Lightweight Audio Segmentation for Long-form Speech Translation

Jaesong Lee[1*], *Soyoon Kim*[1,2*], *Hanbyul Kim*[1], *Joon Son Chung*[2]

[1]NAVER Cloud, South Korea
[2]KAIST, South Korea

{jaesong.lee, soyoon.kim, hanbyul.kim}@navercorp.com, joonson@kaist.ac.kr

Abstract

Speech segmentation is an essential part of speech translation (ST) systems in real-world scenarios. Since most ST models are designed to process speech segments, long-form audio must be partitioned into shorter segments before translation. Recently, data-driven approaches for the speech segmentation task have been developed. Although these approaches improve overall translation quality, a performance gap exists due to a mismatch between the models and ST systems. In addition, the prior works require large self-supervised speech models, which consume significant computational resources. In this work, we propose a segmentation model that achieves better speech translation quality with a small model size. We propose an ASR-with-punctuation task as an effective pre-training strategy for the segmentation model. We also show that proper integration of the speech segmentation model into the underlying ST system is critical to improve overall translation quality at inference time.

Index Terms: speech translation, audio segmentation

1. Introduction

Speech translation (ST), which converts speech signals from one language into text in another language, helps facilitate communication between people who speak different languages and helps overcome language barriers. Integrating an automatic speech recognition (ASR) component with a machine translation (MT) component is commonly referred to as a cascaded architecture, and it is the traditional and common approach for the ST task [1].

Recently, there has been a growing interest in end-to-end (E2E) methods that directly translate spoken source language to target language text using a single sequence-to-sequence model [2, 3]. Since E2E ST doesn't produce intermediate speech recognition results, it can prevent ASR errors from propagating to the translation model. It can also improve latency and model size because it combines the ASR and MT modules into a single model for inference. However, this approach is still less accurate than the cascade system [4, 5].

Although both cascade and E2E ST systems have been actively developed, the models are designed to process segmented speech due to constraints on model architecture and training conditions. Long-form speech must be segmented in advance to use the ST system in real-world scenarios where segmentation is not available. However, until recently, it has been underexplored how the segmentation impacts the overall quality of the ST system.

If a segmentation method is not *matched* to the underlying ST system, it could lead to low-quality translation results [6]. In [7], two common failure modes due to the mismatch are discussed. When a segment is too long or contains multiple sentences, the translation may omit some of the input, called a deletion error. On the other hand, if a segment is too short or does not contain a proper sentence, the translation may contain phrases not in the input, referred to as an insertion error or hallucination [8, 9]. Thus, it is essential to produce segments of appropriate duration and with a single complete sentence to meet the requirements of the underlying ST system.

Several segmentation methods for ST have been previously introduced in the literature [10, 11, 12, 13]. Pause-based segmentation using voice activity detection (VAD) is commonly employed as a preliminary step for ST systems [10, 11, 12]. Another widely used strategy involves length-based segmentation techniques, where speech is divided into segments according to heuristic principles [12, 13]. For cascaded speech translation systems, there are works on re-segmentation of ASR output text [14, 15]. Also, it is proposed to interpret predictions of ASR and ST models for fixed-size chunks as segmentation [16, 17, 18, 19].

Recently, data-driven approaches for audio segmentation have been proposed [20, 21, 22], which consist of a neural network encoder and predict segmentation at frame level. The methods have been shown to improve segmentation performance compared to the traditional methods. However, the translation quality of the proposed methods is still behind the one of oracle segmentation, as a mismatch exists between the two segmentation results [6]. Also, the models are usually based on large self-supervised models, such as wav2vec 2.0 [23], whose computational cost would be a hurdle for deploying ST system on mobile devices.

In this paper, we aim to improve the end-to-end segmentation modeling for long-form speech translation while significantly reducing the number of model parameters. We propose that the ASR-with-punctuation task [24, 25], the joint task of speech recognition and punctuation prediction, is an effective pre-training task for the segmentation model. In addition, we show that tuning the segmentation model at inference time is essential to the overall translation quality, and provide an analysis of the mismatch between the segmentation model and ST system.

Our contributions are as follows:

- We propose a pre-training strategy for the segmentation model using the ASR-with-punctuation task and show that the proposed pre-training strategy improves the segmentation accuracy and the final translation quality.
- We show that reducing the mismatch between the segmentation model and ST systems is crucial, due to varying charac-

* Equal contribution.

teristics among ST systems.

- We demonstrate that the proposed segmentation model achieves better translation quality than the prior methods, and its size is 8% to 14% smaller than that of the previous works.

2. Architecture

We formulate the audio segmentation task as a frame-level classification task, following previous works [20, 21, 22]. The segmentation model is trained to predict a frame-level label sequence for a fixed-length audio input. The label sequence $(l_1, \cdots, l_T)$ consists of the binary label $l_t \in \{0, 1\}$, which indicates whether the t-th frame is a part of segment ($l_t = 1$) or not ($l_t = 0$).

The model consists of an encoder layer and a linear layer. First, the model converts a sequence of acoustic features to a sequence of output vectors $(\mathbf{e}_1, \cdots, \mathbf{e}_T)$, where $\mathbf{e}_t \in \mathbb{R}^D$ represents the t-th output vector. Conformer-M [26] architecture is used for the encoder layer, which accepts log-mel acoustic features and gives the output sequence with a 40ms stride.

The output vector $\mathbf{e}_t$ transformed to a probability p_t by a linear layer. The value p_t indicates the probability that t-th frame is a part of the segment, and it is computed as:

$$p_t = \mathsf{sigmoid}(\mathsf{Linear}_{D \to 1}(\mathbf{e}_t)).$$

During training, the cross-entropy loss is used to match p_t to the segmentation label l_t. The length of the input audio is 20 seconds following prior works [20, 22].

Note that the model size is much smaller than the those of previous works. SHAS [20] has 201M parameters and SHAS-FTPT [22] has 349M parameters, as they are based on XLS-R [27], a large self-supervised wav2vec 2.0 [23] model. On the other hand, our model has 27.3M parameters, which are only **14%** of SHAS and **8%** of SHAS-FTPT. Consequently, the model is more suitable for lightweight applications including streaming and on-device scenarios.

2.1. Inference

At inference time, the long-form audio is partitioned into fixed-size chunks of 20 seconds, with 2-second of overlap. For overlapped frames, the two probability values are averaged. Then, the output probability sequence is processed into a list of segments.

For the processing, SHAS [20] introduces pDAC (probabilistic Divide-And-Conquer), which recursively splits a large segment into smaller segments. pDAC has two hyper-parameters `minlen` and `maxlen` so that the resulting segment is always longer than `minlen` and shorter than `maxlen`. pDAC has a drawback in that it often produces segments longer than oracle segments [22] because it tends not to split a segment shorter than `maxlen`.

SHAS-FTPT [22] proposes pTHR, a threshold-based algorithm, which also ensures the length of the segment is between `minlen` and `maxlen`. pTHR has a drawback in that if the predicted segment is longer than `maxlen`, it is split into fixed-size segments of `maxlen`.

To this end, we use a simple algorithm as follows:

- The predicted probability p_t is converted to a binary label $l_t = \mathbb{I}[p_t > 0.5]$ and consecutive positive labels form a segment.
- Segments shorter than `minlen` are discarded.
- Segments longer than `maxlen` are split at $\hat{t}$-th position, where $\hat{t} = \mathsf{argmin}_t(p_t)$.

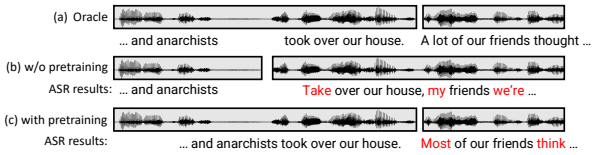

Figure 1: *(a) Oracle segmentation and its corresponding reference text. (b) prediction of segmentation model* ***without*** *pre-training, and its corresponding ASR results. (c) prediction of segmentation model* ***with*** *pre-training, and its corresponding ASR results. ASR errors are colored red. See Section 3 for details.*

- Following the previous methods [20, 22], we expanded each segment by 0.06 seconds.

We found that `maxlen` is an important hyper-parameter that should be tuned for integrating the segmentation model and underlying ST system. See Section 4 for discussion and Section 5 for experiments.

3. Pre-training via ASR-with-punctuation

The ASR-with-punctuation task aims to recognize text and predict punctuation at the same time [28, 24, 25, 13]. In particular, recent works [24, 25] showed that ASR models based on Connectionist Temporal Classification (CTC) [29] are suitable for ASR-with-punctuation with high accuracy.

CTC has a characteristic that the model predicts a text label for each frame (including a special label `<blank>`, which indicates no label corresponds to the frame), and the position of the predictive label is well-aligned to the corresponding speech utterance [30]. Thus, the behavior of the CTC model is closely related to the segmentation task.

We argue that both the ASR-with-punctuation task and the sentence-level segmentation task require understanding the grammar of sentences to certain degree. Some punctuation marks, including period ⟨`.`⟩ and question mark ⟨`?`⟩, indicate the end of the sentence. To predict them, the CTC model needs to understand where the sentence boundaries are. Also, the frame that predicts such marks is likely to be the position in which the segment of the sentence ends.

On the other hand, a spoken sentence may contain long pauses between utterances. In this case, the CTC model should not emit end-of-sentence marks during pauses, and the segmentation model should not partition the sentence at pauses as well.

Therefore, we expect that the CTC model with punctuation prediction learns features related to sentence structure, which are also helpful for sentence-level segmentation. To this end, we propose to pre-train the encoder of the segmentation model with punctuation CTC task.

Following [25], we concatenate two segments of the ASR corpus at training time. This prevents the ASR model from predicting period symbol ⟨`.`⟩ at the end regardless of input, as the symbol is placed at the end of the sentence in many ASR corpus. We apply Intermediate CTC [31] following previous works on ASR-with-punctuation [24, 25].

Figure 1 shows two segmentation results with their corresponding ASR transcriptions, one from the model without pre-training and the other with pre-training. Without pre-training, the model relies on pauses for segmentation, causing mis-segmentation around the phrase "took over our house". This also leads to a critical mis-transcription ("*Take* over our house"), as the phrase is not a proper sentence, while the ASR model

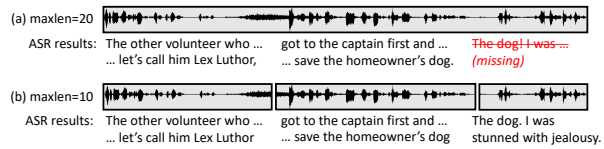

Figure 2: *Segmentation and corresponding ASR results with two different* `maxlen` *configurations. Note that the two results are inferred from the same segmentation model. See Section 4 for details.*

is likely to be trained with complete sentences. More importantly, incomplete sentences from the mis-segmentation are more prone to mis-translation due to missing information [7].

On the other hand, the pre-trained model does not rely only on long pauses. It successfully predicts the sentence boundary, which also leads to fewer transcription errors in the ASR model. This illustrates the segmentation ability based on the speech content, not only the acoustic statistics. We show experimental results that pre-training improves the overall translation quality in Section 5.

4. Integration to speech translation system

Due to the variety in model architectures and training conditions, ST systems require specific conditions of input speech segment for high-quality translation. Importantly, if the audio segmentation is *mismatched* to the segmentation used to train the ST system, its translation quality may be significantly degraded [7, 6].

There are two well-known failure cases due to the mismatch. If the speech segment input is too long or contains too many sentences, the ST model may fail to translate and drop a significant part of the input, causing *deletion* errors [7]. On the other hand, if the segment is too short, it may not contain a complete sentence, which leads to significant *addition* errors, also called as *hallucination* [9, 8, 7]. Also, the exact notation of sentence boundary varies over translation corpora and target domains, contributing to the mismatch problem [7, 32, 33, 34]. Therefore, to prevent such failure modes, it is essential to match the segmentation model and ST model so that the segmentation model produces speech segments that the ST model can handle well.

At the inference time, there is a hyper-parameter `maxlen` regarding the maximum length of the speech segment, as described in Section 2.1. The hyper-parameter can be tuned to reduce the mismatch between the segmentation model and the ST system. If the segment produced by the segmentation model is longer than `maxlen`, it is partitioned into smaller segments. We found that the partitioned segments tend to contain near-complete sentences rather than incomplete phrases. This is because the pre-training task proposed in Section 3 improves the understanding of sentence-level boundaries and prevents non-linguistic splits at long pauses.

Figure 2 shows an example of segmentation configuration and corresponding ASR results, where the ASR model in use tends to require short segments for better recognition accuracy. The two results are inferred from the same segmentation model, except that the first setting (a) uses `maxlen` 20 and the second setting (b) uses `maxlen` 10. The setting (a) yields a long segment that matches the oracle segmentation. However, the ASR model is not able to handle such long input, causing deletion errors towards the end of the segment.

On the other hand, the setting (b) forces the segmentation model to yield short segments. As a result, the audio is split into three segments, and the third segment still contains a complete sentence. The first two segments contain incomplete phrases, as it is impossible to split the sentence. Nonetheless, setting (b) gives a better translation overall, as the last sentence is correctly recognized.

Note that the example is specific to the ASR model in use – when the other ST model is used, the ST model successfully produces high-quality translation results for the long segment. We show experimental results with various ST systems in Section 5.

5. Experiments

MuST-C [35] is a multilingual speech corpus that can be used for automatic speech recognition (ASR), speech translation (ST), and audio segmentation tasks. It consists of long-form speech of English TED Talks, sentence-level segmentation labels, transcription (with punctuation) and translation for each segment.

For the segmentation task, we use two language pairs of MuST-C v2, English-German (En-De) and English-Japanese (En-Ja). For the pre-training task in Section 3, we use the English transcription of En-De pairs.

We evaluate the segmentation models on the En-De and En-Ja ST tasks. For the En-De ST task, we employ two ST systems. The first is the Fairseq-ST[1] [36] E2E ST model, which has also been used in prior works [20, 22]. The second is SeamlessM4T-v2[2] [37]. The model supports English ASR, En-De MT, and En-De E2E ST. We employ the ASR and MT models for the cascaded ST system, as its translation quality is significantly better than the one of the E2E ST model, and it is possible to measure the accuracy of the ASR task as well as the ST task.

For the En-Ja ST task, we employ SeamlessM4T-v2 for the ASR model and employ two En-Ja MT models, SeamlessM4T-v2 and JParaCrawl[3] [38], for cascaded ST.

For evaluating the ST system for long-form speech, mwerSegmenter [39] is used to re-align the results of ASR and ST models to the reference text. Then, sacreBLEU [40] is used to measure BLEU scores [41]. For cascaded ST, we also measure word error rates (WERs) of the ASR model.

We compare our segmentation model to SHAS[4] [20] (En-De and En-Ja) and SHAS-FTPT[5] [22] (En-De) using the model parameters released by the authors.

For the choice of `maxlen`, we evaluate the segmentation model for two ST systems using 8, 10, 15, 20, and 30 seconds and report the best result for each ST system. For SHAS and SHAS-FTPT, we also evaluate the models with `maxlen` reported in the papers. For `minlen`, we use 0.2 seconds following SHAS and SHAS-FTPT.

5.1. Results

Table 1 shows the evaluation results for the En-De task. We found our model consistently outperforms the baseline models, whereas the number of parameters in our model (27.3M)

[1] `https://github.com/facebookresearch/fairseq/tree/main/examples/speech_text_joint_to_text`, En-De MuST-C model

[2] `https://github.com/facebookresearch/seamless_communication`, seamlessM4T_v2_large model

[3] En-Ja big model

[4] `https://github.com/mt-upc/SHAS`

[5] `https://github.com/ahclab/Wav2VecSegmenter`

Table 1: *Results of MuST-C En-De speech translation.*

Segmentation	#param	max len	Fairseq-ST BLEU↑	SeamlessM4T-v2 WER%↓	SeamlessM4T-v2 BLEU↑
Oracle			26.90	14.58	29.49
SHAS	201M	10	24.99	14.70	27.89
		20	25.57	23.61	25.09
SHAS-FTPT	349M	15	25.95	17.00	28.13
		28	26.30	17.98	27.70
Proposed	**27.3M**	10	25.78	**12.85**	**28.80**
		20	**26.66**	15.51	28.45

Table 2: *Results of MuST-C En-Ja speech translation. "JParaCrawl" indicates the cascade ST system of SesmlessM4T-v2 ASR and JParaCrawl MT models.*

Segmentation	#param	max len	SeamlessM4T-v2 WER%↓	SeamlessM4T-v2 BLEU↑	JParaCrawl BLEU↑
Oracle			12.91	10.16	11.91
SHAS	201M	8	13.56	9.03	11.34
		18	21.90	8.44	11.59
Proposed	**27.3M**	10	**12.45**	**10.08**	11.65
		20	14.49	9.70	**11.83**

is much smaller than the baseline models (201M and 349M).

We found Fairseq-ST and SeamlessM4T-v2 require different `maxlen` for the best accuracy – `maxlen` = 20 for Fairseq and `maxlen` = 10 for SeamlessM4T-v2. Figure 3 shows En-De BLEU scores for various `maxlen`. It shows that the BLEU score decreases by 1.5 BLEU points if `maxlen` is not tuned properly.

Table 2 shows the evaluation results for the En-Ja task. Similar to En-De, our model performs consistently better than the baseline model, and SeamlessM4T-v2 MT yields the best BLEU for shorter `maxlen` while JParaCrawl MT does for longer `maxlen`.

For the SeamlessM4T-v2 ASR task, we found shorter `maxlen` leads to lower WER. Note that the WER of `maxlen` = 10 is even lower than the WER of oracle segmentation for both En-De and En-Ja. This is because the oracle segmentation contains long segments, causing deletion errors on SeamlessM4T-v2. When `maxlen` = 8, we obtained the lowest WERs of **12.35** for En-De and **11.70** for En-Ja.

However, for the following MT task, short `maxlen` and low WER do not necessarily improve the BLEU score. We found the best BLEU scores are obtained from `maxlen` = 10 for the SeamlessM4T-v2 MT model and `maxlen` = 20 for the JParaCrawl MT model. This could be explained by the fact that some non-critical ASR errors (e.g., "went and met" becomes "went to meet") do not necessarily cause fatal translation errors, and MT models generally require long input for better translation.

The overall results highlight that each ST system has different requirements for the translation quality, and for cascaded ST, it depends on the both the ASR and MT models. Therefore, for reliable evaluation of segmentation models, employing multiple ST systems for measurement is important.

For the effectiveness of ASR pre-training in Section 3, we measure BLEU scores of two segmentation models, with or without ASR pre-training. The result is shown in Figure 3. We see the ASR pre-training consistently improves BLEU on the two ST systems.

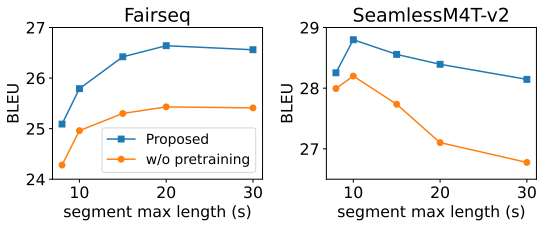

Figure 3: *En-De BLEU scores for various* `maxlen`.

Table 3: *Punctuation F1 scores of SeamlessM4T-v2 ASR.*

Segmentation	#param	Punctuation F1 ↑ (%) avg.	⟨.⟩	⟨?⟩	⟨,⟩
Oracle		75.60	84.43	76.06	66.31
SHAS	201M	65.15	69.59	66.82	59.03
SHAS-FTPT	349M	65.50	71.59	66.67	58.23
Proposed	**27.3M**	**67.79**	**73.62**	**69.91**	**59.84**

5.2. Evaluation of ASR punctuation prediction

For the ASR-with-punctuation task, wrong audio segmentation can lead to incorrect punctuation mark predictions, as the punctuation mark is difficult to predict correctly if the segment does not contain a complete sentence. For example, Figure 1 (a) shows that the ASR model replaces a period ⟨.⟩ with a comma ⟨,⟩ for incorrect segmentation.

To this end, we measure F1 scores of three punctuation marks, period ⟨.⟩, question mark ⟨?⟩, and comma ⟨,⟩ of SeamlessM4T-v2 ASR results from various segmentation models used in Section 5.1. Following prior works on ASR-with-punctuation [24, 25], the output of the ASR model is aligned to the reference text for measurement.

Table 3 shows the punctuation F1 scores of ASR results for the segmentation methods. It shows that our model consistently outperforms baseline models for three punctuation marks, implying the ability to understand sentence structure. This, in turn, improves the ST performance.

6. Conclusion

We propose a lightweight end-to-end audio segmentation modeling for improving the quality of long-form speech translation. We propose to use ASR-with-punctuation as a pre-training task for audio segmentation and show experimental improvements. We emphasize the need for the match between the segmentation model and the speech translation system to achieve optimal translation quality, and show that tuning the inference-time hyper-parameter reduces the mismatch problem. Furthermore, we show the proposed segmentation model achieves better translation quality with a model size of only 8% to 14% of the baseline models.

7. References

[1] H. Ney, "Speech translation: Coupling of recognition and translation," in *Proc. ICASSP*, 1999.

[2] A. Berard, O. Pietquin, C. Servan, and L. Besacier, "Listen and translate: A proof of concept for end-to-end speech-to-text translation," 2016.

[3] R. J. Weiss *et al.*, "Sequence-to-Sequence Models Can Directly Translate Foreign Speech," in *Proc. Interspeech*, 2017.

[4] M. Sperber and M. Paulik, "Speech translation and the end-to-end promise: Taking stock of where we are," in *Proc. ACL*, 2020.

[5] M. Agarwal *et al.*, "Findings of the IWSLT 2023 evaluation campaign," in *Proc. IWSLT*, 2023.

[6] E. Salesky *et al.*, "Evaluating multilingual speech translation under realistic conditions with resegmentation and terminology," in *Proc. IWSLT*, 2023.

[7] R. Wicks and M. Post, "Does sentence segmentation matter for machine translation?" in *Proc. WMT*, 2022.

[8] V. Raunak, A. Menezes, and M. Junczys-Dowmunt, "The curious case of hallucinations in neural machine translation," in *Proc. NAACL-HLT*, 2021.

[9] K. Lee *et al.*, "Hallucinations in neural machine translation," 2018.

[10] T. Potapczyk and P. Przybysz, "SRPOL's system for the IWSLT 2020 end-to-end speech translation task," in *Proc. IWSLT*, 2020.

[11] M. Gaido, M. Negri, M. Cettolo, and M. Turchi, "Beyond voice activity detection: Hybrid audio segmentation for direct speech translation," in *Proc. ICNLSP*, 2021.

[12] G. I. Gállego *et al.*, "End-to-end speech translation with pretrained models and adapters: UPC at IWSLT 2021," in *Pro. IWSLT*, 2021.

[13] A. Radford *et al.*, "Robust speech recognition via large-scale weak supervision," in *Proc. ICML*, 2023.

[14] E. Cho, J. Niehues, and A. Waibel, "Segmentation and punctuation prediction in speech language translation using a monolingual translation system," in *Proc. IWSLT*, 2012.

[15] D. Wan *et al.*, "Segmenting subtitles for correcting ASR segmentation errors," in *Proc. EACL*, 2021.

[16] T. Yoshimura, T. Hayashi, K. Takeda, and S. Watanabe, "End-to-end automatic speech recognition integrated with CTC-based voice activity detection," in *Proc. ICASSP*, 2020.

[17] W. R. Huang *et al.*, "E2E segmentation in a two-pass cascaded encoder ASR model," in *Proc. ICASSP*, 2023.

[18] P. Polák and O. Bojar, "Long-form end-to-end speech translation via latent alignment segmentation," 2023.

[19] Y. Shu *et al.*, "A CIF-based speech segmentation method for streaming E2E ASR," *IEEE Signal Processing Letters*, 2023.

[20] I. Tsiamas, G. I. Gállego, J. A. R. Fonollosa, and M. R. Costa-jussà, "SHAS: Approaching optimal Segmentation for End-to-End Speech Translation," in *Proc. Interspeech*, 2022.

[21] R. Fukuda, K. Sudoh, and S. Nakamura, "Speech Segmentation Optimization using Segmented Bilingual Speech Corpus for End-to-end Speech Translation," in *Proc. Interspeech*, 2022.

[22] ——, "Improving speech translation accuracy and time efficiency with fine-tuned wav2vec 2.0-based speech segmentation," *TASLP*, 2024.

[23] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Proc. NeurIPS*, 2020.

[24] J. Nozaki, T. Kawahara, K. Ishizuka, and T. Hashimoto, "End-to-end Speech-to-Punctuated-Text Recognition," in *Proc. Interspeech*, 2022.

[25] H. Kim, S. Seo, L. Lee, and S. Baek, "Improved Training for End-to-End Streaming Automatic Speech Recognition Model with Punctuation," in *Proc. Interspeech*, 2023.

[26] A. Gulati *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. Interspeech*, 2020.

[27] A. Babu *et al.*, "XLS-R: Self-supervised cross-lingual speech representation learning at scale," 2021.

[28] M. Mimura, S. Sakai, and T. Kawahara, "An end-to-end model from speech to clean transcript for parliamentary meetings," in *Proc. APSIPA*, 2021.

[29] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006.

[30] L. Kürzinger *et al.*, "CTC-segmentation of large corpora for german end-to-end speech recognition," in *International Conference on Speech and Computer*, 2020.

[31] J. Lee and S. Watanabe, "Intermediate loss regularization for ctc-based speech recognition," in *Proc. ICASSP*, 2021.

[32] B. Minixhofer *et al.*, "Where's the point? self-supervised multilingual punctuation-agnostic sentence segmentation," in *Proc. ACL*, 2023.

[33] E. Matusov, P. Wilken, and Y. Georgakopoulou, "Customizing neural machine translation for subtitling," in *Proc. WMT*.

[34] I. Tsiamas, J. Fonollosa, and M. Costa-jussà, "SegAugment: Maximizing the utility of speech translation data with segmentation-based augmentations," in *Proc. EMNLP 2023*, 2023.

[35] M. A. Di Gangi *et al.*, "MuST-C: a Multilingual Speech Translation Corpus," in *Proc. NAACL-HLT*, 2019.

[36] M. Ott *et al.*, "fairseq: A fast, extensible toolkit for sequence modeling," in *Proc. ACL (Demonstrations)*, 2019.

[37] L. Barrault *et al.*, "Seamless: Multilingual expressive and streaming speech translation," 2023.

[38] M. Morishita, J. Suzuki, and M. Nagata, "JParaCrawl: A large scale web-based English-Japanese parallel corpus," in *Proc. LREC*, 2020.

[39] E. Matusov, G. Leusch, O. Bender, and H. Ney, "Evaluating machine translation output with automatic sentence segmentation," in *Proc. IWSLT*, 2005.

[40] M. Post, "A call for clarity in reporting BLEU scores," in *Proc. WMT*, 2018.

[41] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proc. ACL*, 2002.