Context Dependent RNNLM for Automatic Transcription of Conversations

Srikanth Raj Chetupalli, Sriram Ganapathy

LEAP Lab, Indian Institute of Science, Bengaluru, India

sraj@iisc.ac.in, sriramg@iisc.ac.in

Abstract

Conversational speech, while being unstructured at an utterance level, typically has a macro topic which provides larger context spanning multiple utterances. The current language models in speech recognition systems using recurrent neural networks (RNNLM) rely mainly on the local context and exclude the larger context. In order to model the long term dependencies of words across multiple sentences, we propose a novel architecture where the words from prior utterances are converted to an embedding. The relevance of these embeddings for the prediction of next word in the current sentence is found using a gating network. The relevance weighted context embedding vector is combined in the language model to improve the next word prediction, and the entire model including the context embedding and the relevance weighting layers is jointly learned for a conversational language modeling task. Experiments are performed on two conversational datasets - AMI corpus and the Switchboard corpus. In these tasks, we illustrate that the proposed approach yields significant improvements in language model perplexity over the RNNLM baseline. In addition, the use of proposed conversational LM for ASR rescoring results in absolute WER reduction of 1.2% on Switchboard dataset and 1.0% on AMI dataset over the RNNLM based ASR baseline.

Index Terms: Language modeling, recurrent neural network, conversational modeling, speech recognition.

1. Introduction

Language modeling (LM) is the task of predicting the next word given the past history of words in a text stream and it forms an integral part of automatic speech recognition (ASR) systems and other natural language processing systems. In the past decade, following a similar trend in several other domains, the methods used for LM have shifted fundamentally from n-gram models based on frequency of counts to deep neural network based models. The earliest approach to LM using feed-forward networks was proposed by Bengio et al. [1]. These models were advanced using recurrent neural networks (RNNs) by Mikolov et. al [2] and then further using long short-term memory (LSTM) variants of RNNs by Sundermeyer et al. [3] and more recently by Xiong et al. [4]. The RNN models have multiple advantages over the traditional n-gram framework. The continuous-space embedding of words allows word similarities to be computed in an efficient manner for generalization [2], and the recurrent architecture also allows an unlimited history to condition the prediction of next word.

In the current implementation of LM (even for conversations), the potential advantage of unlimited history, however, is not used to its full extent. The LM is typically "reset" at the start of each utterance in current state-of-the-art ASR systems [5, 4]. This approach assumes that the successive utterances are independent of each other, which is not the case in conversational data. In the past, there have been some attempts to incorporate the information from a larger context, spanning multiple speaker utterances, in n-gram models (for example, Bellegarda et. al [6] and Ji et.al [7]). For neural network based LM, the inclusion of a longer context as a slowly varying context vector was attempted by Mikolov et.al. [8] where the context vector was incorporated as a latent semantic embedding of the previous context. Xiong et.al [9] recently proposed a LSTM based LM that takes the history of the conversational utterances in the LSTM model. However, the modeling of long term dependencies spanning multiple sentences can pose a substantial challenge in LSTM models. In addition, all the words of the current sentence may not benefit from the inclusion of the longer context.

In this paper, we propose a novel approach to LM, referred to as context dependent RNNLM (CRNNLM), using the word embeddings from contextual sentences along with the words of the current sentence. The cross correlation of the word embeddings from the contextual sentences with the embedding of the current word is used to derive a context embedding. This contextual embedding is used in a gating network to generate a relevance weighted context vector that is combined with the word embeddings of the current sentence in the LM. Unlike the approach in [8], where a single context vector is computed for the sentence, a separate context vector is computed for each word in the current sentence to capture local context. The entire model, including the gating network and embedding layers, is learned jointly from the training data corresponding to conversational speech. Experiments on Switchboard dataset [10] and AMI meeting dataset [11] show that the proposed approach to LM improves significantly over the state-of-art RNNLM in terms of perplexity measure, and also in terms of WER in the ASR task.

The rest of the paper is organized as follows. Section 2 describes the proposed model to LM using contextual embeddings. The experiments and results using the proposed LM are reported in Section 3, which is followed by a brief summary of the work in Section 4.

2. Context dependent RNNLM

A block diagram of the proposed context dependent RNNLM (CRNNLM) is shown in Fig. 1. The utterances in the conversation are serialized, based on the onset time, as shown in Fig. 2. To model a given utterance, a set of past c utterances is considered as the context. The current utterance and the context utterances are given as two separate input streams to the neural network. The neural network is divided into three stages. In the first stage, a hidden representation is computed for each word in the input and context streams. In the second stage, a cross-attention like operation [12] between input and context streams, is used to derive a context embedding separately for each word

This work was funded by British Telecom India Research Center (BTIRC) project on Speech Analytics.

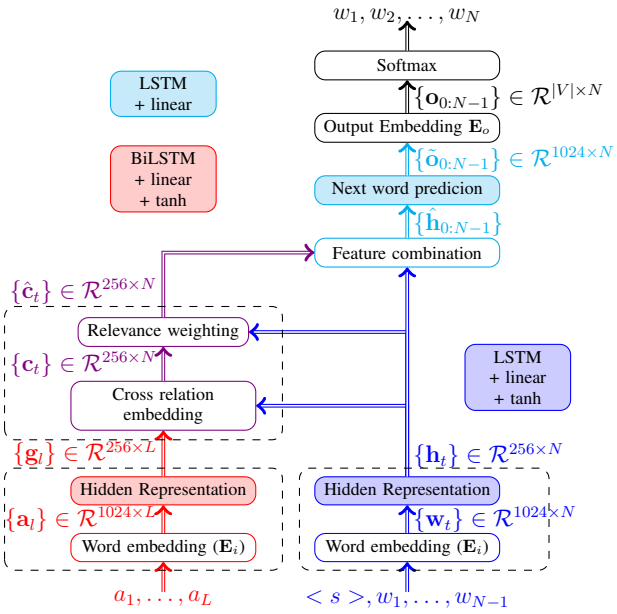

Figure 1: *Block diagram of the C-RNNLM architecture.*

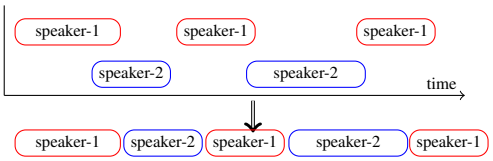

Figure 2: *Conversation progression and utterance serialization.*

in the input stream. The context embedding is further gated using a relevance weight, which quantifies the importance of context in predicting the next word. In the final stage, the relevance weighted context embedding is combined with the input hidden representation and fed to a next word prediction network. A detailed description of the different computation steps is summarized below.

2.1. Input representation

Let $\{\mathbf{w}_0, \ldots, \mathbf{w}_{N-1}\}$ denote the sequence of word embeddings of the current input sentence containing words $< s >, w_1, \ldots, w_{N-1}$. The hidden representation $\mathbf{h}_t$ for the t^{th} input word is obtained via the following operations:

$$\mathbf{h}_t = tanh\left(\text{linear}\left(\text{LSTM}\left(\{\mathbf{w}_t\}\right)\right)\right), \ \forall\, t. \tag{1}$$

At time instant t, the uni-directional LSTM computes a representation dependent on the words $w_{\leq t}$, which is linearly projected and the tanh activation is used for range compaction. The word embedding $\mathbf{w}_t$ is obtained from 1-hot-K representation of input word w_t using the input embedding matrix $\mathbf{E}_i$. We use the transpose of the output embedding matrix $\mathbf{E}_o$ as the input embedding matrix, i.e., $\mathbf{E}_i = \mathbf{E}_o^T$, inspired by [13].

2.2. Context embedding

Let L denote the number of words from context (previous c) utterances, and let $\{\mathbf{a_1}, \ldots, \mathbf{a_L}\}$ be the sequence of corresponding word embeddings computed using the input embedding matrix $\mathbf{E}_i$ shown in Fig. 1. The hidden representation $\mathbf{g}_l$ for the l^{th} word in the context input is computed as:

$$\mathbf{g}_l = tanh\left(\text{linear}\left(\text{BiLSTM}\left(\{\mathbf{a}_l\}\right)\right)\right), \ \forall\, l. \tag{2}$$

The BiLSTM layer looks at all the words in the context, and the linear layer with tanh activation is used for joint projection of the forward and backward LSTM output streams and range compaction. The context embedding $\mathbf{c_t}$ corresponding to the word w_t is then computed as a weighted sum of representations $\{\mathbf{g}_1, \ldots, \mathbf{g}_L\}$ of context input.

$$\mathbf{c}_t = \sum_{l=1}^{L} \alpha_{tl} \mathbf{g}_l, \tag{3}$$

where the weight vector $\boldsymbol{\alpha}_t = [\alpha_{t1}, \ldots, \alpha_{tL}]^T$ is computed using cross correlation as,

$$\boldsymbol{\alpha}_t = softmax(\mathbf{h}_t^T [\mathbf{g}_1, \ldots, \mathbf{g}_L]). \tag{4}$$

The correlation between $\mathbf{h}_t$ and $\mathbf{g}_l$ will be high, if the corresponding words are related, or occur frequently in a similar context. Hence, the derived context embedding $\mathbf{c}_t$ captures local context information.

2.3. Relevance weighted context embedding

The context may not be relevant to predict the next word in a word sequence. To model this, we consider relevance weighting of the context embedding using joint, linear projection of the context embedding and the input representation. This method is inspired by the cold fusion method proposed in [14]. We explore two strategies for relevance weighting, (i) coarse weighting and (ii) fine weighting.

In coarse weighting, the context embedding is multiplied by a *scalar* weight: $\hat{\mathbf{c}}_t = \beta_t \mathbf{c}_t$, where

$$\beta_t = sigmoid\left(\mathbf{w}^T \left[\mathbf{h}_t^T; \mathbf{c}_t^T\right]^T\right), \ \mathbf{w} \in \mathcal{R}^{2h \times 1}, \tag{5}$$

and in fine weighting, the context embedding is multiplied element-wise by a *vector* weight: $\hat{\mathbf{c}}_t = \boldsymbol{\beta}_t \odot \mathbf{c}_t$, where

$$\boldsymbol{\beta}_t = sigmoid\left(\mathbf{W}^T \left[\mathbf{h}_t^T; \mathbf{c}_t^T\right]^T\right), \ \mathbf{W} \in \mathcal{R}^{2h \times h}. \tag{6}$$

2.4. Feature combination

We explore additive and concatenative schemes for combining the relevance weighted context embedding vector with the input feature. In the additive scheme, the combined feature is obtained as,

$$\hat{\mathbf{h}}_t = \mathbf{h}_t + \hat{\mathbf{c}}_t, \tag{7}$$

and in the concatenative scheme, the combined feature is obtained as,

$$\hat{\mathbf{h}}_t = \left[\mathbf{h}_t^T; \hat{\mathbf{c}}_t^T\right]^T. \tag{8}$$

2.5. Next word prediction and output embedding

The next word prediction network consists of a uni-directional LSTM layer followed by a linear layer, which gives an embedding vector as the output $\tilde{\mathbf{o}}_t$:

$$\tilde{\mathbf{o}}_t = \text{Linear}\left(\text{LSTM}\left(\hat{\mathbf{h}}_t\right)\right) \tag{9}$$

The output embedding matrix $\mathbf{E}_o$ is used to project the embedding $\tilde{\mathbf{o}}_t$ to the word level output $\mathbf{o}_t$ ($\mathbf{o}_t = \mathbf{E}_o \tilde{\mathbf{o}}_t$). The softmax of the vector $\mathbf{o}_t$ is taken as the word level posterior distribution for the next word prediction.

Table 1: *CRNNLM architecture variants*

Name	Relevance weight	Feature combination
V1	Unity	Concatenative
V2	Scalar	Concatenative
V3	Vector	Concatenative
V4	Vector	Additive

3. Experiments and results

We consider evaluation of the proposed language model using two datasets, (i) Switchboard (SWB) telephone conversation dataset (300 hour subset) [10], and (ii) AMI meeting dataset [11]. The CRNNLM is trained separately for each dataset, and uses the same architecture. The vocabulary size is 30278 words for the Switchboard dataset and 46950 for the AMI dataset. The number of training conversations is 2405 and 130, for the Switchboard and AMI datasets respectively. Further, we augment the training set with 11699 Fisher conversation transcripts [15] for both the setups. The total number of sentences and words in the training set is (3M,31.7M), (2.5M,24.8M) for the Switchboard and AMI setups respectively. The evaluation set consists of 40 conversations in the Switchboard and 10 conversations in the AMI dataset. The training set is divided into 50 subsets with approximately equal number of conversations. The LM is trained with cross entropy loss using Adam optimizer [16]. The initial learning rate is chosen to be 0.01, and decreased with a constant decay coefficient of 0.99 after every subset (starting from 20^{th} subset). The batch size is chosen to be 32 samples. The length of input training examples is restricted to a maximum of 16 words, and the context example length is not constrained. A ℓ_2 regularization with a weight of 10^{-5} is applied to all the weights in the network, and dropout regularization with $p = 0.2$ is used for the LSTM layer outputs. The CRNNLM is trained using the PyTorch toolkit [17].

We explore four variants of the CRNNLM architecture, which are summarized in Table 1. We also compare CRNNLM with the trigram LM, RNNLM implemented using standard Kaldi (Kaldi-RNNLM) recipe, and the session level language model (SessionLM) proposed in [9]. For the SessionLM, we choose three hidden layers as in [9], but with 256 units each, and the embedding used to encode the word inputs is trained jointly with the LM. We consider the variant of SessionLM in [9] without the speaker change and speaker overlap information, and the network is trained similar to CRNNLM. During training words from previous 10 sentences are used as session history and during evaluation all the past sentences in the conversation are used as session history.

3.1. Discussion and analysis

First, we investigate the importance of context embedding, using the CRNNLM architecture V3 trained on Switchboard dataset. Fig. 3 shows the average relevance weight $\boldsymbol{\beta}_t$ for 5 different test utterances. A higher value indicates the network is giving importance to the context embedding. We see that, on an average, a higher weight is given to the context embedding in the prediction of the first few words of the current utterance, and the context becomes less relevant gradually. Also, there is a difference across the utterances, indicating that the context embedding can be more relevant in certain local context scenarios. To explore this further, we study the perplexity of the prediction of first word in the utterance.

Figure 4 shows a comparison of the probability density function (PDF) of the first word perplexity (FWP). We see that,

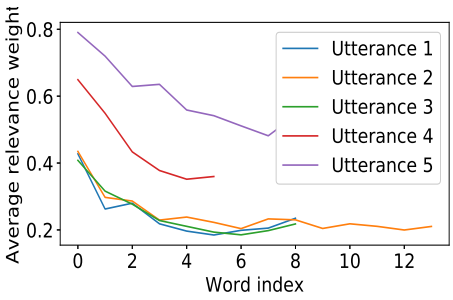

Figure 3: *Relevance weight $\boldsymbol{\beta}_{t=1:N}$ averaged across the context embedding dimension, for the CRNNLM architecture V3, for five different utterances from Switchboard evaluation set.*

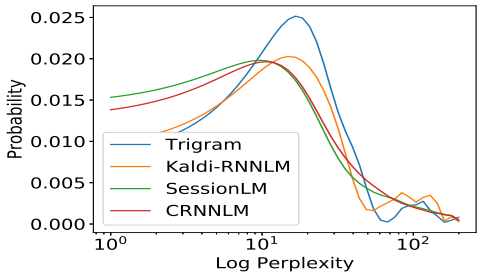

Figure 4: *Comparison of the PDF (kernel density estimate) of first word perplexity, i.e., $1/P(word| < s >)$ for all utterances in the evaluation set of Switchboard dataset.*

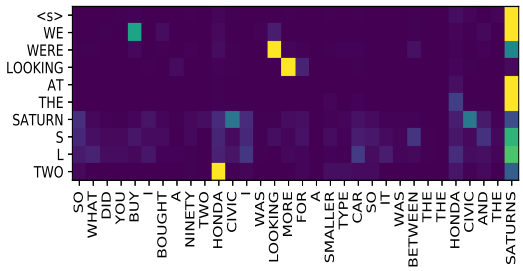

Figure 5: *Cross relation weights $\boldsymbol{\alpha}_t$ (eqn. (4)) computed in the CRNNLM architecture V3. The x-axis corresponds to the words in context utterances from both the speakers.*

the PDF is shifted towards zero for CRNNLM (network V3) and SessionLM, indicating better first word prediction compared to trigram or Kaldi-RNNLM. The min value of FWP for trigram and Kaldi-RNNLM is found to be 8 and 9.43 respectively, but CRNNLM predicts words with a higher probability in certain contexts, and the min FWP is 1.08.

Fig. 5 shows the cross relation weights $\boldsymbol{\alpha}_t$ of eqn. (4) for the sentence "We were looking at the Saturn S L two", which was part of a Switchboard conversation. For the prediction of first word ("We"), i.e., for $< s >$ as input, a relatively higher weight is given to the last word in the context input, indicating that the network is trying to ensure continuity across the sentences. At $t = 3$, '$< s >$ We were' is given as input, a higher weight is given to the word 'looking' in the context input which is also the correct next word; and in the next time step, the weight is high for the 'more', which corresponds to the sub-sequence 'looking more' in the context input. For the subsequent time steps, the weights are spread across all words with relatively higher weights for the words {'Honda',

Table 2: *LM Perplexity of various models.*

Net	Perplexity	
	SWB	AMI
trigram	112.32	124.05
Kaldi-RNNLM	93.00	100.22
SessionLM [9]	61.94	79.28
V1	63.60	72.99
V2	58.69	73.45
V3	58.10	**71.64**
V4	**57.88**	73.72

Table 3: *Effect of (a) number of alternate hypotheses (N), and (b) length of context c, on the ASR performance (WER %).*

(a) WER vs N ($c = 2$)

N	SWB	AMI
20	13.5	19.0
30	13.5	19.0
50	13.3	18.8
70	13.3	18.7
100	**13.2**	**18.6**

(b) WER vs c ($N = 100$)

c	SWB	AMI
0	13.5	19.3
1	13.2	18.7
2	13.2	18.6
3	**13.1**	18.6
4	**13.1**	**18.5**

'Civic','Saturns','Car'}, indicating the computation of an average embedding with a preference to the macro topic of the conversation. Overall, we observe that, the CRNNLM approach promotes continuity across utterances, and gives higher importance to sub-sequences from context input to improve the next word prediction.

We study the LM performance using perplexity measure. Table 2 shows the LM perplexity on the evaluation set. We see that, neural LMs (CRNNLMs, Kaldi-RNNLM and SessionLM [9]) show significant improvement in perplexity compared to the trigram LM. The performance of CRNNLM is better than the Kaldi-RNNLM and the SessionLM [9]. Comparing architecture V1 with V2-V4, we see that relevance weighting improves the LM perplexity. The feature combination method (concatenative or additive), and the relevance weighting scheme (coarse or fine) are found to have a minor effect on perplexity.

3.2. ASR Experiments

The acoustic model (AM) is trained using the Kaldi "chain" recipe [18] with LF-MMI [19] as the minimization objective. A Bi-LSTM architecture[1] is used for the AM for Switchboard dataset, and a TDNN based architecture [2] is used for the AMI dataset. We consider experimentation using the individual head microphone (IHM) subset of the AMI dataset. A trigram language model, built on the training transcripts is used to perform first pass decoding to generate decoder lattices. The neural LMs are then used to rescore the N-best hypothesis list obtained from these lattices. The utterances are processed sequentially in CRNNLM and SessionLM [9] rescoring. To rescore a given utterance hypotheses, a set of past c decoded utterances are given as context input in CRNNLM.

First, we study the effect of the number N of alternate hypotheses considered for rescoring. Table 3(a) shows the WER as a function of N. The WER is found to decrease with increase in N. The effect of the number c of context utterances on WER is shown in table 3(b). The CRNNLM is trained with $c = 2$ and it is varied during evaluation. For $c = 0$ (no context), we consider the symbol '$< unk >$' as the context input. Compared with

[1]KALDI-ROOT/egs/swbd/s5c/local/chain/tuning/run_blstm_6k.sh

[2]KALDI-ROOT/egs/ami/s5b/local/chain/tuning/run_tdnn_1j.sh

Table 4: *ASR performance for different LMs and CRNNLM variants, context $c = 3$ and $N = 100$.*

LM	WER %	
	SWB	AMI
First pass		
trigram	16.0	20.6
Second pass (trigram +)		
Kaldi-RNNLM	13.8	19.2
SessionLM [9]	13.4	19.0
V1	13.4	18.6
V2	13.1	18.7
V3	13.1	18.6
V4	13.1	18.6
Third pass (trigram + Kaldi-RNNLM +)		
V3	**12.6**	**18.2**

$c = 0$, we see that the WER is better for $c > 1$, indicating the usefulness of context. We see that, a context of 3 utterances is sufficient and longer context does not necessarily result in WER improvement. This may be attributed to the local nature of the context embedding, which promotes repeated word sequences and continuity across utterances. Hence, we consider $N = 100$ and $c = 3$ for the following ASR experiments.

The WER obtained using the different CRNNLM variants is shown in Table 4. Rescoring using Kaldi-RNNLM is found to give an absolute improvement of upto 2.2% compared to trigram LM, and using SessionLM gives upto 2.6% in WER. The CRNNLMs are found to result in better WER than other models for both the datasets. Comparing V2-V4 with V1, we see that, relevance weighting of the context embedding helps in improving the WER. The architectures V2-V4 have similar performance for $N = 100$, but for smaller N we observed V3 to have slightly better performance. Table 4 also shows that CRNNLM rescoring of the N-best list generated using Kaldi-RNNLM (last row) improves the WER by absolute $1 - 1.2\%$ compared to Kaldi-RNNLM. Statistical significance analysis using bootstrap-CI approach [20], computed using the Kaldi tool *compute-wer-bootci*, showed the probability of improvement (POI) of the CRNNLM variants and the SessionLM to be 1 compared to the Kaldi-RNNLM output. POI of the four CRNNLM variants V1-V4 compared to SessionLM is found to be $0.05, 0.90, 0.84, 0.92$ respectively, indicating that the output of CRNNLM architectures V2-V4 are statistically different from Kaldi-RNNLM and SessionLM significantly, and architecture V1 output is closer to SessionLM.

4. Summary

In this paper, the use of conversation context from past utterances is shown to improve the LM and ASR transcription performance. The cross attention like architecture is found to extract local context features and also improve the next word prediction near the start of the sentence. The relevance weighting of the context features is also important and useful in improving the LM performance. An absolute improvement of $1 - 1.2\%$ in WER is obtained on two different types of conversations, telephone and multi-party meetings.

5. Acknowledgments

The authors would like to acknowledge the technical discussions with Abhishek Anand and Dr. Michael Free of BT Research that helped in shaping the paper.

6. References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, p. 1137–1155, 2003.

[2] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Proceedings of INTERSPEECH*, 2010, pp. 1045–1048.

[3] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," in *Proceedings of INTERSPEECH*, 2012, pp. 194–197.

[4] W. Xiong, J. Droppo, X. Huang, F. Seide, M. L. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "Toward human parity in conversational speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 12, pp. 2410–2423, 2017.

[5] G. Saon, G. Kurata, T. Sercu, K. Audhkhasi, S. Thomas, D. Dimitriadis, X. Cui, B. Ramabhadran, M. Picheny, L.-L. Lim, B. Roomi, and P. Hall, "English conversational telephone speech recognition by humans and machines," in *Proceedings of INTERSPEECH*, 2017, pp. 132–136.

[6] J. R. Bellegarda, "Statistical language model adaptation: review and perspectives," *Speech communication*, vol. 42, no. 1, pp. 93–108, 2004.

[7] G. Ji and J. Bilmes, "Multi-speaker language modeling," in *Proceedings of HLT-NAACL 2004: Short Papers*, 2004, pp. 133–136.

[8] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model," in *IEEE Spoken Language Technology Workshop (SLT)*, Dec 2012, pp. 234–239.

[9] W. Xiong, L. Wu, J. Zhang, and A. Stolcke, "Session-level language modeling for conversational speech," in *Proceedings of Empirical Methods in Natural Language Processing*, 2018, pp. 2764–2768.

[10] E. H. John J. Godfrey, "Switchboard-1 release 2 LDC97S62," 1993, linguistic Data Consortium.

[11] J. Carletta *et al.*, "The AMI meeting corpus: A pre-announcement," in *Proceedings of the Second International Conference on Machine Learning for Multimodal Interaction*. Springer-Verlag, 2005, p. 28–39.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[13] O. Press and L. Wolf, "Using the output embedding to improve language models," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 2017, pp. 157–163.

[14] A. Sriram, H. Jun, S. Satheesh, and A. Coates, "Cold fusion: Training seq2seq models together with language models," in *Proceedings of INTERSPEECH*, 2018, pp. 387–391.

[15] C. Cieri *et al.*, "Fisher english training speech part 1 transcripts LDC2004T19," 2004.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[17] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.

[18] D. Povey *et al.*, "The kaldi speech recognition toolkit," in *IEEE Workshop on Automatic Speech Recognition and Understanding*, Dec. 2011.

[19] ——, "Purely sequence-trained neural networks for ASR based on lattice-free MMI," in *Interspeech*, 2016, pp. 2751–2755.

[20] M. Bisani and H. Ney, "Bootstrap estimates for confidence intervals in asr performance evaluation," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, 2004, pp. I–409.