Implicit Memory Transformer for Computationally Efficient Simultaneous Speech Translation

Matthew Raffel and **Lizhong Chen**
Oregon State University, USA
`{raffelm, chenliz}@oregonstate.edu`

Abstract

Simultaneous speech translation is an essential communication task difficult for humans whereby a translation is generated concurrently with oncoming speech inputs. For such a streaming task, transformers using block processing to break an input sequence into segments have achieved state-of-the-art performance at a reduced cost. Current methods to allow information to propagate across segments, including left context and memory banks, have faltered as they are both insufficient representations and unnecessarily expensive to compute. In this paper, we propose an *Implicit Memory Transformer* that implicitly retains memory through a new left context method, removing the need to explicitly represent memory with memory banks. We generate the left context from the attention output of the previous segment and include it in the keys and values of the current segment's attention calculation. Experiments on the MuST-C dataset show that the Implicit Memory Transformer provides a substantial speedup on the encoder forward pass with nearly identical translation quality when compared with the state-of-the-art approach that employs both left context and memory banks.

1 Introduction

Simultaneous speech translation (SimulST) refers to the process of producing an output translation concurrently with an oncoming source speech input. For humans, performing accurate SimulST is extremely difficult and becomes nearly impossible to perform over long periods of time. Given the potential broad applications of SimulST in industry and government sectors, there is a strong need for machine learning models to perform the task to a level above the capabilities of humans.

One branch of machine learning models that have been effective in SimulST is transformers (Vaswani et al., 2017) using block processing, a process that breaks an input sequence into segments which the encoder processes sequentially and individually (Dong et al., 2019). As later segments may lose earlier information in a sentence (i.e., context fragmentation), techniques known as left context and memory banks have been introduced. The concept of left context was idealized with the Transformer-XL (Dai et al., 2019), a model optimized for language modeling, which was later adapted for streaming automatic speech recognition (ASR). The Transformer-XL generated left context by saving the previous segment to each encoder layer, so the subsequent segment could include it in the attention calculation at the same encoder layer. Memory banks were later introduced in the self-attention calculation of the Augmented Memory Transformer (Wu et al., 2020), allowing it to outperform the Transformer-XL in streaming ASR and also be state-of-the-art in SimulST (Ma et al., 2021). These memory banks were token summarizations of previous segments and helped retain explicit long-term dependencies. The Augmented Memory Transformer also included the left context alongside the center (main) segment tokens with an additional right context, all of which add computational cost. We argue that the methods to generate and use left context and/or memory banks in both the Transformer-XL and Augmented Memory Transformer are naive, costing both models' performance at a given computational budget.

In this paper, we propose a computationally efficient architecture, the *Implicit Memory Transformer*, that implicitly retains memory through a novel left context generation method, thereby removing the need for memory banks entirely. Briefly, the proposed left context method for a given encoder layer leverages the previous segment's attention output in the attention calculation of the current segment. Our method for calculating left context is broadly applicable to any transformer model that utilizes block processing. The proposed Implicit Memory Transformer is more computationally efficient than the Augmented Memory Transformer, reducing the cost of self-attention calculation, convolution layers, and feed-forward layers.

We conduct our experiments on the English-German, English-French, and English-Spanish language pairs of the MuST-C dataset (Cattoni et al., 2021) and demonstrate a significant speedup over the Augmented Memory Transformer for the forward pass of the encoder, with no reduction in the translation quality across all wait-k values.

2 Background and Related Works

Augmented Memory Transformer: For SimulST, a transformer model waits for k token chunks before beginning translation, a policy referred to as wait-k (Ma et al., 2018). One such transformer that uses this wait-k policy is the Augmented Memory Transformer (Ma et al., 2021). The Augmented Memory Transformer breaks an input sequence into segments $\boldsymbol{S}_n^i \in \mathbb{R}^{s\times d}$, where n denotes the segment position in the sequence and i denotes the layer index in the Augmented Memory Transformer. Each segment is composed of a left context $\boldsymbol{L}_n^i \in \mathbb{R}^{l\times d}$ of size l, a center context $\boldsymbol{C}_n^i \in \mathbb{R}^{c\times d}$ of size c, and a right context $\boldsymbol{R}_n^i \in \mathbb{R}^{r\times d}$ of size r. Each segment is of size $s = l + c + r$ and overlaps with the previous and subsequent segments with the left and right context. Unlike the default transformer, the encoder of the Augmented Memory Transformer possesses two subsampling convolution layers to reduce the size of the segment inputs.

In the self-attention calculation for the encoder, memory banks, $\boldsymbol{M}_n^i \in \mathbb{R}^{N\times d}$, are added to the keys and values where N denotes the maximum number of memory banks for a given layer. Each layer's memory banks summarize the previous segments and are theorized to allow the model to retain explicit long-term memory. Each memory bank is created using the attention output of a summarization query, $\sigma_n^i \in \mathbb{R}^{1\times d}$, included in the attention calculation. This summarization query is calculated by averaging the tokens in the current segment. For any given layer, the queries, keys, and values can be represented by the following equations:

$$\boldsymbol{Q}_n^i = \boldsymbol{W}_q^i[\boldsymbol{L}_n^i, \boldsymbol{C}_n^i, \boldsymbol{R}_n^i, \sigma_n^i] \tag{1}$$

$$\boldsymbol{K}_n^i = \boldsymbol{W}_k^i[\boldsymbol{M}_n^i, \boldsymbol{L}_n^i, \boldsymbol{C}_n^i, \boldsymbol{R}_n^i] \tag{2}$$

$$\boldsymbol{V}_n^i = \boldsymbol{W}_v^i[\boldsymbol{M}_n^i, \boldsymbol{L}_n^i, \boldsymbol{C}_n^i, \boldsymbol{R}_n^i] \tag{3}$$

In each of the equations, $\boldsymbol{W}_q^i$, $\boldsymbol{W}_k^i$, and $\boldsymbol{W}_v^i$ are the query, key, and value projection matrices for layer i. The [.] operator concatenates $\boldsymbol{L}_n^i$, $\boldsymbol{C}_n^i$, $\boldsymbol{R}_n^i$ with σ_n^i or $\boldsymbol{M}_n^i$. After the encoder processes each individual segment, they are concatenated before being provided to a simultaneous decoder (Ma et al., 2020b).

Average Lagging: Average Lagging is one prominent metric to determine the efficacy of a SimulST model (Ma et al., 2018). It denotes in milliseconds the lag between the output translation and the input source sequence (Ma et al., 2020b).

BLEU Score: An equally important metric to evaluate a SimulST model is the BLEU score, which measures the translation similarity between the predicted output and the target output. The BLEU score ranges from 0 to 1 and is often represented with percentages (Papineni et al., 2002).

3 Methods

3.1 Implicit Memory Transformer

We propose an Implicit Memory Transformer that leverages a new left context generation method to retain an implicit memory of previous segments. As such, we are able to remove the explicit memory provided by the memory banks that are expensive to compute in the Augmented Memory Transformer. Our new implicit memory left context is unique at each layer of the encoder, whereby it is composed of a portion of the output from the self-attention calculation of the previous segment's center context.

Specifically, suppose our implicit memory left context is denoted as $\boldsymbol{Z}_n^i \in \mathbb{R}^{l\times d}$. Then, in the self-attention calculation of the Implicit Memory Transformer, the queries, keys, and values for each layer's attention calculation can be calculated as follows:

$$\boldsymbol{Q}_n^i = \boldsymbol{W}_q^i[\boldsymbol{C}_n^i, \boldsymbol{R}_n^i] \tag{4}$$

$$\boldsymbol{K}_n^i = \boldsymbol{W}_k^i[\boldsymbol{Z}_n^i, \boldsymbol{C}_n^i, \boldsymbol{R}_n^i] \tag{5}$$

$$\boldsymbol{V}_n^i = \boldsymbol{W}_v^i[\boldsymbol{Z}_n^i, \boldsymbol{C}_n^i, \boldsymbol{R}_n^i] \tag{6}$$

In comparison with the calculation of the queries, keys, and values of the current state-of-the-art Augmented Memory Transformer shown in Equation 1, 2 and 3, our Implicit Memory Transformer has three notable differences consisting of:

1) **Removed memory banks**: The memory bank terms in Equation 2 and 3 not only provide the model with explicit long-term memory but also introduce a recurrence mechanism to the transformer, which is a form of implicit memory. By removing

memory banks and instead including the recurrence mechanism in the left context, we capture the benefits of this implicit memory without the additional cost to compute memory banks.

2) **Attention-based left context**: In using the output from the attention calculation of the previous segment rather than the raw segment input as left context like the Transformer-XL, we are able to capture a *learned* representation of the previous segment at a given layer. This is similar to the Augmented Memory Transformer using the attention output associated with the summarization query as a memory bank. However, since we do not compress the segment into a summarization query, we capture a more realistic representation.

3) **Removed left context in the queries**: The Augmented Memory Transformer, includes the left context in each segment and, subsequently, the queries to allow it to generate a learned representation of the left context alongside the current segment. However, since our Implicit Memory Transformer already has a saved learned representation of the left context for a given layer, it removes the need to include the left context in the segment.

From the above attributes, the self-attention calculation of the Implicit Memory Transformer becomes more efficient than that of the Augmented Memory Transformer, as memory banks are no longer included in the keys and values, and the left context and summarization query are removed from the queries. Furthermore, our Implicit Memory Transformer reduces the computation cost of the feed-forward neural network and the convolution subsampling layers, as they no longer need to process tokens contained in the left context.

3.2 Complexity Analysis

We will now perform complexity analysis for the self-attention and convolution subsampling layers in the Augmented Memory Transformer. The complexity analysis of a convolution subsampling layer with a kernel size of one is identical to that for the linear transformations in the feed-forward network. The self-attention layer has a complexity of $O(n^2 \cdot d)$ and the convolution layer has a complexity of $O(K \cdot n \cdot d^2)$ where n is the input sequence length, d is the hidden size, and K is the kernel size (Vaswani et al., 2017).

The complexity of the self-attention layer of the old Augmented Memory Transformer would thus be $O((N + l + c + r)(l + c + r) \cdot d)$ and the complexity with the new method of calculating left context would be $O((c + r)(l + c + r) \cdot d)$. Similarly the complexity of the convolution layers would change from $O(K \cdot (l + c + r) \cdot d^2)$ to $O(K \cdot (c + r) \cdot d^2)$. Given the computational complexity decrease for all layers in the Augmented Memory Transformer with respect to the left context size and memory banks, it lends to the possibility of increasing the left context size for greater translation performance.

4 Experimental Setup

We conducted experiments on the English-German (en-de), English-French (en-fr), and English-Spanish (en-es) language pairs from the MuST-C dataset (Cattoni et al., 2021). The data preparation scripts for the MuST-C dataset are provided in Fairseq[1] (Ott et al., 2019; Wang et al., 2020), whereby Kaldi is used to generate 80-dimensional log-mel filter bank features, and text is tokenized with a SentencePiece 10k unigram vocabulary. The statistics of the training, development, and test set (tst-COMMON) for the English-German, English-French, and English-Spanish language pairs of the MuST-C dataset are provided in Table 1.

Language Pair	Train	Dev	Test
en-de	250942	1415	2580
en-fr	275085	1412	2632
en-es	265625	1316	2502

Table 1: The number of sentences in the train, development, and test (tst-COMMON) sets of the MuST-C dataset for the en-de, en-fr, en-es language pairs (Cattoni et al., 2021).

The architectures of the Augmented Memory Transformer and Implicit Memory Transformer trained were nearly identical, containing 33.1 M parameters (Ma et al., 2021). Their encoders consisted of 12 layers beginning with two convolution layers with a combined subsampling factor of 4, followed by a feed-forward neural network. Their decoders consisted of 6 layers. Each of these layers has a hidden size of 256 with 4 attention heads. Relative positional encodings were applied to each self-attention layer with a clipping distance of 16 (Shaw et al., 2018). Layer normalization was performed prior to each layer. Additionally, we trained each model with a wait-1, wait-3, wait-5, and wait-7 policy using a pre-decision ratio of 8 (Ma et al.,

[1] `https://github.com/facebookresearch/fairseq`

2020b). We provide public access to a derivative of Fairseq containing our implementation for the Implicit Memory Transformer[2].

All training was performed on a single V100-32GB. The training process consisted of ASR pre-training followed by SimulST training. For SimulST training, the models were trained with label-smoothed cross-entropy loss, the Adam optimizer (Kingma and Ba, 2014), and an inverse square root scheduler. There was a warm-up period of 7500 updates where the learning rate of 0.0001, followed by a learning rate of 0.00035. To regularize the model weights, we used a weight decay value of 0.0001, a dropout of 0.1, an activation dropout of 0.2, and an attention dropout of 0.2. All models were trained with early stopping using a patience of 10. After the training was complete, the final ten checkpoints were averaged.

The translation quality and latency were determined by detokenized BLEU with SacreBLEU (Post, 2018), and Average Lagging (Ma et al., 2020b), respectively. Both of these metrics were obtained using the SimulEval toolkit[3], which simulates SimulST (Ma et al., 2020a).

5 Results

5.1 Performance Evaluation

We demonstrate the efficacy of our Implicit Memory Transformer on the English-German language pair for a single run in Figure 1 in terms of average lagging and BLEU score. Figure 1 compares our Implicit Memory Transformer against two Augmented Memory Transformers differing by the inclusion or exclusion of memory banks. Each of the compared models consists of a left context of 32 tokens, a right context of 32 tokens, and a center context of 64 tokens. The Augmented Memory Transformer using memory banks have a total of three banks, whereas our Implicit Memory Transformer and the Augmented Memory Transformer without memory banks have zero.

From viewing Figure 1, our Implicit Memory Transformer achieves almost identical performance in terms of BLEU score to the Augmented Memory Transformer using memory banks for SimulST between English and German without affecting the Average Lagging. In contrast, simply removing memory banks in the Augmented Memory Transformer results in an average 4.48 BLEU decrease

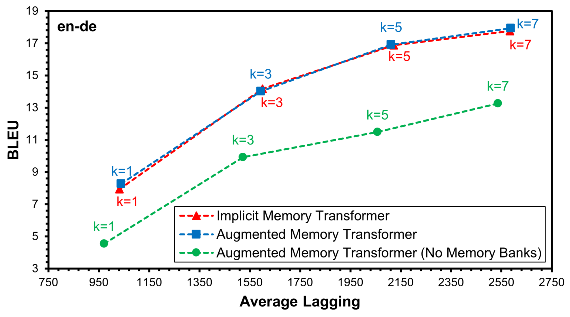

Figure 1: A comparison between the Implicit Memory Transformer and the baseline Augmented Memory Transformers on the **en-de** language pair.

versus its memory bank counterpart across all wait-k values. This confirms the effectiveness of the attention-generated left context of the proposed Implicit Memory Transformer for the English-German language pair.

We see similar results with the English-French and English-Spanish language pairs provided in Figure 2 and Figure 3, respectively.

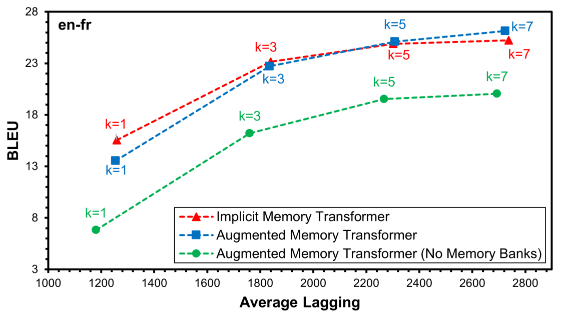

Figure 2: A comparison between the Implicit Memory Transformer and the baseline Augmented Memory Transformers on the **en-fr** language pair.

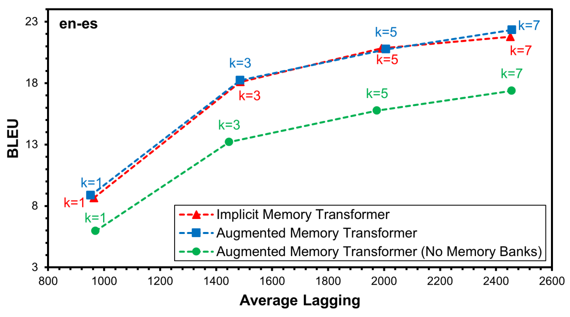

Figure 3: A comparison between the Implicit Memory Transformer and the baseline Augmented Memory Transformers on the **en-es** language pair.

In both cases, the Implicit Memory Transformer performs nearly identically to the Augmented Memory Transformer using memory banks by not

[2]https://github.com/OSU-STARLAB/ImplicitMemory
[3]https://github.com/facebookresearch/SimulEval

negatively impacting either the BLEU score or Average Lagging. Additionally, as with the results in Figure 1, the Augmented Memory Transformer sees an average decrease of 6.23 BLEU and 4.47 BLEU across all wait-k values when memory banks are removed for the English-French and English-Spanish language pairs respectively. Once again substantiating the efficacy of our attention-generated left context in the Implicit Memory Transformer, which does not see a performance decrease without memory banks.

5.2 Evaluation Speedup

We provide a demonstration of how the left context size affects the forward pass time of a segment through the encoder of an Augmented Memory Transformer with three memory banks, an Augmented Memory Transformer without memory banks, and the Implicit Memory Transformer in Figure 4. The left context size is scaled with tokens, and the duration of the forward pass of a segment through the encoder is scaled in milliseconds. Each model compared uses a right context of 32 tokens and a center context of 64 tokens for each tested left context size. Each measurement point in Figure 4 is made by averaging the duration of ten forward passes through the encoder using two 14-core 2.20 GHz Intel Xeon Gold 5120 with 19712 KB cache.

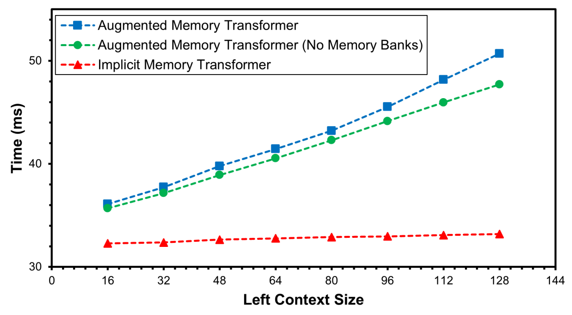

Figure 4: A comparison between the forward pass time (ms) of the Implicit Memory Transformer and two Augmented Memory Transformer variations with respect to the left context size.

In Figure 4, the forward pass time of the Implicit Memory Transformer remains flat with respect to the left context size, whereas the two Augmented Memory Transformer models exhibit a non-linear curved relationship. The separation of two Augmented Memory Transformer curves becomes more apparent for larger left context sizes, indicating the cost of memory banks becomes more apparent as the segment size increases. Figure 4 also shows that dropping the left context from the query in the proposed Implicit Memory Transformer achieves considerable additional computation reduction beyond removing memory banks.

6 Conclusion

Achieving computationally efficient simultaneous speech translation (SimulST) is critical to its deployment in practical real-time applications. However, even with the state-of-the-art SimulST approach of the Augmented Memory Transformer, its method of generating left context is computationally costly and ineffective, requiring the usage of memory banks to compensate for its shortcomings. As such, we propose an Implicit Memory Transformer that utilizes an attention-based left context to provide the model with implicit memory. We found that the Implicit Memory Transformer was able to achieve nearly identical performance to the Augmented Memory Transformer at a significantly reduced computational cost.

Limitations

Our work is limited as it has not explored the effectiveness of our Implicit Memory Transformer in other tasks outside of SimulST, such as ASR. We have also not explored the impact of our implicit memory left context on alternative block-processing-based transformer models. Furthermore, extensive ablation studies could help showcase the potential of the Implicit Memory Transformer.

Acknowledgements

This research was supported, in part, by the National Science Foundation grants 2223483 and 2223484.

References

Roldano Cattoni, Mattia Antonino Di Gangi, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2021. Must-c: A multilingual corpus for end-to-end speech translation. *Computer Speech & Language*, 66:101155.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Linhao Dong, Feng Wang, and Bo Xu. 2019. Self-attention aligner: A latency-control end-to-end model for asr using self-attention network and chunk-hopping. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5656–5660. IEEE.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, et al. 2018. Stacl: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. *arXiv preprint arXiv:1810.08398*.

Xutai Ma, Mohammad Javad Dousti, Changhan Wang, Jiatao Gu, and Juan Pino. 2020a. SIMULEVAL: An evaluation toolkit for simultaneous translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 144–150, Online. Association for Computational Linguistics.

Xutai Ma, Juan Pino, and Philipp Koehn. 2020b. Simulmt to simulst: Adapting simultaneous text translation to end-to-end simultaneous speech translation. *arXiv preprint arXiv:2011.02048*.

Xutai Ma, Yongqiang Wang, Mohammad Javad Dousti, Philipp Koehn, and Juan Pino. 2021. Streaming simultaneous speech translation with augmented memory transformer. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7523–7527. IEEE.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Changhan Wang, Yun Tang, Xutai Ma, Anne Wu, Dmytro Okhonko, and Juan Pino. 2020. fairseq s2t: Fast speech-to-text modeling with fairseq. In *Proceedings of the 2020 Conference of the Asian Chapter of the Association for Computational Linguistics (AACL): System Demonstrations*.

Chunyang Wu, Yongqiang Wang, Yangyang Shi, Ching-Feng Yeh, and Frank Zhang. 2020. Streaming transformer-based acoustic models using self-attention with augmented memory. *arXiv preprint arXiv:2005.08042*.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitations Section

☒ A2. Did you discuss any potential risks of your work?
We proposed a computationally efficient Implicit Memory Transformer for simultaneous speech translation. We do not believe our contributions pose any substantial risks.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 4

☑ B1. Did you cite the creators of artifacts you used?
Section 4

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
The license for the artifacts we used is provided in our GitHub repository.

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
A discussion of our use of existing artifacts being consistent with their intended use is provided in our GitHub repository. Similarly, we mention the intended use of our Implicit Memory Transformer implementation in our GitHub repository.

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
For our experiments, we utilized the MuST-C dataset derived from the publicly available audio of English TED Talks.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 4

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

C ☑ Did you run computational experiments?

Section 5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 5

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 4

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.