SimulMT to SimulST: Adapting Simultaneous Text Translation to End-to-End Simultaneous Speech Translation

Xutai Ma
Johns Hopkins University
xutai_ma@jhu.edu

Juan Pino
Facebook AI
juancarabina@fb.com

Philipp Koehn
Johns Hopkins University
phi@jhu.edu

Abstract

Simultaneous text translation and end-to-end speech translation have recently made great progress but little work has combined these tasks together. We investigate how to adapt simultaneous text translation methods such as wait-k and monotonic multihead attention to end-to-end simultaneous speech translation by introducing a pre-decision module. A detailed analysis is provided on the latency-quality trade-offs of combining fixed and flexible pre-decision with fixed and flexible policies. We also design a novel computation-aware latency metric, adapted from Average Lagging. [1]

1 Introduction

Simultaneous speech translation (SimulST) generates a translation from an input speech utterance before the end of the utterance has been heard. SimulST systems aim at generating translations with maximum quality and minimum latency, targeting applications such as video caption translations and real-time language interpreter. While great progress has recently been achieved on both end-to-end speech translation (Ansari et al., 2020) and simultaneous text translation (SimulMT) (Grissom II et al., 2014; Gu et al., 2017; Luo et al., 2017; Lawson et al., 2018; Alinejad et al., 2018; Zheng et al., 2019b,a; Ma et al., 2020; Arivazhagan et al., 2019, 2020), little work has combined the two tasks together (Ren et al., 2020).

End-to-end SimulST models feature a smaller model size, greater inference speed and fewer compounding errors compared to their cascade counterpart, which perform streaming speech recognition followed by simultaneous machine translation. In addition, it has been demonstrated that end-to-end SimulST systems can have lower latency than cascade systems (Ren et al., 2020).

In this paper, we study how to adapt methods developed for SimulMT to end-to-end SimulST. To this end, we introduce the concept of pre-decision module. Such module guides how to group encoder states into meaningful units prior to making a READ/WRITE decision. A detailed analysis of the latency-quality trade-offs when combining a fixed or flexible pre-decision module with a fixed or flexible policy is provided. We also introduce a novel computation-aware latency metric, adapted from Average Lagging (AL) (Ma et al., 2019).

2 Task formalization

A SimulST model takes as input a sequence of acoustic features $\boldsymbol{X} = [\boldsymbol{x}_1, ...\boldsymbol{x}_{|\boldsymbol{X}|}]$ extracted from speech samples every T_s ms, and generates a sequence of text tokens $\boldsymbol{Y} = [y_1, ..., y_{|\boldsymbol{Y}|}]$ in a target language. Additionally, it is able to generate y_i with only partial input $\boldsymbol{X}_{1:n(y_i)} = [\boldsymbol{x}_1, ...\boldsymbol{x}_{n(y_i)}]$, where $n(y_i) \leq |\boldsymbol{X}|$ is the number of frames needed to generate the i-th target token y_i. Note that n is a monotonic function, i.e. $n(y_{i-1}) \leq n(y_i)$.

A SimulST model is evaluated with respect to quality, using BLEU (Papineni et al., 2002), and latency. We introduce two latency evaluation methods for SimulST that are adapted from SimulMT. We first define two types of delays to generate the word y_i, a computation-aware (CA) and a non computation-aware (NCA) delay. The CA delay of y_i, $d_{\text{CA}}(y_i)$, is defined as the time that elapses (speech duration) from the beginning of the process to the prediction of y_i, while the NCA delay for y_i $d_{\text{CA}}(y_i)$ is defined by $d_{\text{NCA}}(y_i) = T_{\text{s}} \cdot n(y_i)$. Note that d_{NCA} is an ideal case for d_{CA} where the computational time for the model is ignored. Both delays are measured in milliseconds. Two types of latency measurement, L_{CA} and L_{NCA}, are calculated accordingly: $L = \mathcal{C}(\boldsymbol{D})$ where $\mathcal{C}$ is a latency metric and $\mathbf{D} = [d(y_1), ..., d(y_{|\boldsymbol{Y}|})]$.

[1]The code is available at https://github.com/pytorch/fairseq

To better evaluate the latency for SimulST, we introduce a modification to AL. We assume an oracle system that can perform perfect simultaneous translation for both latency and quality, while in Ma et al. (2019) the oracle is ideal only from the latency perspective. We evaluate the lagging based on time rather than steps. The modified AL metric is defined in Eq. (1):

$$\mathrm{AL} = \frac{1}{\tau(|\boldsymbol{X}|)} \sum_{i=1}^{\tau(|\boldsymbol{X}|)} d(y_i) - \frac{|\boldsymbol{X}|}{|\boldsymbol{Y}^*|} \cdot T_s \cdot (i-1) \quad (1)$$

where $|\boldsymbol{Y}^*|$ is the length of the reference translation, $\tau(|\boldsymbol{X}|)$ is the index of the first target token generated when the model read the full input. There are two benefits from this modification. The first is that latency is measured using time instead of steps, which makes it agnostic to preprocessing and segmentation. The second is that it is more robust and can prevent an extremely low and trivial value when the prediction is significantly shorter than the reference.

3 Method

3.1 Model Architecture

End-to-end ST models directly map a source speech utterance into a sequence of target tokens. We use the S-Transformer architecture proposed by (Di Gangi et al., 2019b), which achieves competitive performance on the MuST-C dataset (Di Gangi et al., 2019a). In the encoder, a two-dimensional attention is applied after the CNN layers and a distance penalty is introduced to bias the attention towards short-range dependencies.

We investigate two types of simultaneous translation mechanisms, flexible and fixed policy. In particular, we investigate monotonic multihead attention (Ma et al., 2020), which is an instance of flexible policy and the prefix-to-prefix model (Ma et al., 2019), an instance of fixed policy, designated by wait-k from now on.

Monotonic Multihead Attention (MMA) (Ma et al., 2020) extends monotonic attention (Raffel et al., 2017; Arivazhagan et al., 2019) to Transformer-based models. Each head in each layer has an independent step probability p_{ij} for the ith target and jth source step, and then uses a closed form expected attention for training. A weighted average and variance loss were proposed to control the behavior of the attention heads and thus the trade-offs between quality and latency.

Wait-k (Ma et al., 2019) is a fixed policy that waits for k source tokens, and then reads and writes alternatively. Wait-k can be a special case of Monotonic Infinite-Lookback Attention (MILk) (Arivazhagan et al., 2019) or MMA where the stepwise probability $p_{ij} = 0$ if $j - i < k$ else $p_{ij} = 1$.

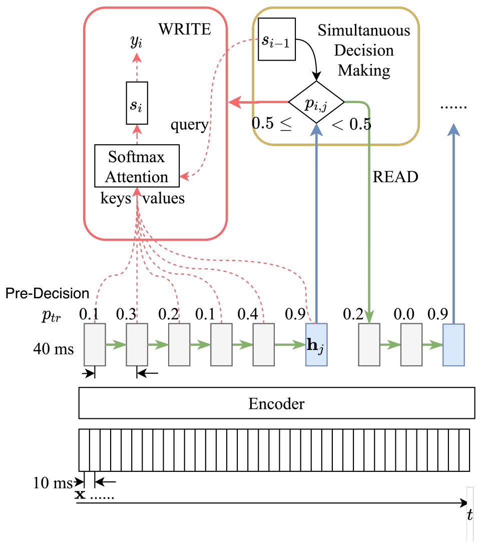


Figure 1: Simul-ST architecture with pre-decision module. Blue states in the figure indicate the point Simul-SST model triggers the simultaneous making process

3.2 Pre-Decision Module

In SimulMT, READ or WRITE decisions are made at the token (word or BPE) level. However, with speech input, it is unclear when to make such decisions. For example, one could choose to read or write after each frame or after generating each encoder state. Meanwhile, a frame typically only covers 10ms of the input while an encoder state generally covers 40ms of the input (assuming a subsampling factor of 4), while the average length of a word in our dataset is 270ms. Intuitively, a policy like wait-k will not have enough information to write a token after reading a frame or generating an encoder state. In principle, a flexible or model-based policy such as MMA should be able to handle granulawhile MMA is more robust tr input. Our analysis will show, however, that o the granularity of the input, it also performs poorly when the input is too fine-grained.

In order to overcome these issues, we introduce the notion of pre-decision module, which groups frames or encoder states, prior to making a decision. A pre-decision module generates a series of trigger probabilities p_{tr} on each encoder states to indicate whether a simultaneous decision should be made. If $p_{tr} > 0.5$, the model triggers the simultaneous decision making, otherwise keeps reading new frames. We propose two types of pre-decision module.

Fixed Pre-Decision A straightforward policy for a fixed pre-decision module is to trigger simultaneous decision making every fixed number of frames. Let Δt be the time corresponding to this fixed number of frames, with Δt a multiple of T_s, and $r_e = \text{int}(|\boldsymbol{X}|/|\boldsymbol{H}|)$. p_{tr} at encoder step j is defined in Eq. (2):

$$p_{tr}(j) = \begin{cases} 1 & \text{if } \text{mod}(j \cdot r_e \cdot T_s, \Delta t) = 0, \\ 0 & \text{Otherwise.} \end{cases} \quad (2)$$

Flexible Pre-Decision We use an oracle flexible pre-decision module that uses the source boundaries either at the word or phoneme level. Let $\boldsymbol{A}$ be the alignment between encoder states and source labels (word or phoneme). $\boldsymbol{A}(h_i)$ represents the token that h_i aligns to. The trigger probability can then be defined in Eq. (3):

$$p_{tr}(j) = \begin{cases} 0 & \text{if } \boldsymbol{A}(h_j) = \boldsymbol{A}(h_{j-1}) \\ 1 & \text{Otherwise.} \end{cases} \quad (3)$$

4 Experiments

We conduct experiments on the English-German portion of the MuST-C dataset (Di Gangi et al., 2019a), where source audio, source transcript and target translation are available. We train on 408 hours of speech and 234k sentences of text data. We use Kaldi (Povey et al., 2011) to extract 80 dimensional log-mel filter bank features, computed with a $25ms$ window size and a $10ms$ window shift. For text, we use SentencePiece (Kudo and Richardson, 2018) to generate a unigram vocabulary of size 10,000. We use Gentle[2] to generate the alignment between source text and speech as the label to generate the oracle flexible predecision module. Translation quality is evaluated with case-sensitive detokenized BLEU with SacreBLEU (Post, 2018). The latency is evaluated with our proposed modification of AL (Ma et al., 2019). All results are reported on the MuST-C dev set.

All speech translation models are first pretrained on the ASR task where the target vocabulary is character-based, in order to initialize the encoder. We follow the same hyperparameter settings from (Di Gangi et al., 2019b). We follow the latency regularization method introduced by (Ma et al., 2020; Arivazhagan et al., 2019), The objective function to optimize is

$$L = -\log\left(P(\boldsymbol{Y}|\boldsymbol{X})\right) + \lambda \max\left(\mathcal{C}(\boldsymbol{D}), 0\right) \quad (4)$$

Where $\mathcal{C}$ is a latency metric (AL in this case) and $\boldsymbol{D}$ is described in Section 2. Only samples with AL > 0 are regularized to avoid overfitting. For the models with monotonic multihead attention, we first train a model without latency with $\lambda_{\text{latency}} = 0$. After the model converges, λ_{latency} is set to a desired value and the model is continue trained until convergence.

The latency-quality trade-offs of the 4 types of model from the combination of fixed or flexible predecision with fixed or flexible policy are presented in Fig. 2. The non computation-aware delays are used to calculate the latency metric in order to evaluate those trade-offs from a purely algorithmic perspective.

Fixed Pre-Decision + Fixed Policy [3] (Fig. 2a). As expected, both quality and latency increase with step size and lagging. In addition, the latency-quality trade-offs are highly dependent on the step size of the pre-decision module. For example, with step size 120ms, the performance is very poor even with large k because of very limited information being read before writing a target token. Large step sizes improve the quality but introduce a lower bound on the latency. Note that step size 280ms, which provides an effective latency-quality trade-off compared to other step sizes, also matches the average word length of 271ms. This motivates the study of a flexible pre-decision module based on word boundaries.

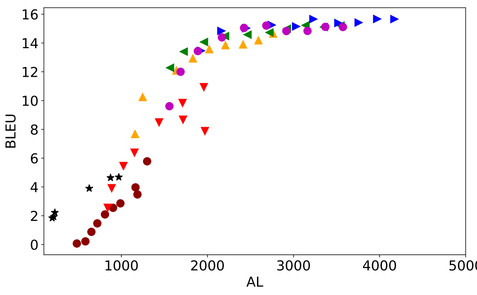
Fixed Pre-Decision + Flexible Policy [4] (Fig. 2b) Similar to wait-k, MMA obtains very poor performance with a small step size of 120ms. For other step sizes, MMA obtains similar latency-quality trade-offs, demonstrating some form of robustness to the step size.

Flexible Pre-Decision Curve ★ and ● in figure Fig. 2 show latency-quality trade-offs when the predecision module is determined by oracle word or phoneme boundaries. Note that a SimulST model

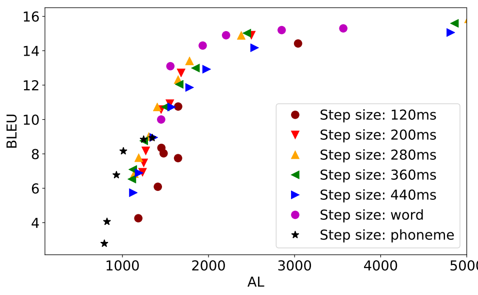
[2]https://lowerquality.com/gentle/

[3]$k = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$

[4]$\lambda = 0.001, 0.004, 0.01, 0.02, 0.04, 0.06, 0.08, 0.1$



(a) Wait-k



(b) MMA

Figure 2: Latency-Quality trade-off curves. The unit of AL is millisecond

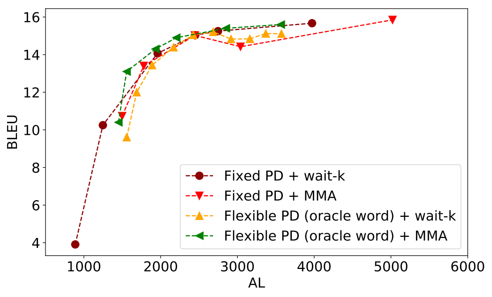


Figure 3: Comparison of best models in four settings

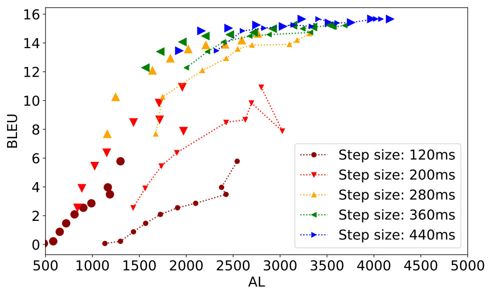


Figure 4: Computation-aware latency for fixed pre-decision + wait-k policy. Points on dotted lines are computation-aware, without lines are non-computation-aware

would not normally have access to this information and that the purpose of this experiment is to guide future design of a flexible pre-decision model. First, as previously observed, the granularity of the pre-decision greatly influences the latency-quality trade-offs. Models using phoneme boundaries obtain very poor translation quality because those boundaries are too granular, with an average phoneme duration of 77ms. In addition, comparing MMA and wait-k with phoneme boundaries, MMA is found to be more robust to the granularity of the pre-decision.

Best Curves The best settings for each approach are compared in Fig. 3. For fixed pre-decision, we choose the setting that has the best quality for each latency bucket of 500ms, while for the flexible pre-decision we use oracle word boundaries. For both wait-k and MMA, the flexible pre-decision module outperforms the fixed pre-decision module. This is expected since the flexible pre-decision module uses oracle information in the form of pre-computed word boundaries but provides a direction for future research. The best latency-quality trade-offs are obtained with MMA and flexible pre-decision from word boundaries.

4.1 Computation Aware Latency

We also consider the computation-aware latency described in Section 2, shown in Fig. 4. The focus is on fixed pre-decision approaches in order to understand the relation between the granularity of the pre-decision and the computation time. Fig. 4 shows that as the step size increases, the difference between the NCA and the CA latency shrinks. This is because with larger step sizes, there is less overhead of recomputing the bidirectional encoder states [5]. We recommend future work on SimulST to make use of CA latency as it reflects a more realistic evaluation, especially in low-latency regimes, and is able to distinguish streaming capable systems.

5 Conclusion

We investigated how to adapt SimulMT methods to end-to-end SimulST by introducing the concept of pre-decision module. We also adapted Average Lagging to be computation-aware. The effects of combining a fixed or flexible pre-decision module

[5]This is a common practice in SimulMT where the input length is significantly shorter than in SimulST (Arivazhagan et al., 2019; Ma et al., 2019; Arivazhagan et al., 2020)

with a fixed or flexible policy were carefully analyzed. Future work includes building an incremental encoder to reduce the CA latency and design a learnable pre-decision module.

References

Ashkan Alinejad, Maryam Siahbani, and Anoop Sarkar. 2018. Prediction improves simultaneous neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3022–3027.

Ebrahim Ansari, amittai axelrod, Nguyen Bach, Ondřej Bojar, Roldano Cattoni, Fahim Dalvi, Nadir Durrani, Marcello Federico, Christian Federmann, Jiatao Gu, Fei Huang, Kevin Knight, Xutai Ma, Ajay Nagesh, Matteo Negri, Jan Niehues, Juan Pino, Elizabeth Salesky, Xing Shi, Sebastian Stüker, Marco Turchi, Alexander Waibel, and Changhan Wang. 2020. FINDINGS OF THE IWSLT 2020 EVALUATION CAMPAIGN. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 1–34, Online. Association for Computational Linguistics.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. 2019. Monotonic infinite lookback attention for simultaneous machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1313–1323, Florence, Italy. Association for Computational Linguistics.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, and George Foster. 2020. Re-translation versus streaming for simultaneous translation. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 220–227, Online. Association for Computational Linguistics.

Mattia A. Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2019a. MuST-C: a Multilingual Speech Translation Corpus. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2012–2017, Minneapolis, Minnesota. Association for Computational Linguistics.

Mattia Antonino Di Gangi, Matteo Negri, Roldano Cattoni, Roberto Dessi, and Marco Turchi. 2019b. Enhancing transformer for end-to-end speech-to-text translation. In *Proceedings of Machine Translation Summit XVII Volume 1: Research Track*, pages 21–31, Dublin, Ireland. European Association for Machine Translation.

Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. 2014. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Proceedings of the 2014 Conference on empirical methods in natural language processing (EMNLP)*, pages 1342–1352.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor OK Li. 2017. Learning to translate in real-time with neural machine translation. In *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017*, pages 1053–1062. Association for Computational Linguistics (ACL).

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71.

Dieterich Lawson, Chung-Cheng Chiu, George Tucker, Colin Raffel, Kevin Swersky, and Navdeep Jaitly. 2018. Learning hard alignments with variational inference. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5799–5803. IEEE.

Yuping Luo, Chung-Cheng Chiu, Navdeep Jaitly, and Ilya Sutskever. 2017. Learning online alignments with continuous rewards policy gradient. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2801–2805. IEEE.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. 2019. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036, Florence, Italy. Association for Computational Linguistics.

Xutai Ma, Juan Pino, James Cross, Liezl Puzon, and Jiatao Gu. 2020. Monotonic multihead attention. In *International Conference on Learning Representations*.

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 on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko

Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, CONF. IEEE Signal Processing Society.

Colin Raffel, Minh-Thang Luong, Peter J Liu, Ron J Weiss, and Douglas Eck. 2017. Online and linear-time attention by enforcing monotonic alignments. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2837–2846. JMLR. org.

Yi Ren, Jinglin Liu, Xu Tan, Chen Zhang, Tao QIN, Zhou Zhao, and Tie-Yan Liu. 2020. SimulSpeech: End-to-end simultaneous speech to text translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3787–3796, Online. Association for Computational Linguistics.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019a. Simpler and faster learning of adaptive policies for simultaneous translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1349–1354, Hong Kong, China. Association for Computational Linguistics.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019b. Simultaneous translation with flexible policy via restricted imitation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5816–5822, Florence, Italy. Association for Computational Linguistics.