INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

I Learned Error, I Can Fix It! : A *Detector-Corrector* Structure for ASR Error Calibration

Heui-Yeen Yeen, Min-Ju Kim*, Myoung-Wan Koo*

Sogang University, Korea

yeen214@sogang.ac.kr, mjmjkk0307@sogang.ac.kr, mwkoo@sogang.ac.kr

Abstract

Speech recognition technology has improved recently. However, in the context of spoken language understanding (SLU), containing automatic speech recognition (ASR) errors causes significant downstream performance degradation. To address this issue, various ASR error correction methodologies have been proposed. ASR error correction mainly focuses on correcting and generating only the error span using a conditional decoding method. To this end, we propose a structure with a *Detector* that uses collaborative training to predict various error patterns and a *Corrector* that corrects the detected error span by *Detector*. This pipeline reduces Word Error Rate (WER) and shows less performance degradation in downstream tasks compared with the original ASR hypotheses. In addition, it was shown that it could be generalized to various downstream data. By leveraging this *Detector-Corrector* pipeline, we expect to achieve effective ASR error correction and enable high-quality SLU downstream tasks.

Index Terms: error correction, Spoken language Understanding, intent classification, emotion recognition.

1. Introduction

Pipeline-based deep learning methods for spoken language understanding (SLU) tasks have driven the growth of the spoken dialogue system. In the pipeline-based system, the performance of the Automatic Speech Recognition (ASR) module is essential. This is because errors from the ASR module can propagate to the natural language understanding (NLU) module in the back end, which can eventually lead to performance degradation for SLU downstream tasks. Pre-trained language models used in NLU tasks are trained on the clean text so that receiving ASR output with error tokens at the inference state can cause significant performance degradation[1]. This is because clean text distribution differs from ASR result data. Therefore, ASR error correction is a method to prevent performance degradation by correcting the error of the ASR hypotheses and making it as close to clean text as possible. However, it is a challenging task to eliminate ASR errors, which encourages several methods to prevent downstream performance degradation

Therefore, previous studies address these issues by ASR error adaptive training on the NLU module [2], or correcting errors using speech information(lattice graph, N-best hypotheses) together [3, 4], these methods have shown improved performance for downstream tasks. In addition, a generalized calibration technique that can respond to error patterns for various recognizers is needed, rather than detecting error patterns for specific recognizers.

*equal contribution

Gold truth	she sent me the pages in question
ASR result	she said that the page is in question
	⬇
Detector output	0 1 1 0 1 1 0 0
	⬇
Corrector input	she <extra_id_0> the <extra_id_1> in question
	⬇
Corrector output	she sent me the pages in question

Figure 1: *An example of the calibration process of the ASR error calibrator. ASR result goes into Detector input. In the Detector output, 0 represents a clean span, and 1 represents a span that contains an error. At this time, the span predicted as 1 is masked with the special token of the T5 model. After that, Corrector generates an masked span.*

We develop the idea from previous research and remaining issues for correcting ASR errors with a *Detector-Corrector* structure. The *Detector* comprises a generator and discriminator. The generator generates various error patterns as an error simulator, while the discriminator is trained to predict which part is the ASR error. This allows the discriminator to be generalized to the performance of detecting errors. In actual inference, as shown in figure 1, when the ASR result is input, the span containing the error is masked based on the predicted result from the *Detector*. The masked sentence becomes the input of the *Corrector* again and is generated by correcting only the corresponding span. This not only reduces inference time compared to the methodology of generating the entire text for correcting process but also improves the performance of correction much more by correcting only the part in error[5, 6]. The main contributions of this study are as follows.

- We present a novel ASR error calibrator of the *Detector-Corrector* structure [1]. The experiments show that performance improves by combining the error calibrator proposed in the pipeline SLU task. WER result and downstream result.
- It showed good generalization performance for various corpus and recognition results through the *Detector* structure of the collaborative learning method of simulating errors and detecting errors. In addition, the *Corrector* shortened the inference time by enabling only the error span to be created through the conditional decoding method.

[1] https://github.com/yeonheuiyeon/Detector_Corrector_SLU

10.21437/Interspeech.2023-2475

2. Related work

2.1. ASR correction

The ASR correction task should be corrected appropriately by finding only the span where the error occurred. Therefore, an ASR correction in the existing studies model that combines encoding and decoding processes for spans needs to be modified[7, 8]. [9] shows good performance in Grammar Error Correction, which corrects only the parts corresponding to incorrect grammar. It suggests that the encoder uses a pointer mechanism to predict DELETE and KEEP for each token and reorder. Then, it generates only the parts that the decoder should edit. Similarly, the encoder predicted where the error occurred so that the decoder could generate only this span and produce a final result[10]. In factual error correction, a pipeline method that predicts a specific part to be corrected, then masking and correcting has also been proposed[11, 12]. In this paper, inspired by these methods, we apply a partial decoding method to the span where the error occurred.

2.2. Simulating error for training SLU

However, the above methodology may overfit error patterns for a specific Speech recognizer. Therefore, the method of building simulators that generate errors in various ways and conducting training with augmented corpus by error simulators has been proposed[13, 14, 15]. Still, this method has similar patterns generated according to the simulation conditions and has limitations in extending to various error patterns. To address this problem, we utilized the collaborative training methodology for training the *Detector*. ELECTRA [16] has shown better performance in various tasks than the existing BERT through an adversarial training structure in which the generator fills masked tokens, and the discriminator learns replaced token detection. Therefore, we use this structure similarly to enable collaborative learning for the generator and discriminator. Specifically, the generator simulates an ASR error, and the discriminator distinguishes the token from the generator in which the error occurred. With this structure, the model itself, rather than a specific condition, can generate various error patterns.

3. Approach

We propose *Detector-Corrector* ASR error calibrator, which has advantages for calibrating error in general tasks. The *Detector* consists of a generator and a discriminator following the structure of the ELECTRA model. The generator is trained to reproduce the error by masking the part where the error occurred. It is trained as a speech error pattern simulator, and the discriminator detects error tokens based on the result generated by the generator. After that, using the result of *Detector*, the T5 [17] *Corrector* properly generates only the tokens that need to be modified.

3.1. Preprocessing

At the training, *Corrector* and *Detector* both receive an ASR hypotheses with a masking part where error occurred as input. To produce this input, preprocessing was performed using ground truth sentence GT and ASR sentence X. First, alignment was conducted based on the Levenshtein distance for the ground truth and ASR sentences. We used the standard Levenshtein method proposed by [18]. At this time, substitution, insertion, deletion, and correction are calculated for each word. Therefore, for the tokenized input $X = \{x_0, x_1, x_2, \ldots, x_n\}$, label $L = \{y_0, y_1, y_2, \ldots, y_n\}$ is created. The tokens that need modification is assigned to label 1(substitution, insertion, or deletion), and the token that does not need modification is assigned to label 0. As an input of *Detector* and *Corrector*, Tokens with label 1 are masked.

3.2. *Detector*

The *Detector* consists of generator G and discriminator D. They both have transformer encoder structures and are trained together as the generator simulates error, and the discriminator distinguishes error tokens. The training process of *Detector* is as follows. The input sentence $X = \{x_0, x_1, x_2, \ldots, x_n\}$ are entered to *Detector* with label $L = \{y_0, y_1, y_2, \ldots, y_n\}$. If the value of the t^{th} label of L is 1, it means that the t^{th} token of X is an error token. We masked all error tokens of X with predefined mask special token $X_{md} = \{x_0, [MASK], [MASK], \ldots, x_n\}$ and get contextualized vector representations $h(x) = \{h_1, \ldots, h_m\}$. Next, the generator predicts the probability for a particular token through the softmax layer to generate the token for the masked location. Finally, the generator produce sentence $X_g = \{x_0, \hat{x_1}, \hat{x_2}, \ldots, x_n\}$ which is filled mask token. The generator is trained using a masked language modeling (MLM) method, and the objective function is as follows:

$$\mathcal{L}_{MLM} = \mathbb{E}\left(\sum_{i \in m} -\log p_G(x_i | x^{masked})\right)$$

However, our proposed method differs from the conventional method, as it masks tokens where the error occurred instead of randomly masking a certain percentage of tokens. Also, since the input includes ASR error types from various speech recognizers, the generator learns and generates various error patterns.

The discriminator takes generated sentence X_g as input and is trained to distinguish the error token. The discriminator makes an output through the sigmoid layer, predicting whether each token contains an error or not. Finally, error predicted label $\hat{L} = \{y_0, \hat{y_1}, y_2, \ldots, \hat{y_n}\}$ is produced. The discriminator loss ℓ_{disc} is trained with binary cross-entropy loss, which is the real label is L and the predicted label is $\hat{L}$.

To detect error token efficiently, generator and discriminator jointly train, and the training objective of *Detector* architecture is as shown below:

$$\mathcal{L} = \ell_{MLM} + \lambda \cdot \ell_{disc}$$

At the training stage, the *Detector* is trained to minimize the loss. In inference, only a trained discriminator is used to predict which token contains an error.

3.3. *Corrector*

The *Corrector* is a module that fills mask tokens to correct ASR errors. The original ASR hypotheses input $X = \{x_0, x_1, x_2, \ldots, x_n\}$ with label $L = \{y_0, y_1, y_2, \ldots, y_n\}$ is converted to the masked ASR hypotheses input $X_{mc} = \{x_0, < extra\_id\_0 >, x_2, < extra\_id\_1 >, ...x_k\}$ where error tokens are masked. $< extra\_id\_i > (0 <= i <= n)$ means a predefined span masking special token. One masking token can be filled with 0 to n (n is max_seq_len) tokens, as following conventional T5 training style.

The *Corrector* takes original ASR hypotheses X and masked hypotheses X_{mc}, to refer ASR hypotheses spans when generates calibrated output sentence. In particular, when the

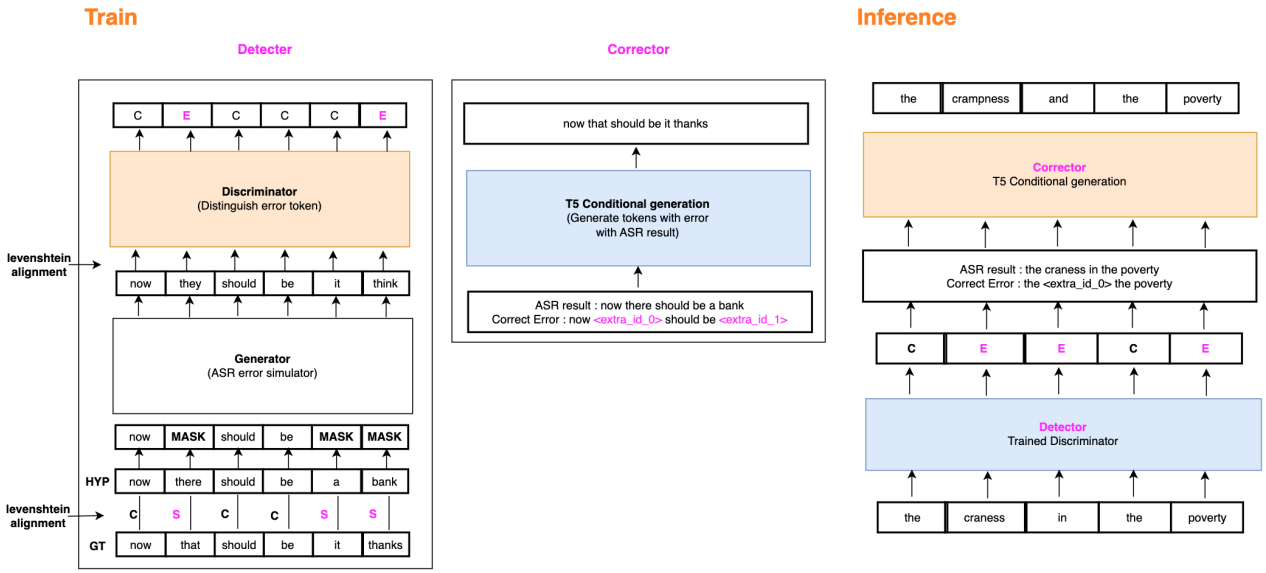

Figure 2: *ASR error calibrator of the Detector-Corrector structure. The figure on the left is the training process of each Detector and Corrector part, and the picture on the right is an example of actual inference using each trained part.*

first span is masked since there is no context, the probability of the decoder generating the correct span is reduced. Then the entered input values are converted to hidden embedding $g(x) = \{g_1, \ldots, g_m\}$.

The decoder learns to generate the masking token of the masked ASR hypotheses in the error-corrected form, the form of ground truth text GT, by referring to the original ASR hypotheses for this hidden representation. The objective uses cross-entropy loss, which is span MLM loss, in the same way as proposed in [17]. This means that the model reconstructs the mask token into consecutive spans. Therefore, the model is trained in the direction where the loss below is minimized.

During inference, the error token predicted by the discriminator is masked and goes into the input. As in the case of learning, it refers to the original hypotheses and fills in the clean form with the ASR error corrected for what the masking span will be. Figure 2 shows the proposed architecture at training and inference stage.

4. Experiment

4.1. Model Training Details

For model training, the *Detector* was initialized with ELECTRA-base [16] and the *Corrector* with T5-base, respectively. We used LibriSpeech(train-clean-100), and ATIS [19] data for training as described in Table 1. LibriSpeech is speech data and consists of relatively long utterances, averaging 34 words. However, ATIS is text data with a shorter utterance, averaging 11 words. We made ATIS speech data using TTS[2] engine as conducted in [10].

To get the ASR hypotheses, we used three types of speech recognizers: conformer [20], google [3], and whisper [21]. The model learned various error patterns from these three speech recognizers using three types of ASR hypotheses. Hence, the final number of training data which consists of three types of ASR output is a total of 97,218 pieces.

Dataset	Train	Valid	Test
Data for calibration			
LibriSpeech	28,539	2,703	2,620
+ATIS train	+3,867		
Downstream task			
SLURP	24,198	4,173	6,281
IEMOCAP	4,416	-	1,065

Table 1: *Statistics of the downstream datasets.*

At the training stage, the *Detector* and *Corrector* were trained separately. The Hyperparameters for training are as follows: The *Detector* was trained with the learning rate 5e-5, optimizer Adam, batch size 32, and max seq length 128. The *Corrector* was trained with the learning rate 1e-4, Adam optimizer[22], batch size 64, max sequence length512, and max target length512. The GPU used for training is NVIDIA RTX A5000 24G.

4.2. Evaluation

4.2.1. Task Description

To evaluate the performance of the proposed methodology, we describe the performance of two tasks: Emotion Recognition and Intent Classification. Table 1 describes statistics for each downstream task dataset.

- Emotion Recognition (ER): The emotion recognition multi-label classification task using the IEmocap speech dataset [23]. This dataset consists of a total of 5 sessions. We used session 1 for the test set and the remaining sessions as a train set.
- Intent Classification (IC): The intent classification multi-label classification task using the SLURP speech dataset [24]. The dataset has different types: recording type in a headset or general environment and TTS synthesis type. We used only the recorded data with a headset.

[2] https://espnet.github.io/espnet/notebook/espnet2_tts_realtime_demo.html

[3] https://www.google.com/intl/en/chrome/demos/speech.html

Data type	IC		ER		Avg Time
	WER	ACC	WER	ACC	
Clean	-	87.2	-	67.4	-
ASR result	30.7	74.9	31.11	65.3	-
Calibrate Method					
T5	24.5	73.5	25.3	62.8	0.15
Const-Decoder	48.2	62.8	26.4	37.3	0.01
Ours	31.1	**77.9**	27.2	**65.4**	0.11

Table 2: *Based on the language model (ELECTRA) pre-trained in clean text, fine-tuning is performed with clean text, ASR result, and corrected text, and performance is measured. The Avg time of the last column is the average of inference time cost in seconds (s) for single data.*

Model	WER	F1 score
Train data		
Google only	6.9	63.1
Unseen data		
A	22.8	88.4
B	23.5	68.9
C	23.2	84.1
Unseen data Avg	23	80

Table 3: *Detector's ASR error detection performance at unseen recognizer learned only from Google ASR results.*

4.2.2. Metrics

Word Error Rate(WER) is used to compare the calibration performance for each speech recognizer. For downstream tasks, we employ an accuracy score. The three best performances were extracted from epochs between 1-10 and averaged. In addition, this process was repeated in three different seeds, and We finally averaged these results and reported them.

4.2.3. Calibrate Method

We compared our proposed method with previous ASR error correction methods.

- T5: It is a T5 model based on the seq2seq structure. The ASR result is learned to correct the output as an input. This model obtains the ASR result as an input, and the output will be a corrected sentence.
- ConstDecoder: This model also consists of an encoder and a decoder. The encoder learns if a token should be kept, deleted, or changed. Next, the decoder does restrict decoding to part of the input sequence embeddings (predicted as the changed tokens)[10].

4.3. Result

Table 2 shows the results of fine-tuning the pre-trained vanilla ELECTRA model on downstream tasks. In the first row, the case of fine-tuning and inference with clean text is reported as the baseline. Also, the second row shows the result of fine-tuning and inference with whisper's ASR hypotheses. In the calibrate method part, we report the results of fine-tuning and inference with corrected hypotheses based on whisper speech recognition results to consider difficulties in getting the golden truth text of audio data in the real world.

Three calibration methods were trained with the same dataset for a fair comparison. Table 2 shows our proposed method achieves high accuracy in terms of downstream tasks compared with other calibrate methods. T5 calibrated, which uses the seq2seq method for calibration, declines WER the most, but shows the lowest accuracy score. In contrast, our proposed method has slightly increased WER in the IC task, but it seems well-corrected errors that are critical contextual parts during fine-tuning. Also, when comparing the inference time, the time was shortened compared to the T5 model despite pipeline structures. In addition, ConstDecoder has significant performance degradation, especially when the downstream corpus are different from the trained corpus. It shows there can be an over-fitting problem on a specific recognizer or corpus.

To confirm whether our proposed model is robust to new error patterns that are unseen in training data, a further experiment was conducted. We trained our model using only google speech recognizer's ASR output of calibration training data and measured the performance of in- and out-of- domain. Table 3 describes the WER and error detecting F1 score of the google ASR hypotheses and three unseen ASR hypotheses. A[4] is an ASR model provided as the ASR baseline in SUPERB. B[5] and C[6] is an ASR model provided by espnet. The results show that even though our proposed method is trained on a limited corpus, it achieves high error detection scores. It means our proposed method can be generalized to the out-of-domain ASR hypotheses. This is because the generator in *Detector* works as an error simulator, and the discriminator can learn various error examples.

5. Conclusions

This paper proposes an ASR error calibrator with a *Detector-Corrector* structure. The *Detector* in the proposed model undergoes collaborative training, with the generator simulating the ASR error pattern and the discriminator distinguishing the error tokens. The *Corrector* is trained to fill the masking span with the corrected form. Since it is a pipeline structure, there can be a limitation on our proposed method that the corrector cannot calibrate errors if the detector does not detect it. However, the experiments showed that our method's novelty, having performance improved. It was confirmed that the ASR WER decreased when the calibrator proposed in this paper was used. In addition, experiments showed that downstream tasks are improved when the proposed calibrator is used for Emotion Recognition and Intent Classification tasks. Although the proposed pipeline structure was trained with limited speech recognizer error patterns and corpus, it showed the generalized performance to out-of-domain. Finally, the proposed calibrator is applied to all pipeline SLU tasks, and the performance can be improved.

6. Acknowledgment

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2022-0-00621,Development of artificial intelligence technology that provides dialog-based multi-modal explainability)

[4] https://github.com/s3prl/s3prl/blob/main/s3prl/downstream/docs/superb.md

[5] transformerASRinhttps://github.com/espnet/espnet_model_zoo

[6] gigaspeechASRinhttps://github.com/espnet/espnet_model_zoo

7. References

[1] P. Zhou, D. Chong, H. Wang, and Q. Zeng, "Calibrate and refine! a novel and agile framework for asr-error robust intent detection," *arXiv preprint arXiv:2205.11008*, 2022.

[2] L. Feng, J. Yu, D. Cai, S. Liu, H.-T. Zheng, and Y. Wang, "Asr-robust spoken language understanding on asr-glue dataset," 2022.

[3] Y. Zou, H. Sun, and Z. Chen, "Associated lattice-bert for spoken language understanding," in *Neural Information Processing: 28th International Conference, ICONIP 2021, Sanur, Bali, Indonesia, December 8–12, 2021, Proceedings, Part VI 28*. Springer, 2021, pp. 579–586.

[4] K. Ganesan, P. Bamdev, A. Venugopal, A. Tushar *et al.*, "N-best asr transformer: Enhancing slu performance using multiple asr hypotheses," *arXiv preprint arXiv:2106.06519*, 2021.

[5] E. Malmi, S. Krause, S. Rothe, D. Mirylenka, and A. Severyn, "Encode, tag, realize: High-precision text editing," *arXiv preprint arXiv:1909.01187*, 2019.

[6] F. Stahlberg and S. Kumar, "Seq2edits: Sequence transduction using span-level edit operations," *arXiv preprint arXiv:2009.11136*, 2020.

[7] Y. Leng, X. Tan, W. Liu, K. Song, R. Wang, X.-Y. Li, T. Qin, E. Lin, and T.-Y. Liu, "Softcorrect: Error correction with soft detection for automatic speech recognition," *arXiv preprint arXiv:2212.01039*, 2022.

[8] Y. Leng, X. Tan, L. Zhu, J. Xu, R. Luo, L. Liu, T. Qin, X. Li, E. Lin, and T.-Y. Liu, "Fastcorrect: Fast error correction with edit alignment for automatic speech recognition," *Advances in Neural Information Processing Systems*, vol. 34, pp. 21 708–21 719, 2021.

[9] J. Mallinson, J. Adamek, E. Malmi, and A. Severyn, "EdiT5: Semi-autoregressive text editing with t5 warm-start," in *Findings of the Association for Computational Linguistics: EMNLP 2022*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 2126–2138. [Online]. Available: https://aclanthology.org/2022.findings-emnlp.156

[10] J. Yang, R. Li, and W. Peng, "Asr error correction with constrained decoding on operation prediction," *arXiv preprint arXiv:2208.04641*, 2022.

[11] J. Thorne and A. Vlachos, "Evidence-based factual error correction," *arXiv preprint arXiv:2012.15788*, 2020.

[12] D. Shah, T. Schuster, and R. Barzilay, "Automatic fact-guided sentence modification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 8791–8798.

[13] E. Simonnet, S. Ghannay, N. Camelin, and Y. Estève, "Simulating ASR errors for training SLU systems," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), May 2018. [Online]. Available: https://aclanthology.org/L18-1499

[14] J. Liu, R. Takanobu, J. Wen, D. Wan, H. Li, W. Nie, C. Li, W. Peng, and M. Huang, "Robustness testing of language understanding in task-oriented dialog," *arXiv preprint arXiv:2012.15262*, 2020.

[15] S. Dutta, S. Jain, A. Maheshwari, G. Ramakrishnan, and P. Jyothi, "Error correction in asr using sequence-to-sequence models," *arXiv preprint arXiv:2202.01157*, 2022.

[16] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," *arXiv preprint arXiv:2003.10555*, 2020.

[17] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[18] N. Ruiz and M. Federico, "Phonetically-oriented word error alignment for speech recognition error analysis in speech translation," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, Dec 2015, pp. 296–302.

[19] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[20] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, 2020.

[21] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," *arXiv preprint arXiv:2212.04356*, 2022.

[22] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[23] A. Zadeh, P. P. Liang, S. Poria, P. Vij, E. Cambria, and L.-P. Morency, "Multi-attention recurrent network for human communication comprehension," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[24] E. Bastianelli, A. Vanzo, P. Swietojanski, and V. Rieser, "Slurp: A spoken language understanding resource package," *arXiv preprint arXiv:2011.13205*, 2020.