Simulstream: Open-Source Toolkit for Evaluation and Demonstration of Streaming Speech-to-Text Translation Systems

Marco Gaido, Sara Papi, Mauro Cettolo, Matteo Negri, Luisa Bentivogli
Fondazione Bruno Kessler, Trento, Italy
{mgaido, spapi, cettolo, negri, bentivo}@fbk.eu

Abstract

Streaming Speech-to-Text Translation (StreamST) requires producing translations concurrently with incoming speech under strict latency constraints, demanding models that balance low latency with high translation quality. Despite rapid progress, evaluation remains fragmented across existing frameworks, which make different assumptions about how systems operate—for example, whether they process continuous speech or short pre-segmented audio, and whether they support output revision (*retranslation*) or not (*incremental*). For instance, SimulEval, the most widely used framework, supports only incremental decoding, assumes short segmented inputs, and lacks a native support for system demonstrations. As a result, comparing systems fairly and consistently across studies remains challenging, with no unified solution for benchmarking and interactive demonstration. To address this gap, we introduce `simulstream`, the first open-source framework for StreamST evaluation and demonstration. It supports both incremental and re-translation decoding on long-form speech, provides fine-grained logging for quality and latency evaluation, and includes an interactive web interface for real-time visualization and comparison.

github.com/hlt-mt/simulstream

1 Introduction

The task of Streaming Speech-to-Text Translation (StreamST) requires translating continuous, long-form speech input concurrently with the speaker's speech (Ren et al., 2020), posing a significant challenge at the intersection of automatic speech recognition and machine translation (Fügen et al., 2008). Fundamentally distinct from offline translation, where the entire utterance is available before processing, StreamST systems must adhere to specific *latency* constraints, defined as the time elapsed between when a segment of speech is spoken and when its translation is produced. Specifically, translation output must be generated *simultaneously*, or almost in sync, with the incoming speech to remain aligned with the pace of the speaker. This process requires making decisions with only partial information available—often called *policy* (Grissom II et al., 2014), which must balance low latency with high-quality translation. Despite these inherent challenges, considerable efforts in both academia and industry have led to rapid progress in the field, as evidenced by the growing number of publications on the topic (Papi et al., 2025).

Two main paradigms emerged for delivering StreamST outputs: *retranslation* and *incremental* decoding. Retranslation entails continuously regenerating the entire target sentence (or part of it) whenever new source speech is received. This strategy utilizes a non-monotonic decoding approach, enabling dynamic self-correction and revision of previous outputs to maximize translation quality, but often leading to visible output instability,[1] known as flickering (Arivazhagan et al., 2020b). Incremental decoding instead does not allow the revision or deletion of previously generated outputs (reflecting an "append-only" approach) at the benefit of a more stable user experience (Dalvi et al., 2018). Retranslation is prevalent in industrial applications (Arivazhagan et al., 2020a) as it is easy to apply, requiring no specific modification or re-training to the underlying offline speech translation model. In contrast, research has largely focused on incremental methods, including both training-based (Schneider and Waibel, 2020; Iranzo-Sánchez et al., 2021; Ouyang et al., 2025), and training-free approaches (Polák and Bojar, 2024; Papi et al., 2024). As both paradigms mature, understanding their trade-offs under consistent conditions has become essential for progress in simultaneous speech translation.

[1] https://research.google/blog/stabilizing-live-speech-translation-in-google-translate/

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 87–97
October 24-29, 2026 ©2026 Association for Computational Linguistics

However, systematic comparison between these paradigms remains an open challenge. Existing frameworks typically support only subsets of the StreamST pipeline, preventing fair and reproducible benchmarking across decoding strategies. The most widely used toolkit, SimulEval (Ma et al., 2020), was only designed for incremental decoding and does not support token deletion, making it incompatible with retranslation methods. In addition, it assumes short speech segments rather than continuous long-form streams, which limits its applicability to modern StreamST benchmarks (Abdulmumin et al., 2025; Polák et al., 2025). It also lacks support for interactive system demonstrations and is no longer actively maintained.[2] Beyond SimulEval, only a limited number of alternative tools have been proposed, with scarce adoption and providing only partial solutions: SLTev (Ansari et al., 2021) supports both paradigms but lacks demonstration capabilities, while Lecture Translator (Huber et al., 2023) offers a web interface without an open evaluation backend. As a result, the community lacks a unified, maintained framework for both evaluating and demonstrating StreamST systems.

To address this gap, we propose `simulstream`, the first open-source (Apache 2.0 License) tool for unified evaluation and demonstration of StreamST systems. Our contributions are as follows:

- We introduce `simulstream`, the first unified open-source framework for long-form generation and evaluation of StreamST systems under both incremental and re-translation strategies, with accurate tracking of emitted and deleted tokens.
- We provide a lightweight proxy interface to the community-standard SimulEval framework, supporting direct reuse of existing systems without modification.
- We release an interactive web-based demonstration platform for real-time visualization and system comparison.[3]

Beyond enabling reproducible benchmarking, `simulstream` establishes a common infrastructure for studying and comparing different StreamST decoding paradigms under controlled and consistent conditions. This unified setup has supported its adoption as the official evaluation toolkit for the Simultaneous Translation track of the IWSLT 2026 Evaluation Campaign (Adelani et al., 2026).

2 The `simulstream` Tool

`simulstream` provides a WebSocket server and supporting utilities for running streaming speech processing experiments and demonstrations. It supports real-time transcription and translation over streaming audio input. In this context, streaming assumes a continuous, unbounded speech signal, rather than a sequence of short, pre-segmented utterances as in earlier simultaneous speech processing evaluations (e.g., SimulEval). The short-form setting can be emulated by splitting the input audio into small consecutive segments.

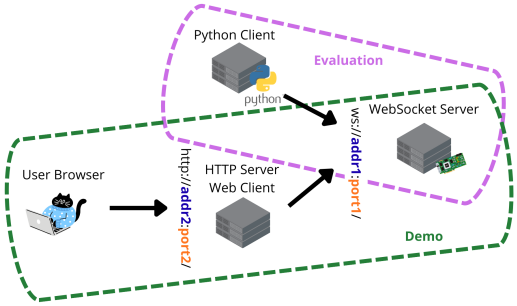


Figure 1: Architecture of the `simulstream` tool.

2.1 Architecture

The package is based on a full-duplex WebSocket client-server interaction, shown in Figure 1. The server waits for connections from the clients. The clients can send configuration messages in JSON format (e.g., to specify the input/output desired languages) and audio chunks, encoded as 16-bit integers. The server processes the input audio and sends to the client JSONs that contain the transcript/translation generated by the configured *speech processor*, which implements the logic to generate outputs from incoming audio, as in SimulEval agents. In addition, if configured to do so, the server writes metric logs into a JSONL file, reporting generated (and deleted) text at each step together with computational costs and total audio processed, which can be used to compute metrics.

The codebase is organized into 3 main blocks: the server, the clients, and the evaluation (see §2.3).

Server. The WebSocket server is launched via the `simulstream_server` command, which accepts two YAML configuration files. The first defines server-level settings (e.g., IP/DNS address and

[2] SimulEval has been archived on September 18th, 2025.
[3] Demo: https://youtu.be/k0WVOIMZFY8

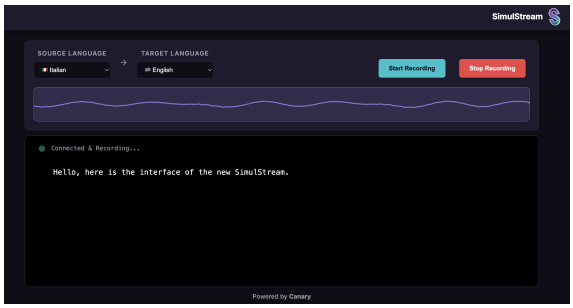


Figure 2: Screenshot of the web interface.

port), while the second specifies the *speech processor* (see §2.2). The server creates a fixed-size pool of processors to support concurrent client execution. Incoming connections exceeding the pool capacity are rejected, providing explicit control over concurrency and preventing resource exhaustion and out-of-memory errors under high load.

Clients. The tool comes with two clients, an HTTP web server and a command-line WAV client, meant, respectively, for demo and evaluation of a speech processor. The HTTP web server provides an HTML/CSS/Javascript web interface interacting with the WebSocket server (see Figure 2). The command-line WAV client, instead, gets as argument a file containing a list of WAV files to stream to the server and it can be used to produce the metric logs necessary for evaluating a speech processor. To further ease experimenting and evaluating speech processors, `simulstream` includes a dedicated command that processes a list of audios without setting up a client-server interaction and writes the metric logs into a specified file.

2.2 Speech Processors

`simulstream` also comes with several streaming speech processors, which can also be used as examples to build custom ones.

- Sliding window (Sen et al., 2022): a speech processor that applies a fixed-length sliding-window retranslation with deduplication to mitigate overlapping outputs when processing unsegmented audio streams. The approach relies on detecting the longest common subsequence between the current window and the previous one, in order to prevent repeating tokens caused by overlapping audio windows. This approach is implemented for Canary v2 (Sekoyan et al., 2025), Seamless (Communication et al., 2023), and any speech-to-text HuggingFace model.
- StreamAtt (Papi et al., 2024): a speech processor that leverages audio-textual alignments based on cross-attention scores. This mechanism serves two purposes: *i)* identifying which part of the generated hypothesis should be emitted (i.e., applying the AlignAtt policy; Papi et al. 2023), and *ii)* determining which part of the audio history to retain in memory, based on the textual history (i.e., the previously generated outputs), thereby enabling long-form processing. This approach is implemented for Seamless and Canary, but can be ported to any speech-to-text attention-based HuggingFace model.
- SimulEval Agent Wrapper: a wrapper processor that calls a SimulEval-based agent.[4] This enables seamless porting of any existing system developed using the legacy tool.
- VAD Wrapper: speech processor that integrates Voice Activity Detection (VAD), specifically the Silero VAD (Team, 2024). Its function is to split continuous audio streams into speech chunks by filtering out parts that do not contain speech before passing the remaining segments to an underlying speech processor.

The repository is structured to be easily extended with custom speech processors. To this aim, users should create a subclass of `simulstream.server.speech_processors.SpeechProcessor`, add the class to the `PYTHONPATH`, and reference it in the YAML file configuration of the speech processor. Subclasses must implement methods to load models, process audio chunks, set source/target languages, and clear internal states. Speech chunks are fed as 1D NumPy arrays containing PCM audio normalized to the range $[-1.0, 1.0]$ sampled at 16 kHz. Their size (in seconds) is determined by the speech processor itself through a dedicated method and the speech processor has to return incremental outputs, containing the tokens to be deleted from previously generated outputs (if any) and the new tokens to be emitted.

2.3 Evaluation

The evaluation of StreamST systems is typically framed as a trade-off between latency and transla-

[4] It supports agents implemented for SimulEval>=1.1.0.

tion quality. `simulstream` provides built-in metrics for both dimensions, while also exposing a simple interface for integrating custom or additional evaluation measures. Moreover, evaluations operate directly on fine-grained metric logs that record the full behavior of a speech processor during inference. As a result, new metrics can be computed a posteriori without requiring additional information, and can be applied to any previously executed run.

Quality. Most simultaneous and streaming ST work has relied solely on BLEU (Papineni et al., 2002), despite growing evidence that it is suboptimal compared to modern neural metrics with higher correlation to human judgments (Freitag et al., 2022). Accordingly, `simulstream` implements both BLEU (computed via sacreBLEU (Post, 2018)) and COMET[5] (Rei et al., 2020). The scores are computed after a re-segmentation of the full generated text for an audio to match the segmentation of the references, which are assumed to be segmented at the sentence level. This is achieved using `mweralign` (Post and Hoang, 2025), a Python implementation of the widespread `mwerSegmenter` binary executable (Matusov et al., 2005) adopted in ST evaluation campaigns (Abdulmumin et al., 2025) and previous work on StreamST (Papi et al., 2024). For streaming processors that may revise or delete previously emitted tokens (e.g., sliding-window re-translation), evaluation is performed on the final output sequence only, while intermediate revised hypotheses are not considered.

Latency. `simulstream` includes a reimplementation of the StreamLAAL metric (Papi et al., 2024) for latency evaluation. The main difference is the use of `mweralign` instead of `mwerSegmenter` for automatic resegmentation of generated text against sentence-level references. StreamLAAL assumes a monotonic word-level alignment between system output and reference. Latency is then computed as the difference between the emission time of each generated word and its expected reference time, where reference words are assumed to be uniformly distributed over the utterance duration (e.g., for a 5-second utterance with 10 words, each word is assigned a 0.5-second timestamp). The emission time of a generated word can be defined in two ways: the *ideal* (or *computational-unaware*) time, corresponding to the amount of audio processed when the word is produced, or the *computational-aware* (CA) time, which additionally accounts for model execution time. StreamLAAL refers to the former, while $\text{StreamLAAL}_{\text{CA}}$ denotes the latter. Since StreamLAAL was originally designed for systems that do not revise previously emitted tokens, we extend its definition to retranslation processors by computing latency on the final output sequence, consistent with the quality evaluation. Specifically, for each word, we use its last update time as the emission time. This design choice may overestimate latency compared to user experience in retranslation systems, while boosting the translation quality as it does not account for intermediate—possibly wrong—outputs. Lastly, for character-level languages such as Chinese, Japanese, and Korean, the latency unit is represented by a single character, similarly to SimulEval.

Other statistics. `simulstream` additionally computes statistics that complement the evaluation metrics defined above. Specifically, it enables computing *normalized erasure* (NE) and the *real time factor* (RTF). NE (Arivazhagan et al., 2020a) measures flickering in retranslation. It is defined as the ratio between the number of tokens that have been deleted and the number of final generated tokens. RTF, instead, is measured as seconds spent in computation for each input audio second. RTF values greater than 1 indicate that the system is not able to process the input in time before the next input arrives. As for the metrics above, the system modularity enables easy integration of additional/customized statistics.

3 Experiments

3.1 Setup

We demonstrate the utility of our tool by comparing the released speech processors (§2.2). Specifically, for both Canary v2 and Seamless medium v1,[6] we include *i)* retranslation sliding window, *ii)* the VAD wrapper with retranslation sliding window as underlying processor, and *iii)* incremental StreamAtt. For the sliding window and VAD wrapper, we slide the window by 2s.[7] For the sliding window approach, we vary the window length (8, 10, 12, 14 seconds) to obtain different operating points in terms of quality and latency and maintain the VAD

[5] The default model is `Unbabel/wmt22-comet-da`, but alternative models can be specified via a command-line parameter.

[6] We also experimented with Seamless large v2 but obtained inferior results (see Appendix A).

[7] We tested 1s, 2s, and 3s on the dev set and chose 2s as it gave the best quality/latency trade-off.

Speech P.		Model	w/f/t	COMET ↑	BLEU ↑	StreamLAAL ↓	StreamLAAL$_{CA}$ ↓	NE ↓	RTF ↓
retranslation	Sliding Window	Canary	8	0.7853	28.71	2.47	2.80	0.5771	0.1438
			10	0.7883	28.48	3.01	3.43	0.8049	0.1865
			12	0.7957	28.92	3.40	3.93	0.9475	0.2315
			14	**0.7986**	**29.20**	3.93	4.47	1.1277	0.2405
		SeamlessM4T	8	0.7174	24.51	2.42	2.80	0.8586	0.1770
			10	0.7359	25.39	2.92	3.38	1.0215	0.2101
			12	0.7439	25.70	3.53	4.07	1.1778	0.2432
			14	0.7469	25.65	4.26	4.90	1.3401	0.2878
	VAD + Sliding Window	Canary	0.6	0.7375	24.70	2.54	2.78	0.1304	**0.0822**
			0.5	0.7426	25.01	2.57	2.82	0.1393	0.0834
			0.4	0.7464	25.26	2.57	2.83	0.1488	0.0854
			0.3	0.7523	25.57	2.61	2.87	0.1631	0.0866
		SeamlessM4T	0.6	0.6949	21.97	2.53	2.81	0.1631	0.0939
			0.5	0.6972	22.16	2.56	2.84	0.1709	0.0949
			0.4	0.7008	22.49	2.63	2.92	0.1842	0.0973
			0.3	0.7058	22.98	2.74	3.04	0.2060	0.1008
incremental	StreamAtt	Canary	2	0.7365	17.62	2.89	3.52	**0.0000**	0.5479
			4	0.7370	17.78	2.92	3.59	**0.0000**	0.5791
			6	0.7422	18.23	3.26	4.04	**0.0000**	0.6315
			8	0.7452	18.26	3.53	4.21	**0.0000**	0.5162
		SeamlessM4T	2	0.7589	26.42	**1.95**	**2.23**	**0.0000**	0.2196
			4	0.7653	27.25	2.25	2.53	**0.0000**	0.2328
			6	0.7695	27.80	2.56	2.85	**0.0000**	0.2447
			8	0.7696	27.94	2.84	3.13	**0.0000**	0.2581

Table 1: Quality (COMET, BLUE), latency (StreamLAAL, StreamLAAL_CA), flickering (NE), and computational cost (RTF) of the supported retranslation and incremental speech processors (**Speech P.**) with Canary and SeamlessM4T v1 medium on the MuST-C test set (averaged over all language pairs). The column w/f/t refers to varying, the window length (w) for sliding window, the number of frames (f) for StreamAtt, and the VAD probability threshold (t) for VAD-based sliding window.

probability threshold fixed at 0.1. For the VAD wrapper, instead, we use 14s as window lenght and we vary the VAD threshold from 0.3 to 0.6, which controls whether to be more conservative (lower values) or more aggressive (higher values) in considering audio as non-speech and filtering it out. Lastly, for StreamAtt, we use speech chunks of 1s, and we vary the cutoff frame with 2, 4, 6, and 8 to get different latency-quality trade-offs. A single NVIDIA A40 40GB is used for all the inferences.

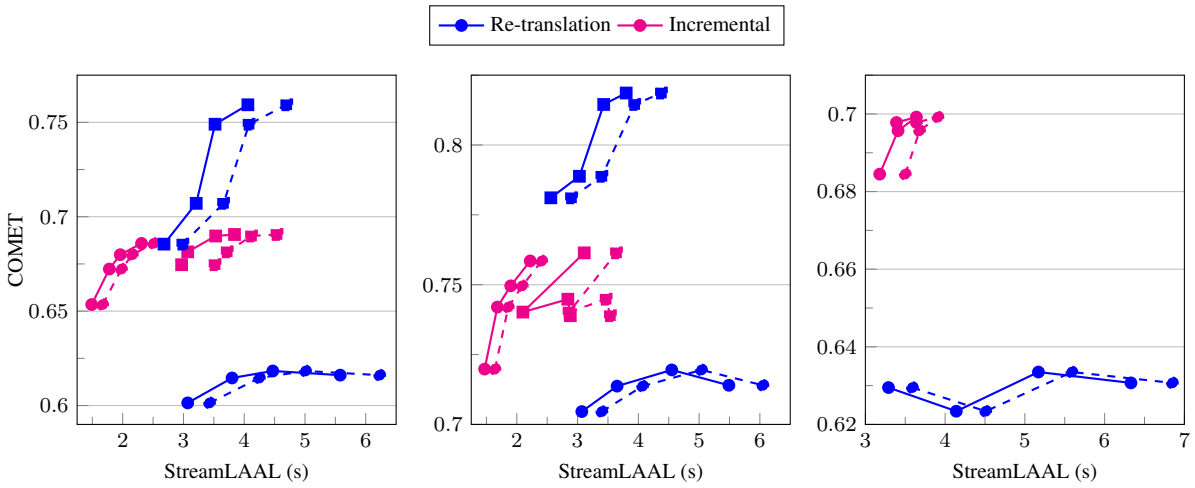
As data, we leverage the 8 language pairs (en→de,es,fr,it,nl,pt,ro,ru) of the widespread MuST-C dataset (Di Gangi et al., 2019) and we process the entire TED talks in the test set (each lasting ∼10 minutes). As an additional test set, we employ MCIF (Papi et al., 2026), consisting of scientific presentations of ∼5-6 minutes each covering en→de,it,zh and featuring additional difficulties compared to MuST-C including different recording qualities, accents, and domain-specific terminology. We do not rely on popular ST corpora, e.g., CoVoST2 (Wang et al., 2020) and FLEURS (Conneau et al., 2023), as they contain sentence-level audios (lasting few seconds), which are not representative of real use cases of long audio streams.

3.2 Results

Table 1 reports the scores obtained by the different speech processors on MuST-C averaged over all the 8 language pairs.[8] By comparing the retranslation approaches with the two models (Canary and SeamlessM4T), Canary emerges as the best performing one by a large margin with analogous trends both with and without VAD. Although their latency (StreamLAAL) is similar, the quality (COMET, SacreBLEU) of Canary's translations is significantly better than that of SeamlessM4T. Canary also displays a lower amount of flickering (NE) and computational costs (RTF).

Moving to the comparison between the pure sliding window approach and the one preceded by the VAD wrapper, the behavior is consistent for both models. The flickering and computational costs are dramatically reduced (flickering is roughly 8× smaller, and RTF 2-3× lower). On the downside,

[8] Results for each language pair are available at https://github.com/hlt-mt/simulstream/blob/main/examples/must_results/README.md.



(a) en-de (b) en-it (c) en-zh

Figure 3: Latency (StreamLAAL↓) - Quality (COMET↑) curves of Sliding-window retranslation and StreamAtt incremental methods applied to SeamlessM4T v1 medium (●) and Canary v2 (■) on the three target languages of MCIF (except for Canary, which does not support Chinese). Dashed lines indicate computationally aware latency, while solid lines computationally unaware. Numerical results are presented in Appendix B.

there is a reduced possibility of controlling the latency-quality tradeoff (with the VAD wrapper, all the operational points are close to each other, with latency differences spanning 0.2s and quality less than 0.015 COMET) and, most importantly, the translation quality suffers a huge drop.

Lastly, we compare incremental decoding with StreamAtt against retranslation using the best performing sliding-window strategy. To the best of our knowledge, this is the first direct comparison between incremental and retranslation approaches, made possible by the unified implementation and evaluation framework provided by `simulstream`. Notably, although our evaluation metrics are designed to penalize latency while favoring the quality of retranslation methods (see §2.3), StreamAtt on Seamless surprisingly outperforms the sliding-window approach not only in latency but also in translation quality, while completely eliminating flickering effects. The main drawback is its higher computational cost (RTF), particularly in low-latency settings, although still far from reaching the critical value of 1.0. In contrast, Canary exhibits the opposite behavior, with the sliding window achieving substantial quality gains at comparable or even lower latency than StreamAtt.

The results on MCIF (Figure 3) further confirm this trend: retranslation consistently achieves the highest translation quality with Canary, whereas StreamAtt yields the lowest latency with Seamless. Overall, no single decoding strategy emerges as universally superior across models, indicating that the optimal quality–latency trade-off is inherently model- and application-dependent.[9] These findings highlight the importance of `simulstream` as a unified framework for fair, systematic, and reproducible comparison of simultaneous decoding strategies, supporting informed design choices for future streaming speech translation systems.

4 Conclusions

We presented `simulstream`, the first open-source framework enabling unified evaluation and demonstration of Streaming Speech-to-Text Translation (StreamST) systems. Unlike previous tools, `simulstream` supports both incremental decoding and retranslation, tracks token emissions and deletions, and accommodates long-form audio, providing a comprehensive platform for assessing translation quality, latency, and computational costs. In addition, its interactive web interface allows real-time visualization and comparison of different StreamST systems, bridging the gap between research and practical deployment. By offering a paradigm-agnostic, flexible, and extendable toolkit, `simulstream` empowers the community to systematically study, benchmark, and showcase modern StreamST approaches, facilitating the advancement of simultaneous ST research.

[9] For instance, incremental decoding remains essential for applications such as cascaded speech-to-speech translation systems (Bach et al., 2009; Sudoh et al., 2020).

Acknowledgments

We acknowledge the support of the project InnovAction: Network Italiano dei Centri per l'Innovazione Tecnologica (CUP B47H2200437000), funded by MIMIT with NPRR - NextGenerationEU funds, in collaboration with Piazza Copernico S.r.l. This paper has received funding from the European Union's Horizon Europe programme grant agreement No. 101213369 (project DVPS).

References

Idris Abdulmumin, Victor Agostinelli, Tanel Alumäe, Antonios Anastasopoulos, Luisa Bentivogli, Ondřej Bojar, Claudia Borg, Fethi Bougares, Roldano Cattoni, Mauro Cettolo, Lizhong Chen, William Chen, Raj Dabre, Yannick Estève, Marcello Federico, Mark Fishel, Marco Gaido, Dávid Javorský, Marek Kasztelnik, and 33 others. 2025. Findings of the IWSLT 2025 evaluation campaign. In *Proceedings of the 22nd International Conference on Spoken Language Translation (IWSLT 2025)*, pages 412–481, Vienna, Austria (in-person and online). Association for Computational Linguistics.

David Ifeoluwa Adelani, Victor Agostinelli, Antonios Anastasopoulos, Luisa Bentivogli, Ondřej Bojar, Sébastien Bratières, Marine Carpuat, Fabrício Carraro, Roldano Cattoni, Mauro Cettolo, Lizhong Chen, Marcello Federico, Marco Gaido, Mahendra Gupta, HyoJung Han, Ali Hatami, Lewis C. Howe, Dávid Javorský, Yejin Jeon, and 36 others. 2026. Speech translation and metrics in 2026: Findings of the IWSLT campaign. In *Proceedings of the 23rd International Conference on Spoken Language Translation (IWSLT 2026)*, pages 336–422, San Diego, USA (in-person and online). Association for Computational Linguistics.

Ebrahim Ansari, Ondřej Bojar, Barry Haddow, and Mohammad Mahmoudi. 2021. SLTEV: Comprehensive evaluation of spoken language translation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 71–79, Online. Association for Computational Linguistics.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, and George Foster. 2020a. 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.

Naveen Arivazhagan, Colin Cherry, I Te, Wolfgang Macherey, Pallavi Baljekar, and George Foster. 2020b. Re-translation strategies for long form, simultaneous, spoken language translation. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7919–7923.

Nguyen Bach, Roger Hsiao, Matthias Eck, Paisarn Charoenpornsawat, Stephan Vogel, Tanja Schultz, Ian Lane, Alex Waibel, and Alan W. Black. 2009. Incremental adaptation of speech-to-speech translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, NAACL-Short '09, page 149–152, USA. Association for Computational Linguistics.

Seamless Communication, Loïc Barrault, Yu-An Chung, Mariano Cora Meglioli, David Dale, Ning Dong, Paul-Ambroise Duquenne, Hady Elsahar, Hongyu Gong, Kevin Heffernan, John Hoffman, Christopher Klaiber, Pengwei Li, Daniel Licht, Jean Maillard, Alice Rakotoarison, Kaushik Ram Sadagopan, Guillaume Wenzek, Ethan Ye, and 49 others. 2023. Seamlessm4t: Massively multilingual & multimodal machine translation. *Preprint*, arXiv:2308.11596.

Alexis Conneau, Min Ma, Simran Khanuja, Yu Zhang, Vera Axelrod, Siddharth Dalmia, Jason Riesa, Clara Rivera, and Ankur Bapna. 2023. Fleurs: Few-shot learning evaluation of universal representations of speech. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pages 798–805. IEEE.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, and Stephan Vogel. 2018. Incremental decoding and training methods for simultaneous translation in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 493–499, New Orleans, Louisiana. Association for Computational Linguistics.

Mattia A. Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2019. 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.

Markus Freitag, Ricardo Rei, Nitika Mathur, Chi-kiu Lo, Craig Stewart, Eleftherios Avramidis, Tom Kocmi, George Foster, Alon Lavie, and André F. T. Martins. 2022. Results of WMT22 metrics shared task: Stop using BLEU – neural metrics are better and more robust. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 46–68, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Christian Fügen, Alex Waibel, and Muntsin Kolss. 2008. Simultaneous translation of lectures and speeches. *Machine Translation*, 21(4):209–252.

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, Doha, Qatar. Association for Computational Linguistics.

Christian Huber, Tu Anh Dinh, Carlos Mullov, Ngoc-Quan Pham, Thai Binh Nguyen, Fabian Retkowski, Stefan Constantin, Enes Ugan, Danni Liu, Zhaolin Li, Sai Koneru, Jan Niehues, and Alexander Waibel. 2023. End-to-end evaluation for low-latency simultaneous speech translation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 12–20, Singapore. Association for Computational Linguistics.

Javier Iranzo-Sánchez, Javier Jorge, Pau Baquero-Arnal, Joan Albert Silvestre-Cerdà, Adrià Giménez, Jorge Civera, Albert Sanchis, and Alfons Juan. 2021. Streaming cascade-based speech translation leveraged by a direct segmentation model. *Neural Networks*, 142:303–315.

Xutai Ma, Mohammad Javad Dousti, Changhan Wang, Jiatao Gu, and Juan Pino. 2020. 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.

Evgeny Matusov, Gregor Leusch, Oliver Bender, and Hermann Ney. 2005. Evaluating machine translation output with automatic sentence segmentation. In *Proceedings of the Second International Workshop on Spoken Language Translation*, Pittsburgh, Pennsylvania, USA.

Siqi Ouyang, Xi Xu, and Lei Li. 2025. InfiniSST: Simultaneous translation of unbounded speech with large language model. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 3032–3046, Vienna, Austria. Association for Computational Linguistics.

Sara Papi, Marco Gaido, Matteo Negri, and Luisa Bentivogli. 2024. StreamAtt: Direct streaming speech-to-text translation with attention-based audio history selection. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3692–3707, Bangkok, Thailand. Association for Computational Linguistics.

Sara Papi, Peter Polák, Dominik Macháček, and Ondřej Bojar. 2025. How "real" is your real-time simultaneous speech-to-text translation system? *Transactions of the Association for Computational Linguistics*, 13:281–313.

Sara Papi, Marco Turchi, and Matteo Negri. 2023. Alignatt: Using attention-based audio-translation alignments as a guide for simultaneous speech translation. In *Interspeech 2023*, pages 3974–3978.

Sara Papi, Maike Züfle, Marco Gaido, Beatrice Savoldi, Danni Liu, Ioannis Douros, Luisa Bentivogli, and Jan Niehues. 2026. MCIF: Multimodal crosslingual instruction-following benchmark from scientific talks. In *The Fourteenth 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 of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Peter Polák, Sara Papi, Luisa Bentivogli, and Ondřej Bojar. 2025. Better late than never: Evaluation of latency metrics for simultaneous speech-to-text translation. *arXiv preprint arXiv:2509.17349*.

Peter Polák and Ondřej Bojar. 2024. Long-form end-to-end speech translation via latent alignment segmentation. In *2024 IEEE Spoken Language Technology Workshop (SLT)*, pages 1076–1082.

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, Brussels, Belgium. Association for Computational Linguistics.

Matt Post and Hieu Hoang. 2025. Effects of automatic alignment on speech translation metrics. In *Proceedings of the 22nd International Conference on Spoken Language Translation (IWSLT 2025)*, pages 84–92, Vienna, Austria (in-person and online). Association for Computational Linguistics.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

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.

Felix Schneider and Alexander Waibel. 2020. Towards stream translation: Adaptive computation time for simultaneous machine translation. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 228–236, Online. Association for Computational Linguistics.

Monica Sekoyan, Nithin Rao Koluguri, Nune Tadevosyan, Piotr Zelasko, Travis Bartley, Nikolay Karpov, Jagadeesh Balam, and Boris Ginsburg. 2025.

Canary-1b-v2 & parakeet-tdt-0.6b-v3: Efficient and high-performance models for multilingual asr and ast. *Preprint*, arXiv:2509.14128.

Sukanta Sen, Ondřej Bojar, and Barry Haddow. 2022. Simultaneous translation for unsegmented input: A sliding window approach. *Preprint*, arXiv:2210.09754.

Katsuhito Sudoh, Takatomo Kano, Sashi Novitasari, Tomoya Yanagita, Sakriani Sakti, and Satoshi Nakamura. 2020. Simultaneous speech-to-speech translation system with neural incremental asr, mt, and tts. *Preprint*, arXiv:2011.04845.

Silero Team. 2024. Silero vad: pre-trained enterprise-grade voice activity detector (vad), number detector and language classifier. https://github.com/snakers4/silero-vad.

Changhan Wang, Anne Wu, and Juan Pino. 2020. Covost 2 and massively multilingual speech-to-text translation. *arXiv preprint arXiv:2007.10310*.

Model	w/f	COMET ↑	BLEU ↑	StreamLAAL ↓	$\text{StreamLAAL}_{\text{CA}}$ ↓	NE ↓	RTF ↓
SeamlessM4T v2 large	8	0.7016	21.72	3.29	4.14	0.7062	0.4417
	10	0.7089	23.31	3.57	4.55	0.8455	0.4505
	12	0.7074	22.53	4.44	5.52	0.9516	0.4903
	14	0.7005	20.02	6.35	7.58	1.0316	0.6101
SeamlessM4T v1 medium	8	0.7174	24.51	2.42	2.80	0.8586	0.1770
	10	0.7359	25.39	2.92	3.38	1.0215	0.2101
	12	0.7439	25.70	3.53	4.07	1.1778	0.2432
	14	0.7469	25.65	4.26	4.90	1.3401	0.2878

Table 2: Quality (COMET, BLUE), latency (StreamLAAL, StreamLAAL_CA), flickering (NE), and computational cost (RTF) of the sliding window processor using SeamlessM4T v2 large and SeamlessM4T v1 medium, averaged across the 8 language pairs of MuST-C.

A Comparison with SeamlessM4T v2 Large

Table 2 reports the scores for SeamlessM4T v2 large and SeamlessM4T v1 medium. Interestingly, SeamlessM4T v1 medium outperforms by a large margin the v2 large version on all metrics – including those related to output quality – but normalized erasure. The latency is higher not only due to the (expected) higher computational cost, evident from the high RTF, but also the ideal StreamLAAL is notably worse. For this reason, the results in the main paper are reported using the medium v1 version.

B Numerical Results on MCIF

Numerical results for Figure 3 are presented in Table 3 in the next page.

Speech P.	Model	w/f	COMET ↑	BLEU ↑	StreamLAAL ↓	StreamLAAL$_{CA}$ ↓	NE ↓	RTF ↓
				en-de				
incremental	SeamlessM4T	2	0.6535	19.48	1.49	1.67	0.0000	0.1625
		4	0.6723	22.17	1.78	1.98	0.0000	0.1780
		6	0.6799	22.86	1.96	2.16	0.0000	0.1906
		8	0.6859	23.82	2.31	2.51	0.0000	0.2009
	Canary	2	0.6746	17.43	2.97	3.52	0.0000	0.4689
		4	0.6814	17.40	3.07	3.72	0.0000	0.5221
		6	0.6898	17.99	3.53	4.12	0.0000	0.5097
		8	0.6906	18.01	3.84	4.54	0.0000	0.5825
retranslation	SeamlessM4T	8	0.6014	18.79	3.07	3.43	0.6830	0.1757
		10	0.6146	18.07	3.80	4.25	0.8009	0.2133
		12	0.6183	17.76	4.47	5.01	0.9323	0.2474
		14	0.6161	16.16	5.58	6.23	0.9987	0.2858
	Canary	8	0.6855	26.01	2.68	2.99	0.7096	0.1571
		10	0.7071	25.67	3.21	3.66	0.8658	0.2135
		12	0.7490	30.19	3.52	4.08	1.0960	0.2595
		14	0.7593	30.02	4.06	4.70	1.2239	0.2918
				en-it				
incremental	SeamlessM4T	2	0.7198	27.59	1.47	1.64	0.0000	0.1474
		4	0.7420	31.84	1.68	1.86	0.0000	0.1595
		6	0.7496	33.56	1.90	2.09	0.0000	0.1689
		8	0.7585	34.79	2.22	2.41	0.0000	0.1784
	Canary	2	0.7389	23.55	2.88	3.54	0.0000	0.5525
		4	0.7448	25.71	2.84	3.47	0.0000	0.5083
		6	0.7402	24.70	2.10	2.86	0.0000	0.5554
		8	0.7614	26.60	3.11	3.64	0.0000	0.4893
retranslation	SeamlessM4T	8	0.7046	31.38	3.07	3.40	0.7313	0.1614
		10	0.7137	31.06	3.65	4.07	0.9122	0.1901
		12	0.7195	30.61	4.55	5.04	1.0497	0.2218
		14	0.7140	29.06	5.49	6.05	1.1904	0.2561
	Canary	8	0.7811	37.96	2.56	2.90	0.6294	0.1673
		10	0.7888	37.59	3.03	3.40	0.7740	0.1824
		12	0.8145	43.18	3.43	3.94	1.0217	0.2376
		14	0.8186	43.81	3.80	4.38	1.1862	0.2701
				en-zh				
incremental	SeamlessM4T	2	0.6845	27.35	3.18	3.50	0.0000	0.2121
		4	0.6957	28.78	3.41	3.68	0.0000	0.1884
		6	0.6992	28.78	3.64	3.91	0.0000	0.1942
		8	0.6978	28.85	3.39	3.64	0.0000	0.1899
retranslation	SeamlessM4T	8	0.6295	25.02	3.29	3.59	0.7956	0.1423
		10	0.6234	24.62	4.14	4.51	1.0344	0.1690
		12	0.6335	24.87	5.17	5.59	1.1488	0.1942
		14	0.6307	23.13	6.33	6.85	1.2302	0.2215

Table 3: MCIF numerical results across speech processors (StreamAtt for incremental, and Sliding Window for retranslation), language pairs, and models.