Over-Generation Cannot Be Rewarded: Length-Adaptive Average Lagging for Simultaneous Speech Translation

Sara Papi[◇,△], **Marco Gaido**[◇,△], **Matteo Negri**[◇], **Marco Turchi**[◇]
[◇]Fondazione Bruno Kessler
[△]University of Trento
{spapi, mgaido, negri, turchi}@fbk.eu

Abstract

Simultaneous speech translation (SimulST) systems aim at generating their output with the lowest possible latency, which is normally computed in terms of Average Lagging (AL). In this paper we highlight that, despite its widespread adoption, AL provides underestimated scores for systems that generate longer predictions compared to the corresponding references. We also show that this problem has practical relevance, as recent SimulST systems have indeed a tendency to over-generate. As a solution, we propose LAAL (Length-Adaptive Average Lagging), a modified version of the metric that takes into account the over-generation phenomenon and allows for unbiased evaluation of both under-/over-generating systems.

1 Introduction

Simultaneous speech-to-text translation (SimulST) is the task in which the generation of the textual translation in the target language starts before the entire audio input in the source language has been ingested by the model. The need to have high quality translations in the shortest possible time therefore becomes the main objective of SimulST systems, which have to satisfy specific time requirements depending on the application scenarios. These requirements are usually expressed in terms of latency, that is the elapsed time between the pronunciation of a word and the generation of its textual translation. How latency is measured hence plays a crucial role in systems evaluation.

The SimulST task was initially addressed using cascaded models (Fügen et al., 2007; Oda et al., 2014) that divide the translation process into two steps: simultaneous automatic speech recognition (Jaitly et al., 2016; Rao et al., 2017), and simultaneous machine translation (Cho and Esipova, 2016; Gu et al., 2017). For this reason, the first latency metrics were designed to evaluate simultaneous machine translation (SimulMT) systems (Cho and Esipova, 2016; Cherry and Foster, 2019; Elbayad et al., 2020). Among them, Average Lagging – AL – (Ma et al., 2019) is the most popular one, and its adaptation to SimulST by Ma et al. (2020a) has become a widely adopted (Ma et al., 2020b; Zeng et al., 2021; Chen et al., 2021; Liu et al., 2021) *de facto* standard.[1] The adaptation by Ma et al. (2020a) sparks from a weakness observed in the original formulation of the metric. Being susceptible to *under-generation*, it results in biased evaluations favouring systems that produce shorter predictions compared to the reference. However, though successful in correcting this behaviour, the proposed adaptation did not consider the opposite case of *over-generation*, which occurs when the prediction is longer than the reference.

To fill this gap, in this paper we introduce LAAL (Length-Adaptive Average Lagging):[2] a simple yet effective extension of AL that takes into account also over-generation and allows for fair SimulST systems comparisons. After a brief explanation of AL calculation (Section 2), we expose its incorrect behaviour in presence of over-generation phenomena (Section 3) and show that over-generation is actually present in the output of recent SimulST systems (Section 4). Then, we present the new LAAL metric (Section 5), whose computation is adjusted at sentence level by looking at the length of model predictions. Through examples, we show that, unlike the previous AL formulation, our metric is able to fairly evaluate both under- and over-generating systems. We conclude our work with a discussion (Section 6) about problems that still need to be solved for latency computation, remarking that our proposal represents a first step toward a more reliable assessment of SimulST systems

[1]For instance, the IWSLT SimulST Shared Task (Anastasopoulos et al., 2021) relies on AL to divide the systems in different latency regimes (low, medium, high) and BLEU (Post, 2018) to rank the them based on translation quality.

[2]The code is available at: https://github.com/hlt-mt/FBK-fairseq.

Proceedings of the Third Workshop on Automatic Simultaneous Translation, pages 12 - 17
July 15-16, 2022 ©2022 Association for Computational Linguistics

performance.

2 Average Lagging

The idea behind the AL metric is to quantify how much time the system is out of sync with the speaker. In SimulST, the input sequence is represented as a stream of audio speech in the source language $\mathbf{X} = [x_1, ..., x_{|\mathbf{X}|}]$ where each element x_j is a raw audio segment of duration T_j, the reference as a stream of words in the target language $\mathbf{Y}^* = [y^*_1, ..., y^*_{|\mathbf{Y}^*|}]$, and the model translation as a stream of predicted words $\mathbf{Y} = [y_1, ..., y_{|\mathbf{Y}|}]$. In the simultaneous setting, a system starts to generate a partial hypothesis while it continues to receive an incremental stream of input. This implies that, to generate the y_i target word at time j, it has access to $\mathbf{X}_{1:j} = [x_1, ..., x_j]$ with $j < |\mathbf{X}|$.

Therefore, the delay with which the y_i word is emitted is $d_i = \sum_{i=1}^{j} T_i$. Using this notation, in (Ma et al., 2020a) Average Lagging was initially defined as follows:

$$AL = \frac{1}{\tau'(|\mathbf{X}|)} \sum_{i=1}^{\tau'(|\mathbf{X}|)} d_i - d^*_i \quad (1)$$

$$d^*_i = (i-1) \cdot \frac{\sum_{j=1}^{|\mathbf{X}|} T_j}{|\mathbf{Y}|} \quad (2)$$

where $\tau'(|\mathbf{X}|) = \min\{i | d_i = \sum_{j=1}^{|\mathbf{X}|} T_j\}$ is the index of the target token when the end of the source sentence is reached and d^*_i represents an oracle that, perfectly in sync with the speaker, starts to emit words as soon as the speech starts.

However, the authors noticed that this adaptation was not robust for models that tend to stop generating the hypothesis too early, that is systems that under-generate. This phenomenon is more likely to happen in SimulST than in SimulMT, for which AL was first proposed. For instance, the presence of long pauses in the speech may induce systems to generate the end of sentence token too early, even if the source utterance is not yet complete. As observed by the authors, when this phenomenon occurs, the lagging behind the oracle becomes negative. It follows that relatively good latency-quality trade-offs can be achieved thanks to inappropriate AL discounts in case of under-generation, while this does not reflect the reality. Thus, in (Ma et al., 2020a), Equation 1 was redefined as:

$$d^*_i = (i-1) \cdot \frac{\sum_{j=1}^{|\mathbf{X}|} T_j}{|\mathbf{Y}^*|} \quad (3)$$

assuming that the oracle delays d^*_i are computed based on the reference length rather than on the system hypothesis length.

3 The Problem of Over-Generation

In this paper, we point out a major issue of AL that arises in presence of over-generation. As we will see, AL improperly favors over-generating systems, potentially leading the community to wrong conclusions due to biased evaluations. To illustrate how over-generation affects AL computation, we consider a real example from the English→Spanish (en-es) section of MuST-C (Cattoni et al., 2021) tst-COMMON translated by the state-of-the-art Cross Attention Augmented Transducer (CAAT) system (Liu et al., 2021).

As shown in Figure 1, the prediction suffers from over-generation, especially in the first part of the sentence where more target words are produced compared to the reference. The system translates "*En primer lugar,*" instead of "*Primero,*", forcing all the predicted words to compare with the successive word in the reference. For instance, "*es*" in the CAAT output is computed against "*juego*" in the oracle and its very low lagging (49ms) is a considerable underestimation of the correct lagging with respect to the "*es*" word in the oracle (763ms). Likewise, "*de*" is assigned a negative lagging (−62ms) instead of the 652ms delay with respect to the time of "*de*" in the oracle. Finally, all the words generated before the end of the utterance (in our example 5000ms) and exceeding the reference length are compared with the last word of the oracle; the same happens to the first word emitted when the utterance is over, while the other words after the end of the utterance are ignored. As a result, the AL of CAAT output for this sentence is 198ms, an extremely low latency that does not reflect the truth. Indeed, if we correctly align and compare the words in the system output and the oracle, we see that lagging is on average 846ms.[3] This represents a problem, since the AL metric is rewarding an over-generating system, potentially hindering a fair comparison with other models.

In light of these observations, two questions arise. First, *what are the conditions leading to biased, i.e. underestimated, AL values?* The example above shows that the problem arises when: *i)* the system over-generates, and *ii)* the over-generated words appear before the utterance ends (the earlier

[3]The detailed calculation is presented in Appendix A.

Figure 1: Example of AL computation between the oracle delays (in green) and the system delays (in blue). The lagging values (in red) are computed as the difference between the system and the oracle delays (the mapping is represented by an arrow). The 198ms AL is obtained by dividing the sum of the lagging (3379ms) by its count (17).

in time, the lower the final AL score will be). Second, *why was this problem overlooked when AL was introduced?* To answer this question, recall that AL for SimulST was initially proposed to evaluate systems showing a biased behaviour toward under-generation, with predictions length reaching at most the reference length. Indeed, earlier SimulST systems (Ma et al., 2020b) were designed to emit only one word at each time step. Consequently, even in the case of over-generation, it was extremely unlikely for the number of words emitted before the end of the utterance to be higher than the number of words in the reference. Moreover, additional words (if any) were generated only once the utterance was over. As mentioned above, the current AL implementation ignores all but the first word emitted at the end of the utterance. Therefore, the over-generation occurring at the end of the utterance does not affect the metric computation. Instead, AL is not robust to over-generation if it occurs before the end of the utterance, an extremely unlikely behavior in early systems but frequent in more recent ones, as we will see in the next section.

4 Over-generation frequency

To quantify the impact of over-generation on system evaluation, we check how frequently it occurs in the output of three SimulST systems: CAAT, the wait-k model by Ma et al. (2020b), and an offline model with the wait-k policy applied only at inference time (Papi et al., 2022; Gaido et al., 2022). We run three systems on the en-es section of MuST-C tst-COMMON by varying the k value at inference time in the range $\{3, 5, 7, 9, 11\}$. We measure over-generation in terms of average word length difference (AWLD) between systems predictions and the corresponding references, that is:

$$\text{AWLD} = \frac{1}{N}\sum_{s=1}^{N} |\mathbf{Y}| - |\mathbf{Y}^*| \qquad (4)$$

where N is the number of samples in the corpus. Accordingly, positive AWLD values indicate that system predictions are on average longer than the reference (over-generation), while negative values indicate systems tendency to under-generate.

Model	k=3	k=5	k=7	k=9	k=11
wait-k	-5.57	-3.82	-2.30	-1.13	-0.74
offline wait-k	0.48	0.49	0.53	0.74	0.80
CAAT	1.57	0.96	0.61	0.35	0.18

Table 1: AWLD on MuST-C en-es tst-COMMON.

Table 1 shows that the wait-k system under-generates – as already noticed by Ma et al. (2020b) – while both CAAT and offline wait-k ones over-generate. In addition, while for the offline wait-k model the over-generation phenomenon is quite constant for each k value, for CAAT this diminishes as k increases. This indicates that over-generation is not an isolated phenomenon affecting only few sentences. On the contrary, it frequently occurs and automatic evaluation should take this into account.

5 Length Adaptive Average Lagging

Based on the observations made in Sections 3 and 4, we propose LAAL (Length-Adaptive Average Lagging), a modified version of AL accounting also for the over-generation phenomena. LAAL defines the oracle delays by dividing the utterance length by the maximum between the reference and the model prediction length. Specifically, we consider the reference length when the prediction is shorter (under-generation), and the prediction length when the prediction is longer (over-generation). This means that the correction is made at sentence level, making the metric applicable to a system disregarding its under- or over-generation tendency.

Figure 2 shows both the under-generation (in blue) and the over-generation (in green) cases. Analyzing the under-generation case, we can clearly see the motivation behind the correction made by Ma et al. (2020a): if we consider the prediction

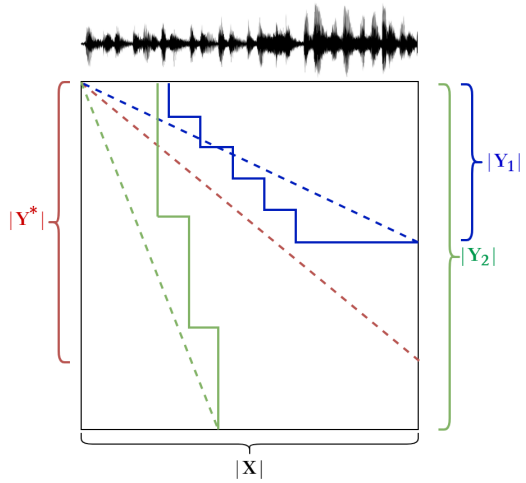

Figure 2: Example of under-generation (in blue), and over-generation (in green). The reference translation is represented by the red dashed line.

length ($|\mathbf{Y_1}|$) in the AL computation and the prediction is too short, the system is favoured since negative delays are summed (the blue straight line is mainly below the blue dashed line). A more reliable evaluation is obtained by considering the reference length ($|\mathbf{Y}^*|$), since the straight blue line is always above the dashed red line. By analyzing the over-generation case, we observe the opposite problem: if we consider the reference length ($|\mathbf{Y}^*|$) in the AL computation and the prediction is too long ($|\mathbf{Y_2}| > |\mathbf{Y}^*|$), the system is favoured since negative delays are summed (the straight green line stays almost always below the red dashed line). This can be corrected by considering the prediction length ($|\mathbf{Y_2}|$) instead. Therefore, to make a more reliable evaluation where neither under-generation nor over-generation are rewarded, we have to take the maximum between $|\mathbf{Y}^*|$ and $|\mathbf{Y}|$ in the delay computation (the two conditions are: $|\mathbf{Y}^*|$ if $|\mathbf{Y}| \leq |\mathbf{Y}^*|$, and $|\mathbf{Y}|$ if $|\mathbf{Y}| > |\mathbf{Y}^*|$). Accordingly, Equation 3 can be modified to obtain LAAL as:

$$d_i^* = (i-1) \cdot \frac{\sum_{j=1}^{|\mathbf{X}|} T_j}{\max\{|\mathbf{Y}|, |\mathbf{Y}^*|\}} \qquad (5)$$

The difference between applying AL and LAAL to evaluate our three systems is shown in Table 2. As we can see, the LAAL of the wait-k system is almost equal to the AL, with differences from 17 to 73ms. Conversely, for the offline wait-k system we notice a quite constant increment in LAAL of about 120ms while for the CAAT system we observe that LAAL is visibly greater than AL, with differences from 117 to 283ms that are more marked at low latency. These differences are coherent with the over-generation trend observed in Table 1.

Model	Metric	k=3	k=5	k=7	k=9	k=11
wait-k	AL	1761	1970	2272	2582	2931
	LAAL	1778	2001	2332	2655	3003
offline wait-k	AL	1522	1959	2463	2926	3350
	LAAL	1682	2093	2588	3043	3457
CAAT	AL	735	1149	1533	1905	2265
	LAAL	1018	1365	1708	2046	2382

Table 2: AL and LAAL results in ms of the wait-k and CAAT systems on MuST-C en-es tst-COMMON.

Going back to the example in Figure 1, the latency value computed with LAAL is 707ms. Compared to the AL value of 198ms, this is much closer to the real measure of 846ms calculated in Section 3. In light of these observations, we can conclude that the LAAL metric gives a more reliable evaluation of the SimulST systems compared to AL.

6 Limitations

The proposed LAAL metric is a first step toward a more accurate evaluation of SimulST systems. Although in this work we focused on the over-generation problem, we did not address another limitation of AL (and, in turn, of LAAL). The problem is that, as shown in Section 2, AL compares the system output with an oracle that emits only one word at each time step, each one with a fixed word duration.[4] This means that, in its computation, we assume that the reference words are uniformly distributed in each utterance. However, considering that the amount of information contained in audio segments of the same length could be extremely different, this represents an unrealistic approximation. For instance, a speech segment can contain silences, long pauses, and the speech rate can vary considerably. As a consequence, the latency scores obtained can still largely differ from the latency experienced by the user. This advocates for the development of more human-centric solutions that go beyond AL-like metrics despite their success, accounting for different audio phenomena and their impact on the actual latency perceived by the users, also considering the visualization strategy selected (Karakanta et al., 2021; Papi et al., 2021). We leave this line of investigation for future work.

[4] In our example in Figure 1, the word duration is 357ms and is computed dividing the source audio duration (5000ms) by the reference length (14).

7 Conclusions

We showed through examples based on real systems that the current Average Lagging computation is inadequate to correctly measure SimulST performance in presence of over-generation phenomena. To overcome this problem, we proposed Length-Adaptive Average Lagging (LAAL), a latency metric that can effectively handle both under- and over-generation at sentence level, leading to a more reliable evaluation of SimulST systems.

Acknowledgments

This work has been carried out as part of the project Smarter Interpreting (https://kunveno.digital/) financed by CDTI Neotec funds.

References

Antonios Anastasopoulos, Ondřej Bojar, Jacob Bremerman, Roldano Cattoni, Maha Elbayad, Marcello Federico, Xutai Ma, Satoshi Nakamura, Matteo Negri, Jan Niehues, Juan Pino, Elizabeth Salesky, Sebastian Stüker, Katsuhito Sudoh, Marco Turchi, Alexander Waibel, Changhan Wang, and Matthew Wiesner. 2021. FINDINGS OF THE IWSLT 2021 EVALUATION CAMPAIGN. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, pages 1–29, Bangkok, Thailand (online). Association for Computational Linguistics.

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.

Junkun Chen, Mingbo Ma, Renjie Zheng, and Liang Huang. 2021. Direct simultaneous speech-to-text translation assisted by synchronized streaming ASR. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4618–4624, Online. Association for Computational Linguistics.

Colin Cherry and George Foster. 2019. Thinking slow about latency evaluation for simultaneous machine translation.

Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation?

Maha Elbayad, Michael Ustaszewski, Emmanuelle Esperança-Rodier, Francis Brunet-Manquat, Jakob Verbeek, and Laurent Besacier. 2020. Online versus offline NMT quality: An in-depth analysis on English-German and German-English. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5047–5058, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Christian Fügen, Alex Waibel, and Muntsin Kolss. 2007. Simultaneous translation of lectures and speeches. *Machine Translation*, 21(4):209–252.

Marco Gaido, Sara Papi, Dennis Fucci, Giuseppe Fiameni, Matteo Negri, and Marco Turchi. 2022. Efficient yet competitive speech translation: FBK@IWSLT2022. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 177–189, Dublin, Ireland (in-person and online). Association for Computational Linguistics.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O.K. Li. 2017. Learning to translate in real-time with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1053–1062, Valencia, Spain. Association for Computational Linguistics.

Navdeep Jaitly, Quoc V Le, Oriol Vinyals, Ilya Sutskever, David Sussillo, and Samy Bengio. 2016. An online sequence-to-sequence model using partial conditioning. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Alina Karakanta, Sara Papi, Matteo Negri, and Marco Turchi. 2021. Simultaneous speech translation for live subtitling: from delay to display. In *Proceedings of the 1st Workshop on Automatic Spoken Language Translation in Real-World Settings (ASLTRW)*, pages 35–48.

Dan Liu, Mengge Du, Xiaoxi Li, Ya Li, and Enhong Chen. 2021. Cross attention augmented transducer networks for simultaneous translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 39–55, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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, 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. In *Proceedings of the 1st Conference of the*

Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, pages 582–587, Suzhou, China. Association for Computational Linguistics.

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Optimizing segmentation strategies for simultaneous speech translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 551–556, Baltimore, Maryland. Association for Computational Linguistics.

Sara Papi, Marco Gaido, Matteo Negri, and Marco Turchi. 2022. Does simultaneous speech translation need simultaneous models? *arXiv preprint arXiv:2204.03783*.

Sara Papi, Matteo Negri, and Marco Turchi. 2021. Visualization: The missing factor in simultaneous speech translation. In *Proceedings of the Eighth Italian Conference on 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.

Kanishka Rao, Haşim Sak, and Rohit Prabhavalkar. 2017. Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 193–199. IEEE.

Xingshan Zeng, Liangyou Li, and Qun Liu. 2021. RealTranS: End-to-end simultaneous speech translation with convolutional weighted-shrinking transformer. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2461–2474, Online. Association for Computational Linguistics.

A Example Manual Latency Calculation

To manually compute the latency measure of the example shown in Figure 1, we compare the model output words to the reference words of the oracle by correctly aligning them. For instance "*En premen lugar,*" of the model prediction is aligned to "*Primero,*" of the oracle, "*es*" of the model prediction to "*es*" of the oracle, and so on. Therefore, the lagging calculation will be the following:

$$(1120-0)+(1120-357)+(2080-714)+$$
$$(2080-1071)+(2080-1428)+(2080-1785)+$$
$$(3040-2142)+(3040-2500)+(4000-2857)+$$
$$(4000-3214)+(4960-3571)+(4960-4285)+$$
$$(5000-4642)=10994$$

Then, we divide the lagging sum of 10994ms by their count (13) to obtain the latency of 846ms.