

A SimulMEGA-TTS Implementaion Details

Figure 6 shows how to implement SimulMEGA on top of a uni-directional decoder-only backbone. We take CosyVoice 2 as the base model and fine-tune it into SimulMEGA-TTS. We take the last layer hidden states output of text tokens as the h^{prefix} and the hidden state of the text EOS token as H^{global} .

In S2TT, normally, when the input reaches the end of the stream, we no longer turn to the read/write strategy and instead continue the generation until the EOS token appears. However, in the TTS task, the end of speech is relatively ambiguous. Therefore, if we employ the same strategy as S2TT, the hallucination problem may occur at the end of the generation. Therefore, we employ a mixed strategy to tell the end of the generation:

$$\text{End of Generation} = \begin{cases} \text{True} & \text{if } E_t \in T_{\text{text}} \text{ and } (E_s \in T_{\text{speech}} \text{ or } p_{\cdot,i} > \lambda_{\text{end}}) \\ \text{False} & \text{otherwise} \end{cases} \quad (9)$$

Where $E_{t/s}$ denotes text/speech EOS token. $T_{\text{text/speech}}$ denotes text or speech tokens in the current sequence. $p_{\cdot,i}$ is the router output at the current position after all input streams are read. $\lambda_{\text{end}} = 0.9$.

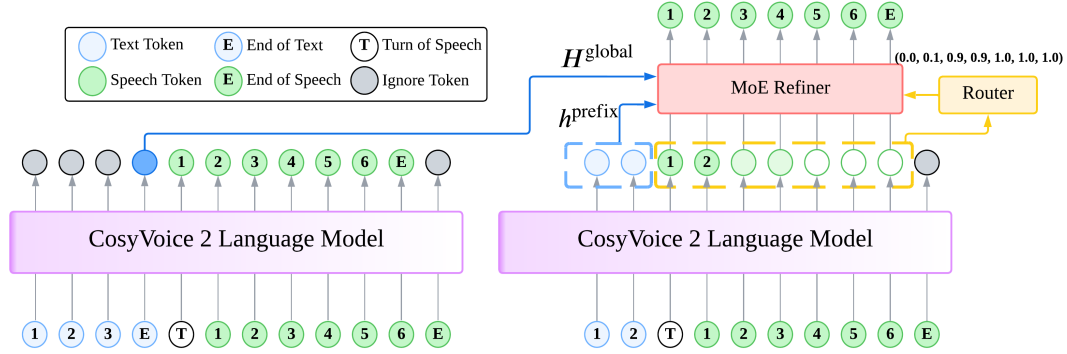


Figure 6: Illustration of SimulMEGA-TTS. The left uses offline text input while the right uses prefix text input for policy learning.

B S2TT computation overhead.

For evaluating the computation overhead, we plot the computation-aware(CL) quality-latency trade-off curve in the CoVoST2 FR-EN testset. Computation-aware latency considers the actual inference time of the model. The evaluation is performed on a fully idle machine with a single Nvidia-H100 GPU. As shown in Figure 7, for SimulMEGA, the extra AL due to computation is around 50 ms, whereas the number for Seamless is around 200 ms.

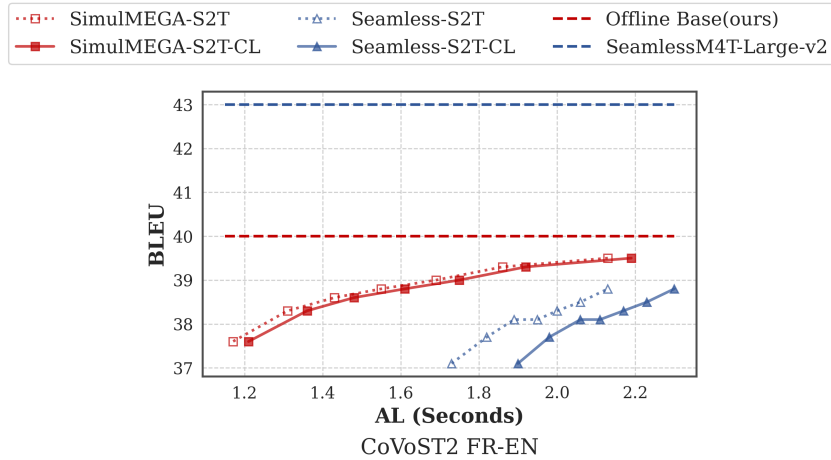


Figure 7: Computation aware quality-latency curve on CoVoST2 FR-EN. CL denotes Computation-Aware.

464 C S2ST System Deployment & Latency.

465 We deploy our S2ST system on an Ubuntu server equipped with NVIDIA H100 GPUs. For the frontend and
 466 backend, we utilize FastRTC³ and FastAPI⁴, respectively. To enhance inference efficiency, vLLM⁵ is used to
 467 accelerate the language model in TTS, and TensorRT⁶ is used to accelerate the flow model.

468 We evaluate the system latency with streaming inputs, using four language pairs (ZH-EN, EN-ZH, FR-EN,
 469 FR-ZH). For each direction, we test two samples, each 20–30 seconds in length, and record event timestamps
 470 for latency calculation. Virtual Audio Cable⁷ software is employed to eliminate echo interference and ensure
 471 consistent audio input across tests. The average latency results are presented in Table 3. Here, the S2T/S2S
 472 Start Offset indicates the time required to generate the first text or speech chunk, while the S2T/S2S End Offset
 473 measures the delay of the final text chunk or speech frame relative to the end of the source speech.

		Offline N	SimulMEGA N	SimulMEGA S	Seamless -
S2TT	Start Offset	13.56	3.12	2.95	3.55
	End Offset	0.78	0.44	0.58	0.61
S2ST	Start Offset	15.98	7.43	3.87	4.33
	End Offset	14.16	7.65	7.54	6.42

Table 3: Simultaneous S2ST system latency statistics (in seconds). N denotes using non-stream TTS and S denotes stream TTS

474 It should be noted that these results provide only a rough estimation of system latency, as actual values may vary
 475 due to server or network conditions, output content length, speech rate, or other factors.

476 D S2ST Examples

477 We visualize two S2ST example of SimulMEGA in FR-EN and ZH-EN, respectively. The second row is the
 478 isochronic text label of source speech and the forth row indicates the timestamps when each target text chunk are
 479 generated.

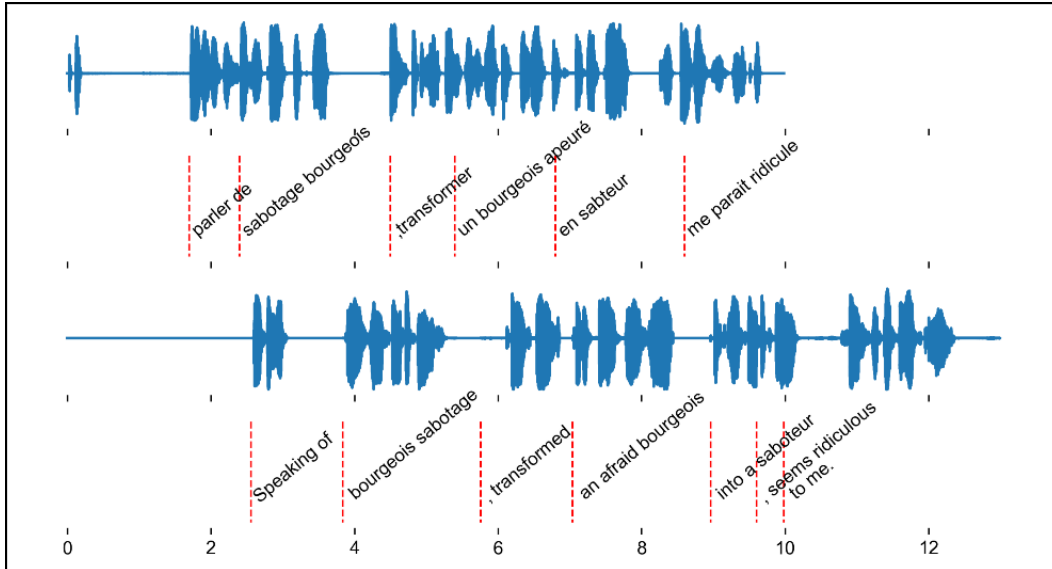


Figure 8: (a) FR-EN

³<https://github.com/gradio-app/fastrtc>

⁴<https://github.com/fastapi/fastapi>

⁵<https://github.com/vllm-project/vllm>

⁶<https://github.com/NVIDIA/TensorRT>

⁷<https://vac.muzychenko.net/en/>

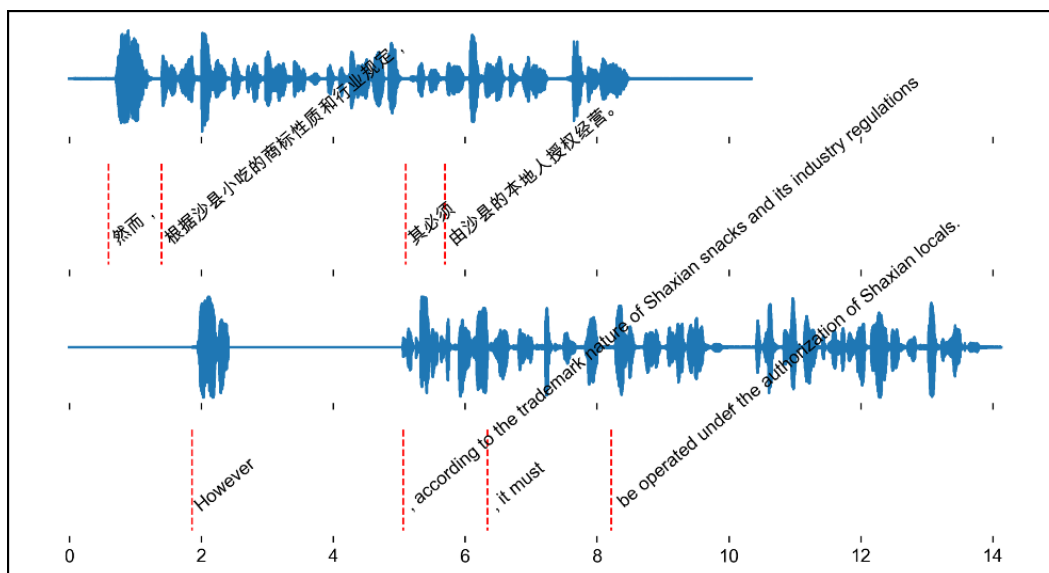


Figure 8: (b) ZH-EN