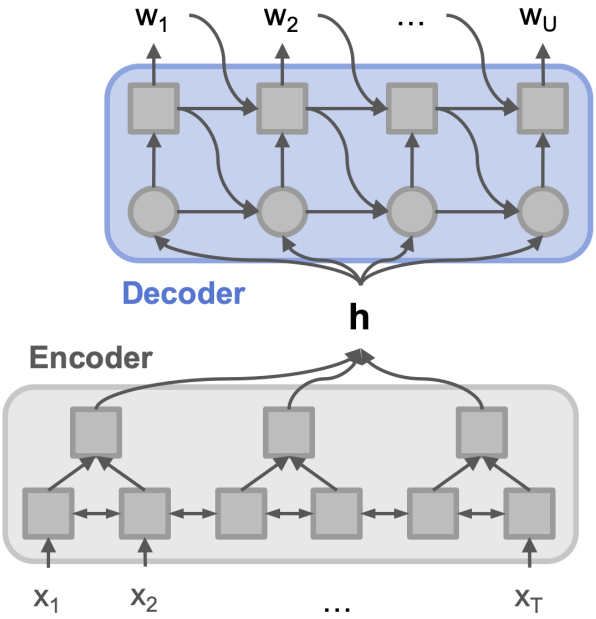


Figure 3.2: Previous Sequence-to-Sequence based end-to-end framework

sequences.

3.2 Joint learning of CTC/Seq2Seq

The main idea of our joint CTC/Seq2Seq ASR model is to use both CTC and Seq2Seq objective function within a multi-task learning framework (MTL). We 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 our joint CTC/Seq2Seq ASR framework. Our 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 our framework is more robust in acquiring appropriate alignments in noisy conditions.

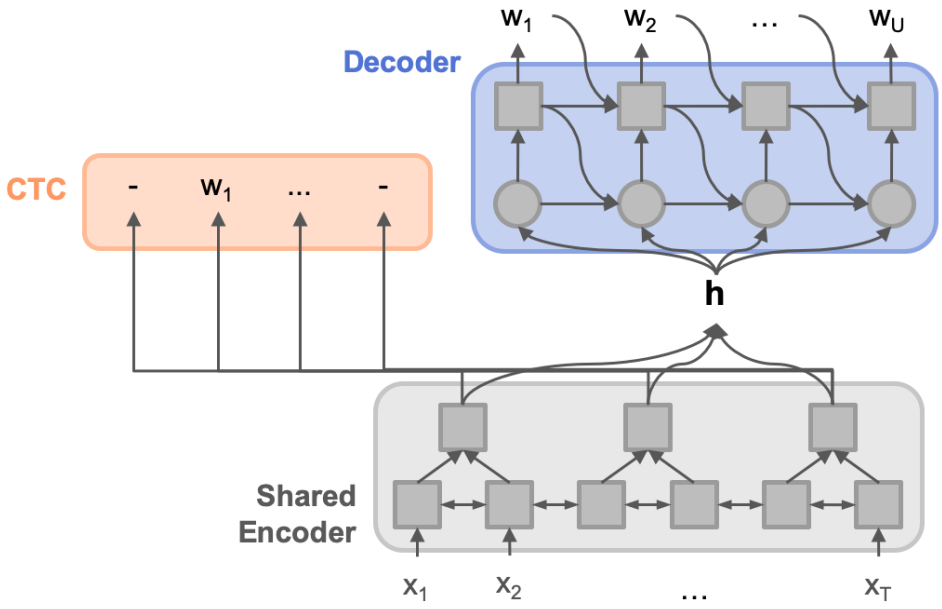


Figure 3.3: Our proposed Joint CTC/Seq2Seq based end-to-end framework

Another advantage of using CTC as an auxiliary task is that the network is learned quickly. In our 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 [49]:

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

3.3 Experiments

3.3.1 Datasets

WSJ and CHiME-4

We performed three sets of 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, we used 40 mel-scale filterbank coefficients, with their first and second order temporal derivatives to obtain a total 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 our experiments we do not use any language model and lexicon information. For Seq2Seq model, we 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

Our model used 4 layers of 320 Bidirectional Long Short-Term Memory Networks(BLSTM) [29, 30] in the encoder, and 1 layer of 320 LSTM in 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. We 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 our MTL, we tested three different task weights, λ: 0.2, 0.5, and 0.8.

For decoding, we used beam search algorithm similar to [55] with the beam size 20 to minimize the cost. We adjusted the cost by adding length penalty, length(hyp) $* 0.3$ for CHiME-4 and length(hyp) $* 0.1$ for WSJ experiments. Note that we do not use any lexicon or language model. Our framework for this 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 our 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 our 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 our joint CTC/Seq2Seq end-to-end model significantly outperformed over both CTC or Seq2Seq only models. Our 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 our proposed joint CTC/Seq2Seq end-to-end model on the noisy dataset CHiME-4. Our 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 our 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 our joint CTC/Seq2Seq increased in noisy condition, 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 our framework outperformed over both CTC and Seq2Seq model even in 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, our 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 our proposed joint CTC/Seq2Seq end-to-end model with (λ = 0.2, 0.5, 0.8) over training epoch. The character accuracy on validation set of CHiME-4 is calculated by edit distance. Note that the accuracy of Seq2Seq and our proposed joint CTC/Seq2Seq end-to-end model were obtained with given gold standard history. As shown in Figure 3.4, the CTC only model converged the

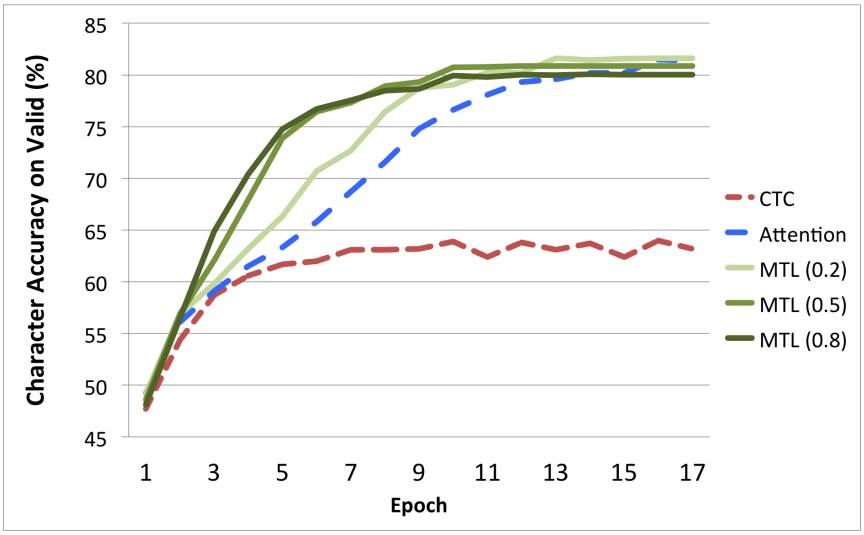


Figure 3.4: Comparison of learning curve: CTC, Seq2Seq, and our Joint CTC/Seq2Seq.

fast however its character accuracy is worse than the other models, Seq2Seq only, and our our proposed joint CTC/Seq2Seq end-to-end model as we expected. We observed that Seq2Seq only model converged the slowest even though the character accuracy is similar to our proposed joint CTC/Seq2Seq end-to-end models. We also observed that when we use large λ giving more weight on CTC loss, the network learns fast and converges early. This results demonstrates the effectiveness of using CTC as an additional objective function to boost the convergence speed as we 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, we also visualize the learned attention alignments between input and output at different training epoch stages to analyze. From this analysis, we demonstrate why our 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, we only show Seq2Seq models and our 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.

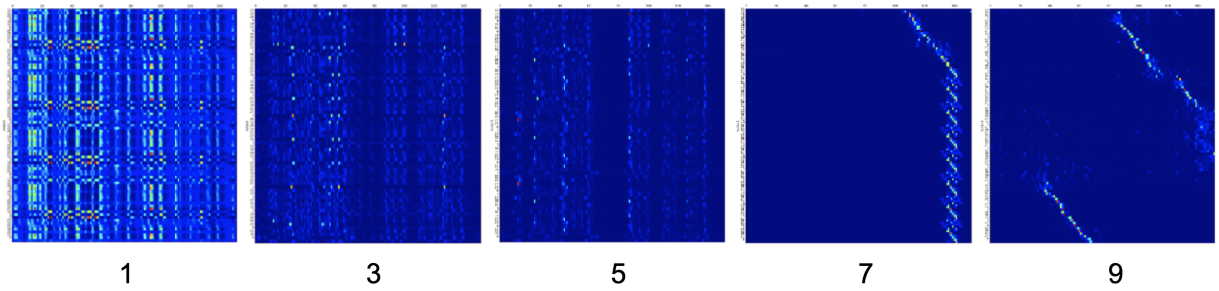


Figure 3.5: Visualization of the alignment between input frames and output characters of Seq2Seq only model.

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

Figure 3.6 shows the attention alignments between characters (y-axis) and acoustic frames (x-axis) of our 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.

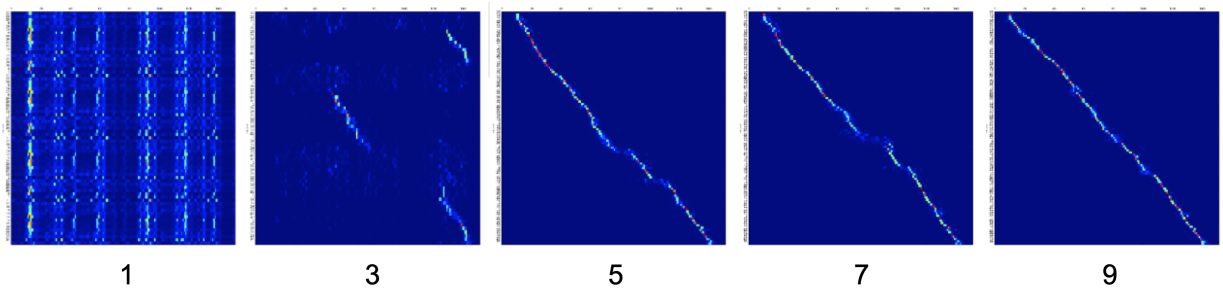


Figure 3.6: Visualization of the alignment between input frames and output characters of our proposed joint CTC/Seq2Seq end-to-end model.

We observed that our 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 learned the desired alignment. From this results we 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. Our 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. Our method has outperformed both CTC and an Seq2Seq model on a speech recognition task in real-world noisy condition as well as clean condition. In next chapters 4 6, we will show conversational end-to-end ASR based on 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, we present a novel conversational end-to-end ASR system that process entire conversations with a technique to preserve long conversational contexts, and present our context-aware end-to-end ASR models.

4.1 Preserving Conversational Context

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

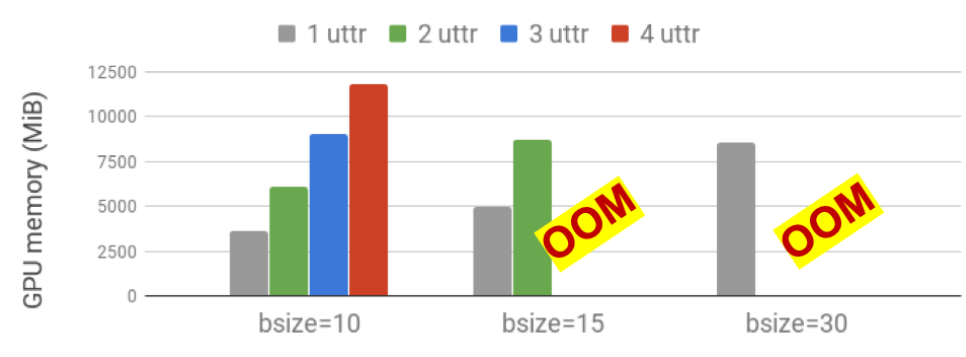


Figure 4.1: BPTT on entire conversation is computationally infeasible.

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

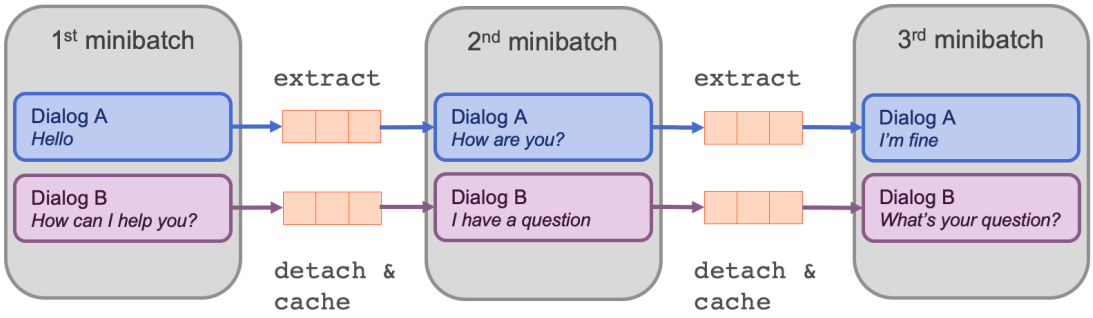


Figure 4.2: Extract/Detach/Cache context embedding on serialized utterances based on their onset time in dialog.

In order to keep track the context embeddings over each minibatch iteration during the training procedure, we create minibatches with serialized utterances based on their start time in dialog. We 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 we create minibatches to keep track the context embeddings. 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 we shown in previous section, the backpropagation through time (BPTT) on the entire conversation is computationally infeasible. As our solution, we propose to `extract` the context information in a single, fixed-dimensional vector. We 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 we focus on linguistic conversational context by using previous spoken utterances. We 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 we `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, we additionally store the dialog identities of the minibatch so that we can pass the correct context embedding to the next minibatch properly.

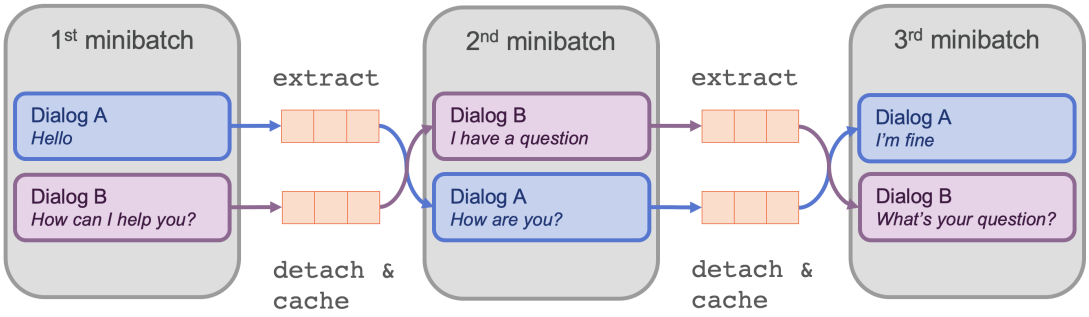


Figure 4.3: In case of the order of utterances are changed within minibatch, we additionally keep track the dialog ID.

Efficient Minibatch Creation

After the minibatch computations of a dialog finished, we 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, we 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 our 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 our approach, not for each iteration, but for a series of iteration (at dialog level), 1) we 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

```
procedure CREATEUTTRMINIBATCH([dialog-group], m, k)
    shuffle [dialog-groups]
    if multi-GPU then:
        for dgroup in [dialog-groups]:
            for dbatch in dgroup:
                split dialogs into k
                distribute one (m-dialogs) to each GPU
                [uttr-batch] += unpack(dbatch)
    else:
        for dbatch in [dialog-groups]:
            [uttr-batch] += unpack(dbatch)

    end if
    return [uttr-batch]
```

4.2 Vanilla Context-Aware End-to-End ASR

In this section, we describe our 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. We formulate our 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.

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

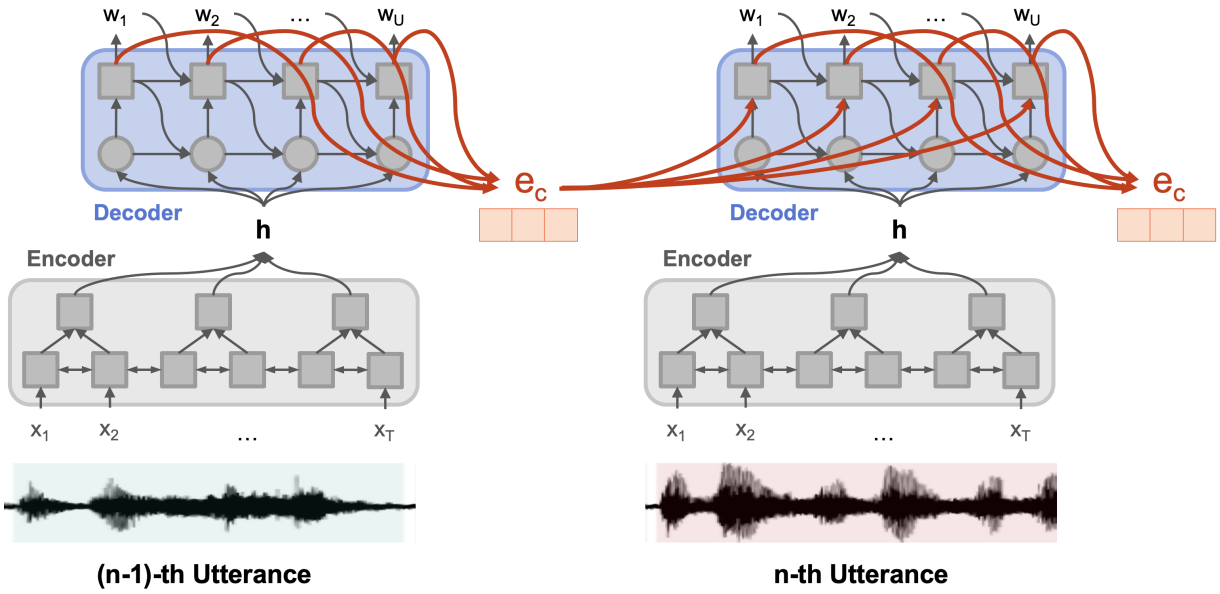


Figure 4.4: Overall architecture of our conversational end-to-end ASR models.

Figure 4.4 shows the standard end-to-end ASR that we 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 our vanilla conversational end-to-end model, we simply use a single preceding

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

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

Our 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)

We investigated the accuracy performance of our proposed conversational end-to-ennd ASR models on the Switchboard LDC corpus (97S62) which has a 300 hours training set. Note that in this experiment we did not use the Fisher dataset. We 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 we show separate results for the Callhome English (CH) and Switchboard (SWB) evaluation sets. We 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.

We 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. We 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. We 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, we used joint CTC/Seq2Seq end-to-end speech recognition architecture [47, 48] with ESPnet toolkit [58] as we described in previous Chapter 3. We used a CNN-BLSTM encoder as suggested in [49, 59]. We followed the same six-layer CNN architecture as the prior study, except we used one input channel instead of three, since we 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. We used a location-based attention mechanism [14], where 10 centered convolution filters of width 100 were used to extract the convolutional features.

The decoder network of both our proposed models and the baseline models was a one-layer LSTM with 300 cells. Our 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. We 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 we did not use any extra text data other than the training transcription.

The AdaDelta algorithm [53] with gradient clipping [60] was used for optimization. We used $\lambda = 0.5$ for joint CTC/Attention training. We bootstrap the training our proposed dialog-context aware end-to-end models from the baseline end-to-end models. For decoding of the models, we used joint decoder which combines the output label scores from the AttentionDecoder, CTC, and Char-RNNLM by using shallow fusion [49]:

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

The scaling factor of CTC, and RNNLM scores were $\alpha = 0.3$, and $\beta = 0.3$, respectively. We used a beam search algorithm similar to [55] with the beam size 20 to reduce the computation cost. We 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 [61], and ESPnet toolkit [47, 48, 58].

4.3.3 Results

We evaluated both the end-to-end speech recognition model which was built on sentence level data (***sentence-level end2end***) and our 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 our baseline, our 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 our conversational end-to-end ASR models (vanilla) on the Switchboard dataset.

Models	CH WER (%)	SWB WER (%)
sentence-level end2end		
Seq2Seq A2C [62]	40.6	28.1
CTC A2C [11]	31.8	20.0
CTC A2C [63]	32.1	19.8
sentence-level end2end		
Our baseline (CTC/Seq2Seq)	34.4	19.0
dialog-context aware end2end		
Our method (a)	34.1	**18.2**
Our method (b)	**33.2**	18.6

300 hours Switchboard training data.

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

Table 4.3: Substitution rate (Sub), Deletion rate (Del), and Insertion rate (Ins) for the baseline and our 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

We also analyze the WER results by decomposing it into the insertion, deletion, and substi-

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

In addition to the WER results, we 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, our 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 our 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, our conversational end-to-end ASR model can predict correctly. In the previous utterance exists semantically related words were appeared, and our 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 our 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, we described our 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. We also described our

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. Our conversational end-to-end ASR model was shown to outperform previous end-to-end models trained on utterance-level data, even when we 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

Our 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 our end-to-end ASR architecture (in Chapter 3). In this chapter, we explore the way to integrate conversational context embeddings e_c more effectively by using gating mechanism. In Section 5.1, we show the example of usage of gating mechanism that was motivated us to use in our conversational end-to-end ASR models. In Section 5.2, we 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 [29, 42]. In this section, we show our language-specific gating mechanism [46] 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 [46] 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.

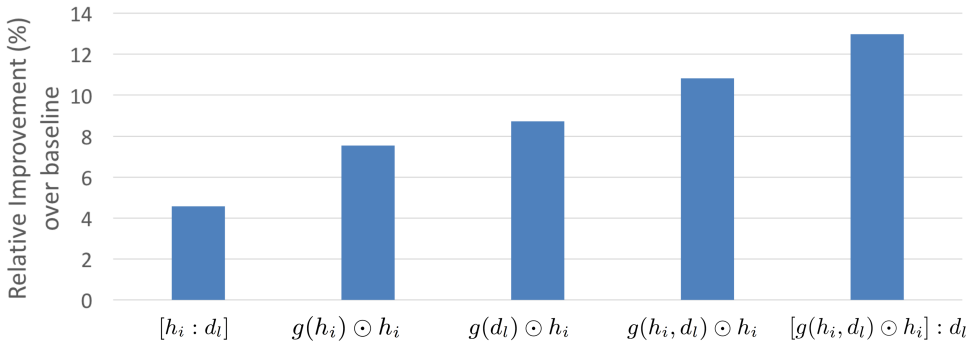


Figure 5.1: Comparison of the WER performance improvement (relative) over different types of gating mechanism.

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. We will describe the details in the next Section 5.2.

5.2 Conversational Context Fusion in Decoder

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

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

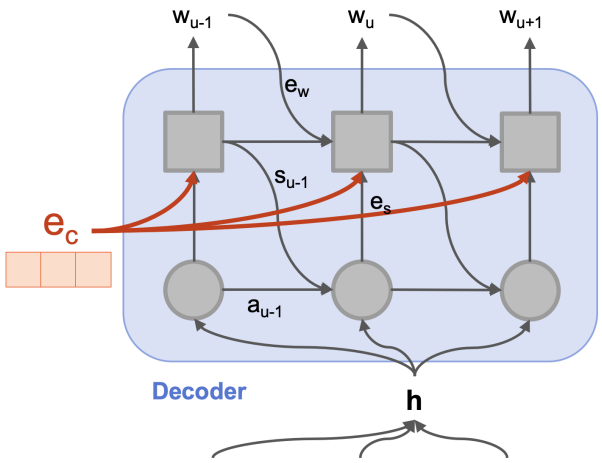


Figure 5.2: Decoder network of end-to-end ASR that takes conversational context embeddings.

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 our conversational context embedding, we 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. We extend this decoder, since we now have "context embedding" e_c from previous spoken utterances (in Equation 5.4. So, our 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.

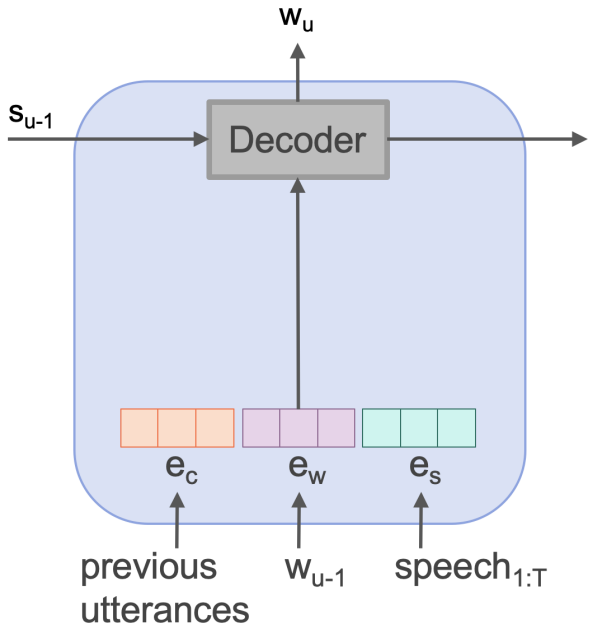


Figure 5.3: Concatenation of context/word/speech embeddings

5.2.2 Gated Contextual Decoder

Rather than simple concatenation methods as discussed in previous section, we 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 [44, 45], and for learning different langauges in speech recognition task [46]. The gating mechanism decides how to weigh different types of embeddings, and we can shape information flow using multiplicative interactions. We 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 our proposed contextual gating mechanism.

5.3 Experiments

In this section, we evaluate our 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.

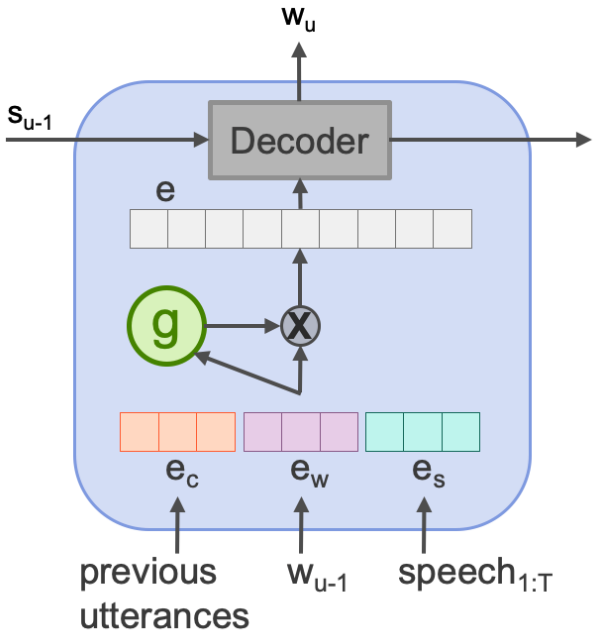


Figure 5.4: Our contextual gating mechanism in decoder network of end-to-end ASR to integrate context/word/speech embeddings

5.3.1 Output Units

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

Word+Character units

We first explore the use of word and character for the output units of our 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 [64, 65, 66, 67, 68]. 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, we first restrict the vocabulary

to 10k frequently occurring words. We then additionally use character units and start-of-OOV (sunk), end-of-OOV (eunk) tokens to make our 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 our 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, we used joint CTC/Seq2Seq end-to-end ASR [47, 48] as we proposed in Chapter 3. As suggested in [49, 59], 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, we used a location-based method [14]. For the decoder network, we 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 7.2 summarizes our 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 our proposed conversational end-to-end ASR with gated contextual decoder on SWBD task.

Model		#params.	LM	SWB	CH
Other A2W E2E systems					
CTC [66]	Word output, phone pretrain.	n/a	✗	14.6	23.6
Our systems					
our baseline	Char output	23M	✗	19.0	34.4
our baseline	`WordChar`	32M	✗	17.9	30.6
Our conv.E2E					
vanilla	`WordChar`	33M	✗	17.3	30.3
vanilla + gate	`WordChar`	35M	✗	17.2	29.8

We first present the other systems, our baseline system that use the word level output units, and then our conversational systems. Unlike other systems, note that our system does not use any phonetic information [66].

We first observed that our new baseline with our 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 we observed that our vanilla conversational end-to-end ASR model obatained 3.3% and 1.0% further, relative WER improvement over our new improved baseline. This result again confirmed the finding that incorporating context embedding generated from previous utterance helps the current utterance prediction.

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

shows the improved WER results. As shown in this Table 5.1, our 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, we described our proposed effective way to integrate the context embeddings and our proposed `WordChar` output sets. By using our proposed gated contextual decoder and our proposed `WordChar` output sets, we 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, we 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 our 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, we create an additional sub-network in our end-to-end ASR that generates conversational context embedding from previous spoken utterances.

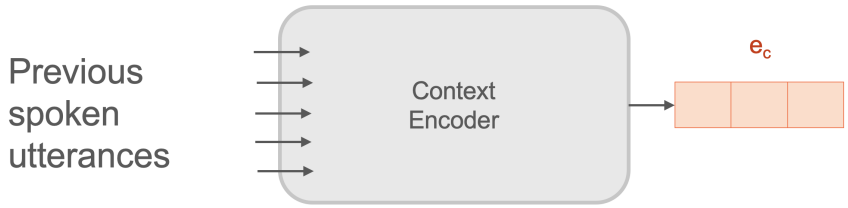


Figure 6.1: Our context encoder generates conversational context embedding from previous spoken utterances

We 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, we 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

We 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 our context encoder may take multiple past utterances, not just a single preceding utterance, we 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.

We first investigate the effect of the number of utterance history being encoded. We tried

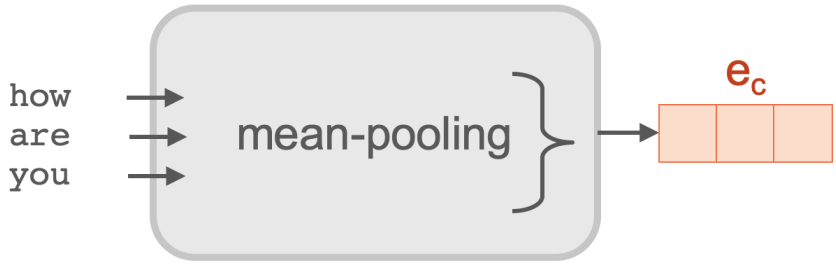


Figure 6.2: Our vanilla context encoder with word-level input and mean-pooling method

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.

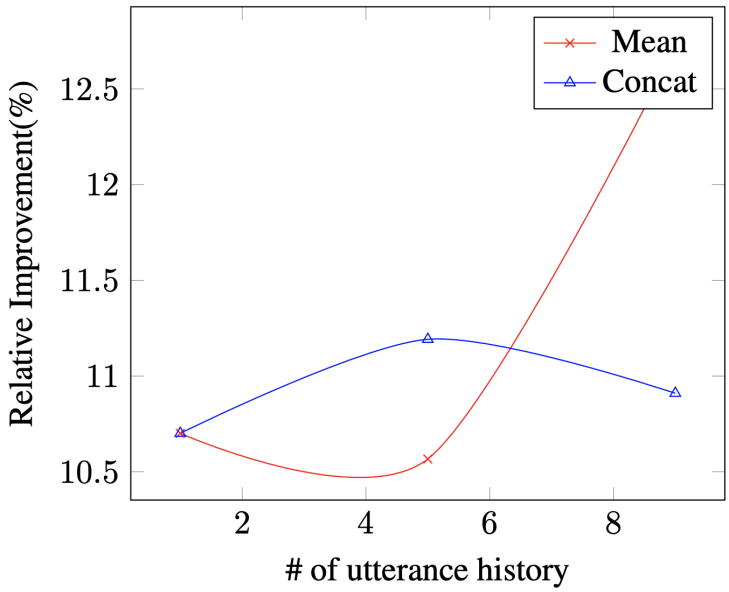


Figure 6.3: The relative WER improvement on validation set with different number of utterance history.

First we observed that more history helps to improve performance generally. However, as we increase the number of history, we 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 our experiments, we therefore decided not to use concatenation method to integrate the multiple histories since it is not scalable.

6.1.3 Sampling Strategy

Similar to sampling strategy [69] which is widely used in Seq2Seq model for generating previous word embedding, we 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.

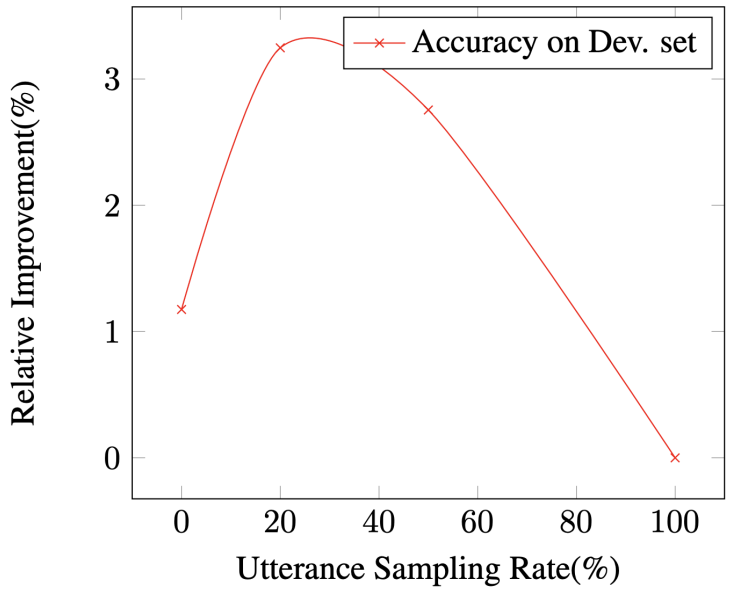


Figure 6.4: The relative WER improvement on validation set with sampling strategy with various ratio $[0.0, 0.2, 0.5, 1.0]$ to choose model outputs.

We 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.. We observed that a sampling rate of 0.1 or 0.2 performed best slightly empirically. In the rest of our experiments, we 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. In order to generate rich context representation the massive amount of training data requires. However, the annotated conversational speech corpus is more expensive to obtain, in contrast with the textual corpus, we typically use thousands of hours annotated corpus even in the industry [70]. For example, in case of the text corpus which is widely used in building language model, BookCorpus [71] 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, we propose to use external word or sentence embeddings that are pre-trained on large textual corpora to augment "world knowledge". As the input of our context encoder described in conversational end-to-end speech recognition framework (in previous sections), we 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 [4], GloVe [72], fastText [73], 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 [74, 75, 76], sentence embeddings, such as deep contextualized word representations [75], Pre-training of Deep Bidirectional Transformers for Language Understanding (BERT) [74], 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

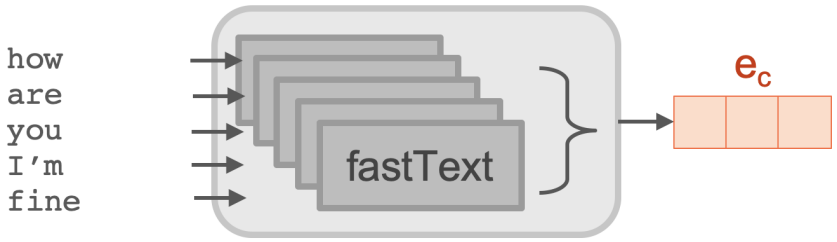


Figure 6.5: Our context encoder with external word embeddings (fastText)

We 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: Our 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 our 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, we 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 conversations 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 our proposed model is described in Figure 6.7. Specifically, our model works as follows.

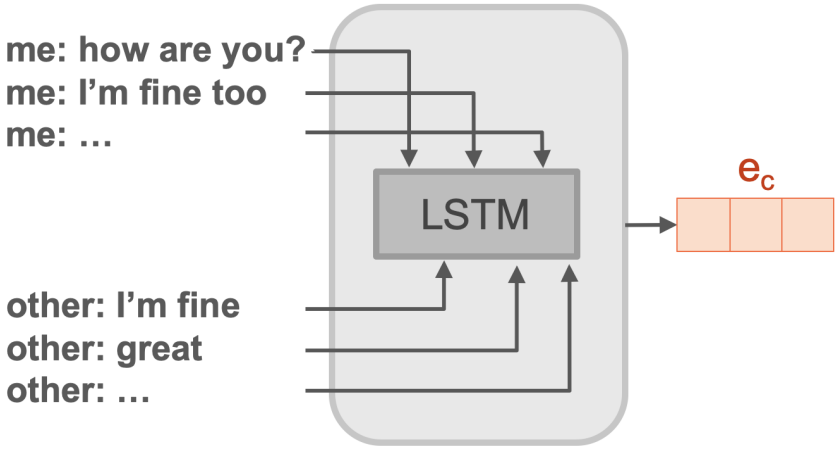


Figure 6.7: Our context encoder with LSTM and Attention mechanism for learning the interaction of two-speaker conversations

We create a queue for each speaker to store the history of utterance embeddings so that the utterance embeddings can be stored separately. We 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, we swap the queues properly. We 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, $\texttt{att\_e}^A_k$, is generated as follows:

$$\mathbf{G}^A_k = \texttt{tanh}(\mathbf{W} e^A_{k-N:k-1} + \mathbf{b}) \tag{6.1}$$

$$\alpha^A_k = \texttt{softmax}(\mathbf{w}^T \mathbf{G}^A_k + \mathbf{b}) \tag{6.2}$$

$$\texttt{att\_e}^A_k = \sum_N a^A_k \odot e^A_{k-N:k-1} \tag{6.3}$$

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) [77, 78], 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 [79] 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 our 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:

$$\mathbf{G}_{k_i}^A = \texttt{tanh}(\mathbf{W}e_{k-N:k-1}^B + \mathbf{V}e_i^A + \mathbf{U}\mathbf{h}_{i-1}^A + \mathbf{b})$$
$$\alpha_{k_i}^A = \texttt{softmax}(\mathbf{w}^T\mathbf{G}_{k_i}^A + b)$$

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.

$$\mathbf{z}_i^A = [e_i^A, e_{k-N:k-1}^B \odot \alpha_{k_i}^A]$$
$$\mathbf{h}_i^A = \texttt{matchLSTM}(\mathbf{z}_i^A, \mathbf{h}_{i-1}^A)$$

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

$$\texttt{att\_e}_k^A = \mathbf{h}_{k-1}^A$$

6.4 Training of Context Encoder

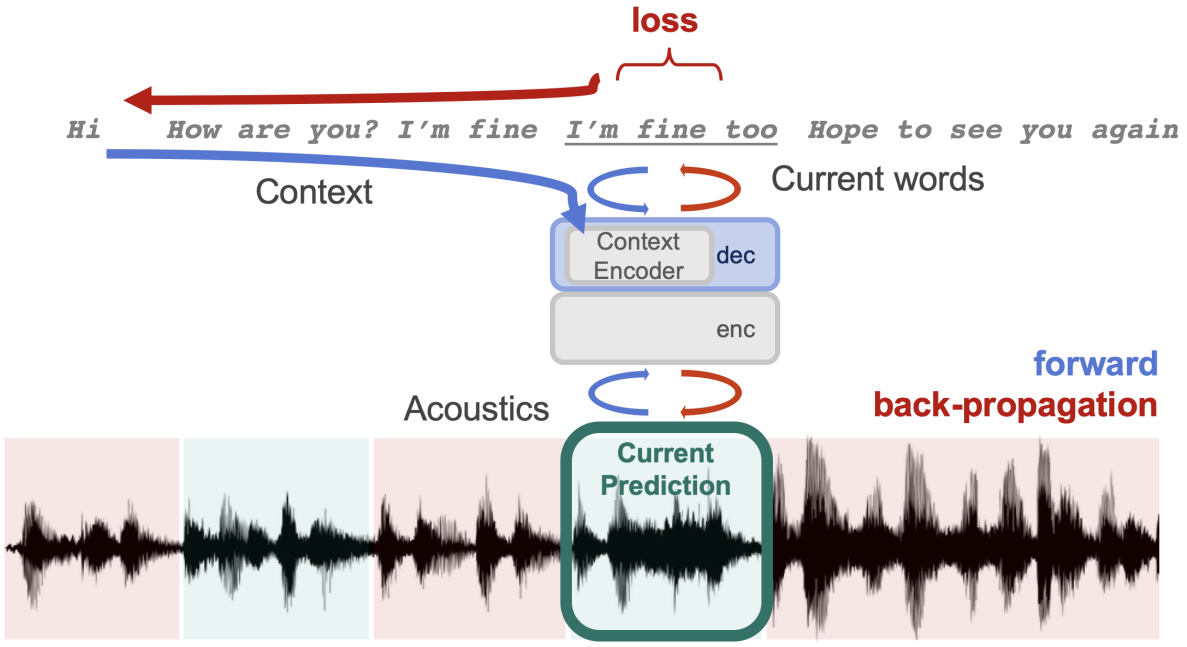


Figure 6.8: Our context encoder is designed to be trained over all (or window) of past utterances

Our 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 our context

encoder. The acoustics and words of current utterance, and contexts from past utterances are forwarded into our 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 8, as future work.

6.5 Summary

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

Chapter 7

A Large Scale Conversational Speech Recognition Tasks

7.1 Datasets

To evaluate our proposed conversational end-to-end ASR, we use three different conversational datasets: Switchboard (300h), Fisher (2,000h), and Presidio (3,700h). In Table 7.1, we show the number of dialogs and the average number of utterances per a single dialog.

Table 7.1: Experimental dataset description: SWBD, Fisher, and Medical datasets.

	training		validation	evaluation	
	SWBD	**Fisher**	**SWBD**	**SWBD**	**CallHm**
dialogs	2,402	11,692	34	20	20
utters./dialog	80	160	118	92	131
		Medical	**Medical**		**Medical**
dialogs		25,500	45		100
utters./dialog		155	149		151

7.1.1 Switchboard

We first use the Switchboard (SWBD) LDC corpus (97S62) task. We 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. We evaluate the model performance on the HUB5 Eval2000 which consists of the Callhome English (CH) and Switchboard (SWBD) (LDC2002S09, LDC2002T43).

7.1.2 Fisher

We also use the Fisher dataset, which consists around 2,000 hours of training data. The Fisher data includes 285 hours of SWBD training set as well. We evaluate the model on the same evaluation sets as the Switchboard task: the Callhome (CH) and Switchboard (SWBD).

7.1.3 Medical Conversation

In addition to the above well-known speech corpora, we use 1,700 hours of deidentified medical conversation datasets. The data is monophonic, and consists of doctor-patient conversations with labelled transcriptions. The audio data is recorded in the real and noisy environment and the transcription is noisy, for example, the alignments are off by the order of a few seconds.

7.2 Models

7.2.1 Input Features

All of our experiments in this study, we use 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 [80].

7.2.2 Output Units

In this study, we use two different output units: word+character (`WordChar`), byte-pair encoding (`BPE-1k`).

Word+Character units

We first explore the use of word and character for the output units of our 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 [64, 65, 66, 67, 68]. 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, we first restrict the vocabulary to 10k frequently occurring words. We then additionally use character units and start-of-OOV (sunk), end-of-OOV (eunk) tokens to make our 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.

BPE units

We also explore the use of subword-level symbols generated from the byte-pair encoding (BPE) algorithm [81]. The idea of the original BPE algorithm is iteratively merging the most frequent pairs of bytes in a sequence into a single, unused bytes. replacing the most frequent pair of bytes

in a sequence with a single, unused bytes. Sennrich, et al., adapt this for word segmentation by merging characters or character sequences instead. First, we represent each word in training transcript as a sequence of character symbols, we then count all symbol pairs (i.e. 'a', 'b') and replace each occurrence of the most frequent pair with a new merged symbol (i.e. 'ab') iteratively. From this way, we create the subword output units based on the frequency of occurreence in training sets. Similar to [82, 83], we use BPE-1k contains roughly 1k units (1,070), including 633 words and 437 sub-words.

7.2.3 Architecture

For the architecture of the end-to-end speech recognition, we used joint CTC/Seq2Seq end-to-end ASR [47, 48] as we proposed in Chapter 3. As suggested in [49, 59], 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, we used a location-based method [14]. For the decoder network, we used a 2-layer LSTM with 300 cells. In addition to the standard decoder network, our proposed models additionally require extra parameters for gating layers in order to fuse conversational context embedding to the decoder network compared to baseline. We denote the total number of trainable parameters in Table 7.2.

7.3 Training and Decoding

For the optimization method, we use AdaDelta [53] with gradient clipping [60]. We 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). We bootstrap the training of our proposed conversational end-to-end models from the baseline end-to-end models. To decide the best models for testing, we monitor the development accuracy where we always use the model prediction in order to simulate the testing scenario. At

inference, we used a left-right beam search method [55] with the beam size 10 for reducing the computational cost. We adjusted the final score, $s(\boldsymbol{y}|\boldsymbol{x})$, with the length penalty 0.5. The models are implemented using the PyTorch deep learning library [84], and ESPnet toolkit [47, 48, 58].

7.4 WER Results

In Table 7.2 summarizes our 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 7.2: Comparison of word error rates (WER) of baseline and our proposed conversational end-to-end ASR on SWBD task.

Model		#params.	LM	SWB	CH
Other E2E systems					
CTC [63]	Char output	53M	✓	19.8	32.1
CTC [66]	Word output, phone pretrain	n/a	✗	14.6	23.6
Seq2Seq [82]	BPE-1k, layer-wise pretrain	*150M	✗	13.1	26.1
LF-MMI [85]	Char output, data augment	26M	✓	13.0	23.6
Seq2Seq [83]	BPE-1k, data augment	360M	✗	7.2	14.6
Our systems					
our baseline	Char output	23M	✗	19.0	34.4
our baseline	Word-10k output	32M	✗	17.9	30.6
Our conv.E2E					
vanilla	Word-10k output	33M	✗	17.3	30.3
vanilla + gate	Word-10k output	35M	✗	17.2	29.8
+ fastText	Word-10k output	34M	✗	15.9	28.9
+ BERT	Word-10k output	34M	✗	**15.5**	29.0
+ LSTM-Attn	Word-10k output	34M	✗	15.6	**28.5**

We first present the other systems, our baseline system, and then our conversational systems. Unlike other systems, note that our system has relatively smaller model parameters and does not

use any phonetic information [66, 86], any external language model [63, 66, 86, 87], or any data augmentation [83], or any additional, complex training procedure (i.e. layer-wise pretraining [82] or hierarchical multi-task learning [87] or MBR training [88]).

We show the improved WER results by adding each of the individual components that we discussed in previous sections, our conversational system using onehot vectors for encoding conversational context with gated decoder (vanilla and vanilla + gate), using fastText for encoding conversational context (+ (2) fastText for context), using BERT for the conversational context (+ (3) BERT for context), increasing number of utterance history (+ (3) + 5-utterances). As shown in this Table 7.2, our best model (+ (3) + 5-utterances) gets around 13.4% relative improvement on the Switchboard (SWBD) evaluation set and 5.2% relative improvement on the CallHome (CH) evaluation set. Our vanilla conversational model with the gated decoder shows (Gated Decoder (1) onehot) 10.0% and 4.6% relative improvements over our baseline.

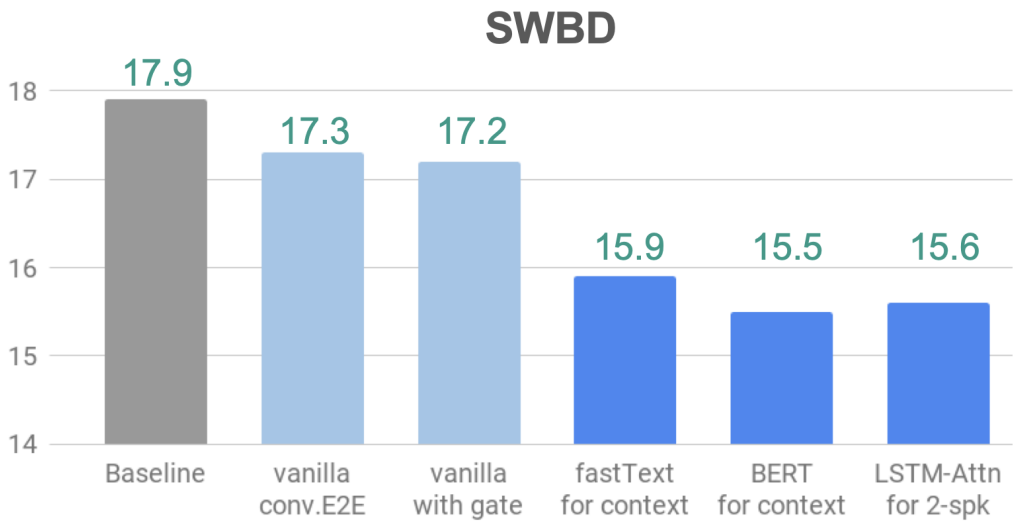


Figure 7.1: WER on SWBD evaluation set over different proposed context encoder methods

Next, we evaluated the use of pretrained external embeddings (fastText and BERT). We obtained further accuracy improvement by using external language resource, fastText or BERT for

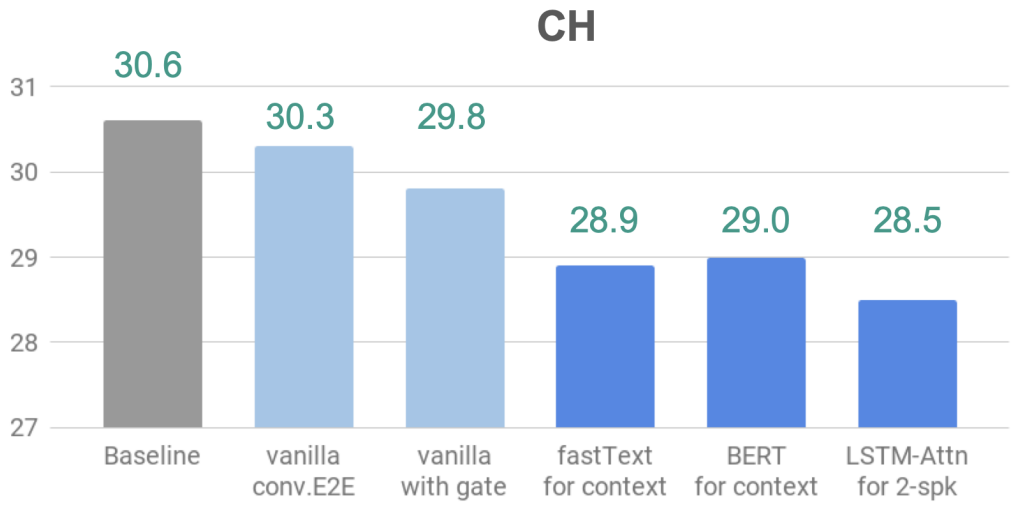


Figure 7.2: WER on CallHome evaluation set over different proposed context encoder methods

encoding conversational context vector. Using the BERT embeddings performed better than using the fastText embeddings, ((2) vs. (3)). It is possible that the BERT encodes conversational context vector better because it is contextualized embeddings.

We also investigate the effect of the number of utterance history being encoded. We tried different $N = [1, 5, 9]$ number of utterance histories to learn the conversational context embeddings. Typically increasing the number of utterance histories improves the accuracy. However, We also observed that the improvement diminished when we used 9-utterance history in case of using the concatenation. It is possible that the increased the number of trainable parameters of the concatenate model makes it harder for the model to train.

The WER over different types of proposed context encoder are also illustrated in Figure 7.2

Table 7.3 and 7.4 shows the WER results on larger dataset, Fisher (2,000 hours), and Medical (3,700 hours). As seen in these Tables, our proposed approach outperformed over baseline in large dataset consistently.

[90] is a conversational speech recognition model that relies on extensive system combination

Table 7.3: Comparison of word error rates (WER) of baseline and our proposed conversational end-to-end ASR on Fisher task.

Model		#params.	LM	SWB	CH
Other E2E systems					
CTC [63]	Char output	n/a	✓	10.2	17.7
CTC [66]	Word output, phone pretrain	n/a	✗	8.8	13.9
LF-MMI [85]	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
Our systems					
our baseline	BPE-1k output	24M	✗	9.5	17.3
our conv. E2E	BPE-1k output	25M	✗	**9.3**	**16.7**

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 our baseline, and multiple acoustic models and language models to produce the best score on this task thus far, to the best of our knowledge. Our baseline does not use an external LM like [63], or Minimum Bayes Risk Training like [88].

Table 7.4: Comparison of word error rates (WER) of baseline and our proposed conversational end-to-end ASR on Medical dataset.

Model		#params.	LM	Medical
Our systems				
our baseline	BPE-1k output	24M	✗	22.1
our conv. E2E	BPE-1k output	25M	✗	21.6

Statistical Significance

To understand the "statistical significance" of WER improvement using our conversational E2E models, we performed the significant test proposed in [1] by using the `compute-wer-bootci` too in the Kaldi library [91]. The confidence interval provides a range of reliability that defines how reliable is a observed improvement from our 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 our models.

The formulation of ΔW, the difference in the number of errors of two systems: baseline and conversational model, on 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 total number of 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 7.5 shows the bootstrap word error rates of our four systems trained on three sets: SWBD, +Fisher, +Medical with confidence estimates: 95% confidence intervals based on standard error and on bootstrap $B = 10^4$ [1]. The improvements of all our conversational models shows statistically significant, and the probability of improvement is 99.72 - 100.00 %.

Table 7.5: Probability of improvement of our conversational models over baselines based on 95% confidence intervals [1].

	Conv.E2E	Conv.E2E	Conv.E2E	Conv.E2E
Train set	SWBD	SWBD	+Fisher	+Medical
Output unit	wordchar	BPE-1k	BPE-1k	BPE-1k
#uttrs in Test set	4,458	4,458	4,458	15,111
Bootstrap WER (Base)	42.56	40.13	32.57	22.12
Bootstrap WER (Conv)	40.68	39.67	32.17	21.67
Confidence Intervals	[39.38 - 41.97]	[38.38 - 40.96]	[30.94 - 33.41]	[21.27 - 22.07]
Prob. Improvement	**100.00**	**99.72**	**99.99**	**100.00**

7.5 Validation of Context Learning

In addition to simply comparing WER performance, we show our analysis to demonstrate the effectiveness of our conversational end-to-end ASR in this Section.

7.5.1 Visualization of Attention Weights

Figure 7.3 visualizes the attention weights over the utterance-history. The figure shows the example when the model predicts the utterance (*come out here to California*) in evaluation set. The dark color represents higher attention weight.It shows that the model focuses on the long, informative utterances, rather than the short, meaningless utterances, i.e. *oh*, *heck yeah*, etc.

7.5.2 Comparison of Oracle and Random

We also compared WER of conversational end-to-end ASR with different input of context encoder: `Oracle`/`Random`/`Model output` at inference. In general, our model always use previous spoken utterances from the model output at inference time. To validate the effectiveness of our proposed framework, for the `Oracle` case we forwarded groundtruth utterance histories

Figure 7.3: The visualization of the attention weights over utterance-history of the speaker A (top) and the speaker B (bottom).

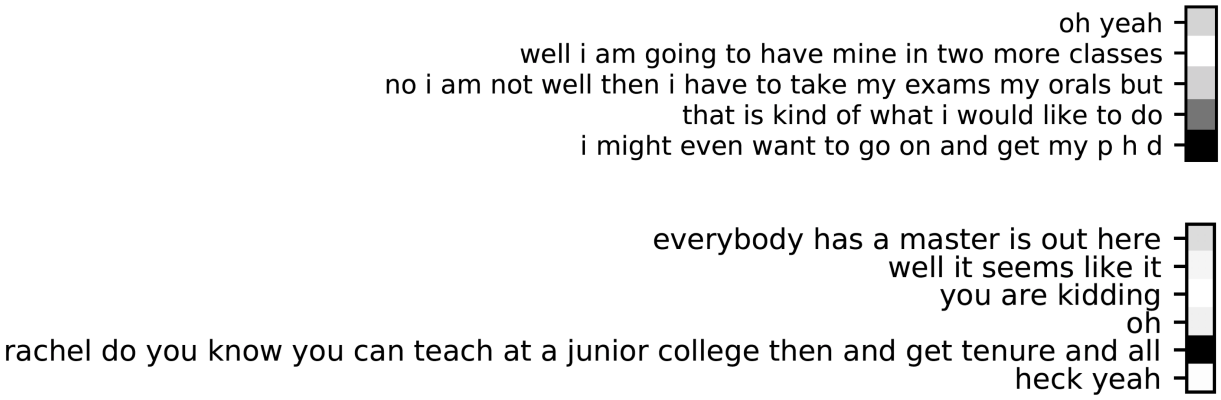
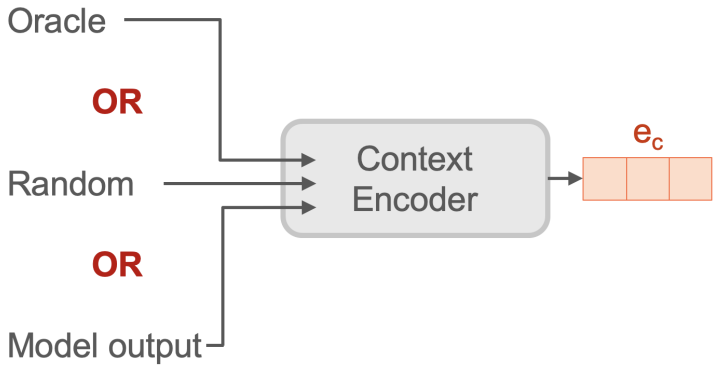


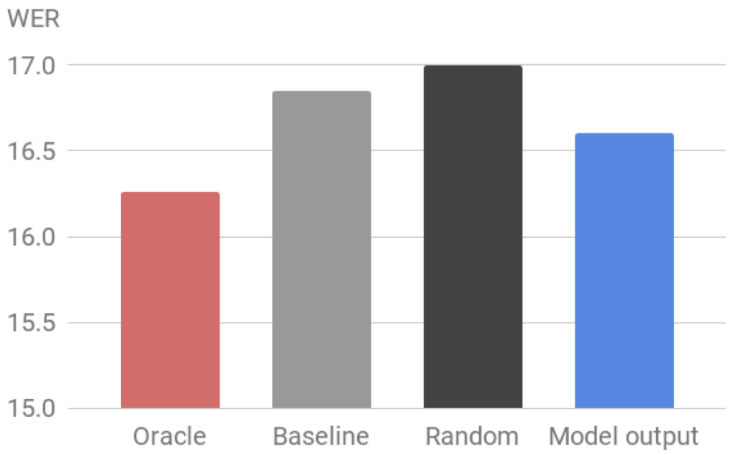
Figure 7.4: Context Encoder with different input at inference time: oracle/random/ model outputs



to encode context embeddings, and for the `Random` case we generate random output tokens and forwarded it to encode context embeddings as illustrated in Figure 7.4.

As in Figure 7.4, the `Oracle` case performed best as we expected, while the `Random` case was 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 our proposed method is coming from actually learning the conversational context, not jut from a random or regularization.

Figure 7.5: Comparison of WER of different inputs of our context encoder at inference time: oracle/random/ model outputs



7.5.3 Analysis of Conversational Consistency

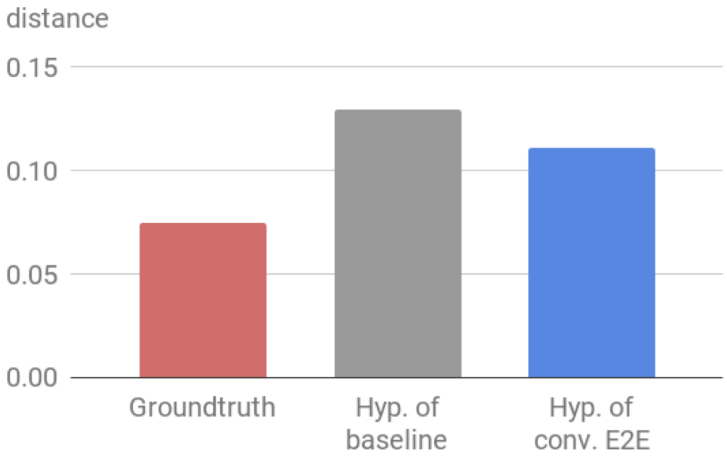
We also analyze whether our approach can conserve the conversational "consistency" in addition to WER improvement. To verify this, we develop a scoring function, $s(i, j)$ that measure the average distance between pairs of two consecutive hypotheses generated from particular model, i and j, in the evaluation sets. The conversational distance is calculated by the Cosine distance, dist(e_i, e_j) of the fixed-length vectors e_i, e_j which represent the model's i, j-th hypothesis, respectively. To obtain a fixed-length vector, utterance embedding, given the model hypothesis, we

use BERT sentence embedding as an oracle. Mathematically it can be written as,

$$s(i,j) = \frac{1}{N} \sum_{i,j \in \texttt{eval}} (\text{dist}(e_i, e_j))$$

where, i, j is a pair of consecutive hypotheses in evaluation data `eval`, N is the total number of i, j pairs, e_i, e_j are BERT embeddings. In our experiment, we select the pairs of consecutive utterances from the reference that show shorter distance score at least baseline hypotheses.

Figure 7.6: Comparison of the conversational distance score on the consecutive utterances of 1) reference, 2) hypothesis of baseline, and 2) hypothesis of our conversational end-to-end ASR.



From this process, we obtained three conversational distance scores from 1) the reference transcripts, 2) the hypotheses of our vanilla conversational model which is not using BERT, and 3) the hypotheses of our baseline model. As in Figure 7.6, our approach shows shorter distance score than the baseline. We found that our proposed model was 7.4% relatively closer to the reference than the baseline. This indicates that our conversational context embedding leads to improved similarity across adjacent utterances, resulting in better processing a long conversation. This implies that our approach can provide better conversational consistency in addition to the improved accuracy.

7.6 Examples

Figure 7.7, and 7.8, show examples from SWBD tasks, and Figure 7.9, and 7.10, show examples from Medical dataset. Each column shows utterances of reference, hypothesis of baseline, hypothesis of our conversational model in chronological order. The final row shows corresponding conversational distance score of the above utterances.

Reference	*Baseline*	*Conv.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*
0.118	*0.172*	*0.169*

Figure 7.7: Our context encoder generates conversational context embedding from previous spoken utterances

As in Figure 7.7, the blue word in reference column is "you guys" and baseline could not predict it, while our model can predict it well. As seen in the preceding utterance "oh boy you guys been all", the repeated word in current prediction may benefit from utterance history.

Reference	*Baseline*	*Conv.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***
0.130	*0.142*	*0.138*

Figure 7.8: Our context encoder generates conversational context embedding from previous spoken utterances

As in Figure 7.8, the blue word in reference column is "joints" and baseline could not predict it, while our model can predict it well. As seen in the preceding utterances, the semantically related words in current prediction may benefit from utterance history, i.e. doctor, muscle.

As in Figure 7.9, the blue word in reference column is "uh naprosyn" which is medicine name and baseline could not predict it, while our model can predict it well. As seen in the preceding

Reference	Baseline	Conv.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*
0.110	*0.161*	*0.141*

Figure 7.9: Our context encoder generates conversational context embedding from previous spoken utterances

utterances, even medical terms in current prediction may benefit from utterance history, as well as "world knowledge".

Reference	Baseline	Conv.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*
0.098	*0.128*	*0.115*

Figure 7.10: Our context encoder generates conversational context embedding from previous spoken utterances

As in Figure 7.10, the blue word in reference column is "psoriasis" which is disease name and baseline could not predict it, while our model can predict it well. As seen in the preceding utterances, even medical terms in current prediction may benefit from utterance history, as well as "world knowledge". We observed that even p is silence syllable, our model can predict it well.

As in Figure 7.11, the blue word in reference column is "plegridy" which is treatment name related to skin and baseline could not predict it, while our model can predict it well. As seen in the preceding utterances, even medical terms in current prediction may benefit from utterance history, as well as "world knowledge".

Reference	*Baseline*	*Conv.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*
0.120	*0.163*	*0.157*

Figure 7.11: Our context encoder generates conversational context embedding from previous spoken utterances

Chapter 8

Conclusions

8.1 Thesis Conclusions

In this thesis, we have presented an end-to-end speech recognition system that processes entire conversations. We first discussed the recent progress of end-to-end speech recognition systems that integrate all available information (e.g. acoustic, language resources) and directly transcribes speech to text using a single deep neural network architecture with a data-driven learning method. We then identified challenges that arise in modeling longer context in processing conversations, optimizing the system in an end-to-end manner. We proposed three novel techniques to address these challenges: 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 efficient way to integrate conversational contexts into end-to-end models using a gating mechanism; and 3) various methods to encode conversational contexts by using previous spoken utterances and augmenting with world knowledge using external linguistic resources (e.g. BERT, fastText). We showed accuracy improvements with three different large corpora, Switchboard (300 hours), Fisher (2,000 hours), and Medical conversation (3,700 hours), and shared our analysis to demonstrate the effectiveness of our approach. In this chapter, we will conclude the thesis and discuss future directions for conversational speech recognition system research.

In Chapter 3, we described our end-to-end ASR models that provide fast convergence and better accuracy performance. In Chapter 4, we described our conversational end-to-end ASR models that model entire conversations by two novel techniques 1) an efficient way to preserve conversational context, and 2) an effective way to integrate conversational context. In Chapter 6, we described various types of our context encoder that takes previous spoken utterances and generate a single, fixed dimensional context embedding. We explored different history unit and representation as an input of context encoder, and methods to augment "world knowledge" to encode rich and better context representation by using external word/sentence embeddings which trained on massive amount of textual data. Also we explored a way to learn turn-change information or interaction between two speakers in the case that we have the access of each speaker information. In Chapter 7, we showed improved accuracy on three large conversational tasks, Switchboard, Fisher, and Medical data, and shared analysis to demonstrate the effectiveness of our proposed framework. We compared oracle/random context encoder inputs, and developed a scoring function that measure conversational similarity.

8.2 Future Work

In this final section, we 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.

8.2.1 Acoustic Conversational Context

Our 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 our context encoder, we can use the previ-

ous 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.

8.2.2 From Audio to Semantics

Our approach can potentially 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 consists 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 understanding task, not errors of filler words. Evidence has recently been shown that optimizing directly for the semantic understanding task may yield better results [92, 93].

Bibliography

[1] Maximilian Bisani and Hermann Ney, "Bootstrap estimates for confidence intervals in asr performance evaluation," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2004, vol. 1, pp. I–409. (document), 7.4, 7.4, 7.5

[2] Susan E Brennan and Herbert H Clark, "Conceptual pacts and lexical choice in conversation.," *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 22, no. 6, pp. 1482, 1996. 1.1

[3] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010. 1.1, 2.1, 2.3.1

[4] Tomas Mikolov and Geoffrey Zweig, "Context dependent recurrent neural network language model.," *SLT*, vol. 12, pp. 234–239, 2012. 1.1, 2.3.1, 6.2

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

[6] Yangfeng Ji, Trevor Cohn, Lingpeng Kong, Chris Dyer, and Jacob Eisenstein, "Document context language models," *ICLR (Workshop track)*, 2016. 1.1, 2.3.1

[7] Bing Liu and Ian Lane, "Dialog context language modeling with recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5715–5719. 1.1, 2.3.1

[8] Wayne Xiong, Lingfeng Wu, Jun Zhang, and Andreas Stolcke, "Session-level language

modeling for conversational speech," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 2764–2768. 1.1

[9] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 369–376. 1.1, 2.2, 2.2.1

[10] Alex Graves and Navdeep Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1764–1772. 1.1, 2.2, 2.2

[11] 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

[12] Yajie Miao, Mohammad Gowayyed, and Florian Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015, pp. 167–174. 1.1, 2.2, 2.2, 2.2.1, 3.1

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

[14] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, "Attention-based models for speech recognition," in *Advances in neural information processing systems*, 2015, pp. 577–585. 1.1, 2.2, 2.2, 2.2.2, 2.2.2, 3.1, 4.3.2, 5.3.2, 7.2.3

[15] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE,

2016, pp. 4960–4964. 1.1, 2.2, 2.2

[16] Steven Davis and Paul Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE transactions on acoustics, speech, and signal processing*, vol. 28, no. 4, pp. 357–366, 1980. 2.1

[17] Hynek Hermansky, "Perceptual linear predictive (plp) analysis of speech," *the Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738–1752, 1990. 2.1

[18] Chanwoo Kim and Richard M Stern, "Power-normalized cepstral coefficients (pncc) for robust speech recognition," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 24, no. 7, pp. 1315–1329, 2016. 2.1

[19] Leonard E Baum and Ted Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The annals of mathematical statistics*, vol. 37, no. 6, pp. 1554–1563, 1966. 2.1

[20] 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, IEEE*, vol. 29, no. 6, pp. 82–97, 2012. 2.1, 2.2

[21] Nelson Morgan and Herve Bourlard, "Continuous speech recognition using multilayer perceptrons with hidden markov models," in *International conference on acoustics, speech, and signal processing*. IEEE, 1990, pp. 413–416. 2.1

[22] George E Dahl, Dong Yu, Li Deng, and Alex Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on audio, speech, and language processing*, vol. 20, no. 1, pp. 30–42, 2011. 2.1

[23] Navdeep Jaitly, Patrick Nguyen, Andrew Senior, and Vincent Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition," in *Thirteenth*

Annual Conference of the International Speech Communication Association, 2012. 2.1

[24] 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 *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 8604–8608. 2.1

[25] Frank Seide, Gang Li, and Dong Yu, "Conversational speech transcription using context-dependent deep neural networks.," in *Interspeech*, 2011, pp. 437–440. 2.1

[26] 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 *2012 IEEE international conference on Acoustics, speech and signal processing (ICASSP)*. IEEE, 2012, pp. 4277–4280. 2.1

[27] Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran, "Deep convolutional neural networks for lvcsr," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 8614–8618. 2.1

[28] 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 *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2013, pp. 315–320. 2.1

[29] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. 2.1, 2.4, 3.3.2, 5.1

[30] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 273–278. 2.1, 3.3.2

[31] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics,*

speech and signal processing. IEEE, 2013, pp. 6645–6649. 2.1

[32] 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

[33] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4580–4584. 2.1

[34] 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

[35] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio, "End-to-end attention-based large vocabulary speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4945–4949. 2.2, 2.2, 2.2.2, 2.2.2

[36] Liang Lu, Xingxing Zhang, and Steve Renals, "On training the recurrent neural network encoder-decoder for large vocabulary end-to-end speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 5060–5064. 2.2

[37] Abdel-rahman Mohamed, George E Dahl, and Geoffrey Hinton, "Acoustic modeling using deep belief networks," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 14–22, 2012. 2.2

[38] David M Blei, Andrew Y Ng, and Michael I Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003. 2.3.1

[39] 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 *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5255–5259. 2.3.1

[40] Golan Pundak, Tara N Sainath, Rohit Prabhavalkar, Anjuli Kannan, and Ding Zhao, "Deep context: end-to-end contextual speech recognition," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 418–425. 2.3.2

[41] Uri Alon, Golan Pundak, and Tara N Sainath, "Contextual speech recognition with difficult negative training examples," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6440–6444. 2.3.2

[42] 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

[43] Pawel Swietojanski and Steve Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," in *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2014, pp. 171–176. 2.4

[44] 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

[45] Jamie Kiros, William Chan, and Geoffrey Hinton, "Illustrative language understanding: Large-scale visual grounding with image search," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, vol. 1, pp. 922–933. 2.4, 5.2.2

[46] Suyoun Kim and Michael L Seltzer, "Towards language-universal end-to-end speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4914–4918. 2.4, 5.1, 5.2.2

[47] Suyoun Kim, Takaaki Hori, and Shinji Watanabe, "Joint ctc-attention based end-to-end speech recognition using multi-task learning," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 4835–4839. 3, 4.3.2, 4.3.2, 5.3.2, 7.2.3, 7.3

[48] Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R Hershey, and Tomoki Hayashi, "Hybrid ctc/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017. 3, 4.3.2, 4.3.2, 5.3.2, 7.2.3, 7.3

[49] 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," *Interspeech*, 2017. 3.2, 4.3.2, 5.3.2, 7.2.3

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

[51] John Garofalo, David Graff, Doug Paul, and David Pallett, "CSR-I (wsj0) complete," *Linguistic Data Consortium, Philadelphia*, 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," in *Computer Speech and Language, to appear*. 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 *Advances in neural information processing systems*, 2014, pp. 3104–3112. 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," *IEEE signal processing letters*, vol. 6, no. 1, pp. 1–3, 1999. 4.1.2

[58] 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, pp. 2207–2211. 4.3.2, 4.3.2, 7.3

[59] Yu Zhang, William Chan, and Navdeep Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 4845–4849. 4.3.2, 5.3.2, 7.2.3

[60] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, 2013, pp. 1310–1318. 4.3.2, 7.3

[61] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton, "Chainer: a next-generation open source framework for deep learning," in *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*, 2015. 4.3.2

[62] Thomas Zenkel, Ramon Sanabria, Florian Metze, Jan Niehues, Matthias Sperber, Sebastian Stüker, and Alex Waibel, "Comparison of decoding strategies for ctc acoustic models," *arXiv preprint arXiv:1708.04469*, 2017. 4.2

[63] Geoffrey Zweig, Chengzhu Yu, Jasha Droppo, and Andreas Stolcke, "Advances in all-neural speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 4805–4809. 4.2, 7.2, 7.4, 7.3, 7.4

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

[65] Kartik Audhkhasi, Bhuvana Ramabhadran, George Saon, Michael Picheny, and David Nahamoo, "Direct acoustics-to-word models for english conversational speech recognition," *CoRR*, vol. abs/1703.07754, 2017. 5.3.1, 7.2.2

[66] Kartik Audhkhasi, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Michael Picheny, "Building competitive direct acoustics-to-word models for english conversational speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4759–4763. 5.3.1, 5.1, 5.3.3, 7.2.2, 7.2, 7.4, 7.3

[67] Jinyu Li, Guoli Ye, Amit Das, Rui Zhao, and Yifan Gong, "Advancing acoustic-to-word ctc model," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5794–5798. 5.3.1, 7.2.2

[68] Shruti Palaskar and Florian Metze, "Acoustic-to-word recognition with sequence-to-sequence models," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 397–404. 5.3.1, 7.2.2

[69] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 1171–1179. 6.1.3

[70] 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

[71] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 19–27. 6.2

[72] Jeffrey Pennington, Richard Socher, and Christopher Manning, "Glove: Global vectors

for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543. 6.2

[73] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017. 6.2

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

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

[76] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi, "Bidirectional attention flow for machine comprehension," *CoRR*, vol. abs/1611.01603, 2017. 6.2

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

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

[79] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly, "Pointer networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2692–2700. 6.3.2

[80] Yajie Miao, Mohammad Gowayyed, Xingyu Na, Tom Ko, Florian Metze, and Alexander Waibel, "An empirical exploration of ctc acoustic models," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 2623–2627. 7.2.1

[81] Rico Sennrich, Barry Haddow, and Alexandra Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015. 7.2.2

[82] Albert Zeyer, Kazuki Irie, Ralf Schlüter, and Hermann Ney, "Improved training of end-to-end attention models for speech recognition," *Interspeech*, 2018. 7.2.2, 7.2, 7.4

[83] 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. 7.2.2, 7.2, 7.4

[84] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017. 7.3

[85] Hossein Hadian, Daniel Povey, Hossein Sameti, Jan Trmal, and Sanjeev Khudanpur, "Improving lf-mmi using unconstrained supervisions for asr," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 43–47. 7.2, 7.3

[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, pp. 2751–2755. 7.4

[87] Ramon Sanabria and Florian Metze, "Hierarchical multitask learning with ctc," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 485–490. 7.4

[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," 09 2018, pp. 761–765. 7.4, 7.3, 7.4

[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 *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 206–213. 7.3

[90] W. Xiong, L. Wu, Fil Alleva, Jasha Droppo, Xuedong Huang, and Andreas Stolcke, "The microsoft 2017 conversational speech recognition system," 2017, vol. abs/1708.06073. 7.4

[91] 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 *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011, number CONF. 7.4

[92] 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 *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 720–726. 8.2.2

[93] Dmitriy Serdyuk, Yongqiang Wang, Christian Fuegen, Anuj Kumar, Baiyang Liu, and Yoshua Bengio, "Towards end-to-end spoken language understanding," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5754–5758. 8.2.2