

# Redefining Machine Simultaneous Interpretation: From Incremental Translation to Human-Like Strategies

**Qianen Zhang**

The Chinese University of Hong Kong,  
Shenzhen  
Shenzhen, Guangdong, China  
qianenzhang@link.cuhk.edu.cn

**Zeyu Yang**

The Chinese University of Hong Kong,  
Shenzhen  
Shenzhen, Guangdong, China  
zeyuyang1@link.cuhk.edu.cn

**Satoshi Nakamura**

The Chinese University of Hong Kong,  
Shenzhen  
Shenzhen, Guangdong, China  
Nara Institute of Science  
and Technology  
Nara, Japan  
snakamura@cuhk.edu.cn

## Abstract

Simultaneous Machine Translation (SiMT) requires high-quality translations under strict real-time constraints, which traditional policies with only READ/WRITE actions cannot fully address. We extend the action space of SiMT with four adaptive actions: **Sentence\_Cut**, **Drop**, **Partial\_Summarization** and **Pronominalization**, which enable real-time restructuring, omission, and simplification while preserving semantic fidelity. We adapt these actions in a large language model (LLM) framework and construct training references through action-aware prompting. To evaluate both quality and word-level monotonicity, we further develop a latency-aware TTS pipeline that maps textual outputs to speech with realistic timing. Experiments on the ACL60/60 English-Chinese, English-German and English-Japanese benchmarks show that our framework consistently improves semantic metrics and achieves lower delay compared to reference translations and salami-based baselines. Notably, combining **Drop** and **Sentence\_Cut** leads to consistent improvements in the balance between fluency and latency. These results demonstrate that enriching the action space of LLM-based SiMT provides a promising direction for bridging the gap between human and machine interpretation.<sup>1</sup>

## 1 Introduction

Simultaneous speech translation requires real-time translation with high quality, which poses unique

<sup>1</sup>Our code is available at <https://github.com/Cadente1/Action-based-Simultaneous-Machine-Translation>.

challenges compared to offline machine translation (MT) due to the incompleteness of information. While traditional MT systems generate fluent and accurate translations by relying on complete source sentences, such a paradigm is unsuitable for simultaneous machine translation (SiMT), where incremental processing and low latency are mandatory. The central bottleneck of SiMT therefore lies in maintaining an optimal balance between translation quality and latency. To address this challenge, the system must be capable of deciding both *when* and *how* to translate under partial input.

A prospective approach is to learn from professional human interpreters, who employ adaptive strategies such as rephrasing, summarization, and selective omission to cope with simultaneity constraints while preserving the core meaning of the source speech (Doi et al., 2021). For instance, the widely adopted *salami technique* breaks the sentences into minimal segments that contains enough information for translation. This segmentation effectively reduces long-distance word reordering caused by syntactic divergence across languages, while preserving semantic fidelity. By applying this technique to SiMT, the system benefits from the monotonicity and improved word-order alignment (Makinae et al., 2024).

Most existing work on SiMT builds on traditional machine translation systems, where translation is generated incrementally and the quality–latency trade-off is controlled by READ/WRITE policies. Such formulations restrict the space of translation behaviors and struggle to capture higher-level strategies used by

human interpreters, such as sentence segmentation, selective omission, and partial reordering. This motivates the incorporation of summarization- and omission-oriented operations into SiMT modeling. In this work, we adopt large language models (LLMs) as the translation backbone and exploit their capacity for partial summarization, flexible reordering, and discourse-level rewriting to generate translations that more closely resemble human interpreter output under simultaneous constraints.

Another obstacle lies in training data. Most SiMT systems adopt offline translations as references. Although such translations are fluent and semantically faithful, they are unsuitable as references for SiMT because of different generation patterns. Training on them biases models toward waiting for complete input, thus increasing latency and contradicting the real-time requirement. As a result, it is crucial to generate high-quality reference interpretations that align with simultaneous interpretation patterns while preserving semantic fidelity.

In this study, we propose a LLM-based framework for simultaneous machine interpretation that extends the conventional READ/WRITE paradigm with four adaptive actions: **Sentence\_Cut**, **Partial\_Summarization**, **Drop**, and **Pronominalization**. By expanding the action space, our approach shifts the focus of SiMT from purely incremental token emission to the explicit modeling of human-like interpretation strategies, while remaining within the standard SiMT formulation.

Our framework is built on decoder-only large language models such as GPT-4o and Qwen3-8B, and is evaluated against strong baselines including salami-based segmentation, the LLM-based TransLLaMA system, and prompting-based methods such as few-shot prompting. Although our experiments are conducted in a text-to-text setting, we view the proposed method as modeling the decoder component of a full SiMT system. Speech input can be incorporated through an upstream encoder in future work.

Our main contributions are as follows.

- We propose a LLM-based SiMT framework introducing four novel actions (**Sentence\_Cut**, **Partial\_Summarization**, **Drop** and **Pronominalization**) integrated into a sequential decision-making process to balance translation quality and latency.

- We adapt offline translations into SiMT-like references using these actions, and produce training data that better reflects real-time interpretation constraints while preserving semantic fidelity.
- We systematically compare multiple LLM-based inference paradigms for SiMT to analyze their respective strengths in balancing translation quality and latency.

## 2 Related work

Current SiMT systems are typically evaluated in terms of translation quality and latency. Translation quality is commonly measured using surface-based metrics such as BLEU (Papineni et al., 2001), chrF (Popović, 2015), and TER (Snover et al., 2006), which rely on n-gram overlap and may undervalue semantically valid but lexically divergent translations. In contrast, neural metrics such as COMET (Rei et al., 2020, 2022) compare meaning using pre-trained multilingual encoders and correlate better with human judgments (Glushkova et al., 2023).

Latency is commonly evaluated using AL (Ma et al., 2019), AP (Cho and Esipova, 2016), DAL (Cherry et al., 2018), and LAAL (Papi et al., 2022), with more recent work proposing ATD to explicitly account for delays caused by long target segments (Kano et al., 2023). Since BLEU depends on offline references and may encourage excessive waiting in SiMT, several recent studies on simultaneous and simulMT evaluation modify offline references or adopt salami-style segmentation to reduce long-distance reordering effects and better reflect real-time interpretation behavior (Chen et al., 2020; Makinae et al., 2024; Zhao et al., 2021).

In terms of architecture, most SiMT systems adopt encoder–decoder frameworks, where a policy determines when to READ source tokens and WRITE target tokens to balance translation quality and latency. Early work explored fixed policies such as wait- $k$  (Ma et al., 2019) and segmentation-based approaches (Oda et al., 2014), while later studies proposed adaptive policies that adjust READ/WRITE decisions based on source content or model predictions (Arivazhagan et al., 2019; Oda et al., 2015). Notably, Oda et al. (2015) introduced one of the earliest syntax-informed approaches by predicting unseen constituents. More recent simulMT research has explored decoder-only architectures and large pretrained language models for incremental and low-latency translation

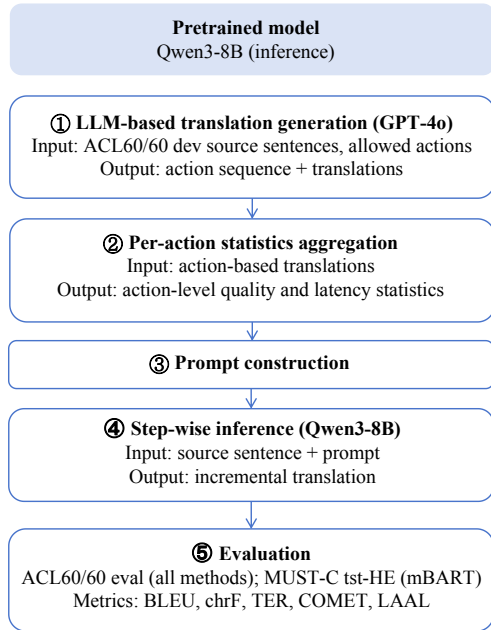


Figure 1: Block diagram of the inference procedure. We first obtain per-action translation BLEU and LAAL statistics by evaluating LLM generated translations of ACL60/60 dev set. These statistics are provided for the LLM, which is prompted to translate the input sentence by choosing an action at each step according to the statistics.

(Guo et al., 2024; Koshkin et al., 2024; Wang et al., 2024).

Distinct from previous work, we propose a SiMT framework based on LLMs empowered by four novel actions (**Sentence\_Cut**, **Partial\_Summarization**, **Drop** and **Pronominalization**) to balance quality and latency by emulating the adaptive strategies of professional interpreters, which we describe below.

### 3 Method

We propose a LLM-based framework for SiMT that formulates generation as a step-wise, action-guided process. The method consists of three main components: (1) an extended action space that abstracts human interpreter strategies beyond the conventional READ/WRITE paradigm, (2) a prompt-based inference procedure that enables a decoder-only LLM to select actions dynamically during generation, and (3) a latency-aware TTS pipeline for evaluating word-level monotonicity in the time domain. Figure 1 provides an overview of the inference procedure, and Figure 2 illustrates the prompt design and step-wise generation process.

#### 3.1 Extended action space

Conventional SiMT policies are limited to two actions: READ and WRITE. Although the optimization of policies can improve quality-latency trade-off by making better decisions of when to commit WRITE actions, they cannot fully capture the techniques developed by human interpreters, which make translations more fluent and accurate.

Building on this idea, we generalize such human interpreter techniques into four new actions that can be dynamically invoked by LLMs during generation, enabling real-time application beyond static reference adaptation.

- **Sentence\_Cut** Split long or syntactically complex clauses into shorter, grammatical sentence segments. For example, the sentence in ACL60/60 dataset “*At the position of at the pool party with Barack Obama, we got a graph with the right nodes on the person and the event subject, but guess the wrong timing information*” can be split before “with” since it connects two semantically complete parts. By inserting appropriate punctuation and connective words (which LLMs are able to supply), the sentence is divided into two fluent units, reducing reordering and improving latency.
- **Drop** Remove only truly non-informative content (e.g., “uh”, “you know”), repeated words, or self-corrections. For instance, in “*These are the morphology level, these are the morphology level embeddings,*” the phrase “*these are the morphology level*” is repeated without adding new meanings. Applying **Drop** removes the redundancy and yields a cleaner, semantically accurate translation.
- **Partial\_Summarization** Combine or simplify semantically equivalent or repetitive expressions while preserving the original meaning and tone (e.g., speculation, politeness). This is useful when multiple clauses convey essentially the same information. For example, in “*And here you have the number of spans that were labeled as English and the spans that were labeled as other borrowings and how many of them were unique,*” both clauses share the subject “*the number of spans.*” Summarization condenses the sentence into a more concise form, improving readability and reducing latency without loss of meaning.
- **Pronominalization** Replace repeated or al-

① LLM-based translation generation	② Per-action statistics aggregation	③ Prompt construction
<p>Example</p> <p><b>Input:</b> source sentence <i>This is uh a simple example.</i></p> <p><b>Translation machine:</b> GPT-4o.</p> <p><b>Prompt:</b> Allowed actions (e.g., Read, Write, Drop), language pairs (e.g., En-Zh).</p> <p><b>Output:</b> action sequence and hypothesis</p> <p>(1) Action sequence: Read, Read, Read, Drop, Write, Read, Read, Read, Write.</p> <p>(2) Hypothesis: 这是一个简单的例子.</p>	<p>Example (Statistics of the Drop action)</p> <p><b>Input:</b> source and hypothesis translation</p> <p>(1) Source: <i>This is uh a simple example.</i></p> <p>(2) Hypothesis: 这是一个简单的例子.</p> <p><b>Output:</b> BLEU and LAAL for each action</p> <p>(1) BLEU score: compare hypothesis with reference.</p> <p>(2) LAAL: use Whisper and latency-aware TTS to get the timestamps of source and hypothesis sentences. Calculate LAAL by its formula described in Section 3.3.</p>	<p>Example (Allowed actions: Read, Write, Drop, Sentence_Cut)</p> <p><b>Prompt:</b> You are a simultaneous translation (En-Zh) agent. Your task is ... You can choose from the following actions at each step: Read, Write, Drop, Sentence_Cut...Based on dev set evaluation:</p> <p>- Drop → LAAL ≈ 2.581s, BLEU ≈ 58.94</p> <p>- Sentence_Cut → LAAL ≈ 2.300s, BLEU ≈ 60.28</p> <p>Source sentence: The only parts of the inputs are relevant to the desired output.</p>

Figure 2: Examples accompanying the inference pipeline. The figure provides simplified input–output examples for LLM-based action generation, explains how translation quality (BLEU) and latency (LAAL) are obtained, and shows a sample inference prompt that combines allowed actions with action-level statistics. This figure is intended for clarification and does not reflect full inference traces.

ready mentioned noun phrases with pronouns only if referents are unambiguous. In two consecutive sentences “*Lexical borrowing is a type of linguistic borrowing um which is basically reproducing in one language patterns of other languages. There are however some differences between lexical borrowing and code-switching*”, the phrase “*lexical borrowing*” is repeated. Since no ambiguity would be caused if we replace the second phrase into a pronoun like “it”, and the two phrases are close enough to each other, we can apply **Pronominalization** to convert this long phrase into a pronoun for more fluent expression.

These actions are applied at each decision point, and enables the model to adjust its behavior according to available context and the latency constraints. Training references are prepared by prompting GPT-4o to generate translations under different action combinations. Examples in En-Zh and En-De are shown in Appendix A.1, and detailed experiment settings can be found in A.5.

### 3.2 Prompt-based action-guided inference

Figure 1 illustrates the overall inference pipeline of the proposed framework. We formulate simultaneous interpretation as a step-wise decision process, where an LLM incrementally generates translations by explicitly selecting actions from the action space.

The inference procedure consists of three stages. First, we perform per-action statistics aggregation on the ACL60/60 development set. Using GPT-4o, we generate translations by applying each individual action as well as selected action combinations (sub-figure ① in Figure 2). For each action, we compute translation quality (BLEU) and word-level

monotonicity measured by LAAL. These statistics serve as empirical indicators of the quality-latency trade-offs associated with different actions, as indicated in sub-figure ② in Figure 2.

Second, we construct an action-aware inference prompt, as illustrated in sub-figure ③ in Figure 2. The prompt specifies the allowed action set, describes the interpretation task, and incorporates the aggregated BLEU and LAAL statistics for each action. By exposing the model to these statistics, the prompt explicitly informs the LLM of the expected consequences of choosing different actions during generation.

Finally, we perform step-wise action selection at inference time. Given the source word sequence and the constructed prompt, the LLM selects an action at each decision point and generates the corresponding translation segment. This process allows the model to dynamically balance translation quality and latency based on both the input context and the action-level statistics. A formal pseudocode definition of the step-wise action-guided inference procedure is provided in Appendix A.4.

### 3.3 TTS-based LAAL via a Latency-aware TTS Pipeline

We evaluate word-level monotonicity using a time-based variant of Length-Adaptive Average Lagging (LAAL), referred to as *TTS-based LAAL*. Unlike token-level latency metrics, TTS-based LAAL measures target emission behavior on a real-time axis in seconds, and can therefore be interpreted more directly in the context of simultaneous interpretation.

Let  $X = (x_1, \dots, x_{|X|})$  be the source sequence and  $Y = (y_1, \dots, y_{|Y|})$  the generated target sequence. We denote by  $\{t_1, \dots, t_{|X|}\}$  the source-

side word timestamps and by  $\{\tau_1, \dots, \tau_{|Y|}\}$  the target-side emission timestamps. For each target step  $u$ , we define

$$g(u) = |\{j \mid t_j \leq \tau_u\}|,$$

that is, the number of source words that have been fully observed by the time the  $u$ -th target unit is emitted.

Following LAAL (Papi et al., 2022), we use a length-adaptive diagonal and define

$$d(u) = \min\left(|X|, \frac{g(u)}{u}(u-1)\right).$$

Let

$$\tau^* = \min\{u \mid g(u) = |X|\},$$

or  $\tau^* = |Y|$  if the source is never fully covered before generation ends. We then project both the policy index  $g(u)$  and the adaptive diagonal index  $d(u)$  onto the source time axis, and compute TTS-based LAAL as

$$\text{LAAL}_{\text{sec}} = \frac{1}{\tau^*} \sum_{u=1}^{\tau^*} (\text{time}(g(u)) - \text{time}(d(u))).$$

In speech-based simultaneous translation, both source and target timestamps can in principle be obtained directly from audio. In our text-only setting, however, target-side speech timing is unavailable. To address this, we introduce a latency-aware TTS pipeline that maps target text into causally consistent synthesized speech, from which target emission timestamps can be recovered. Concretely, we obtain source-word timestamps with Whisper (Radford et al., 2022), construct causal source-target alignments, insert <WAIT> tokens when needed to prevent target words from appearing before their aligned source words, and then synthesize the target speech according to the resulting segment timetable. This allows us to compute TTS-based LAAL in seconds while preserving the simultaneity constraints of the original decoding process. Full implementation details of the latency-aware TTS pipeline and the complete computation procedure are provided in Appendix A.10.

## 4 Experiment and analysis

### 4.1 Data

All experiments are conducted in a text-only simultaneous translation setting, where the input is an English word-level transcript and the output is incrementally generated target-side text.

We use two benchmarks with clearly separated roles: MUST-C (Di Gangi et al., 2019) and ACL60/60 (Salesky et al., 2023). The MUST-C corpus is used for supervised encoder-decoder experiments under different reference styles. For En-Zh and En-De, we sample 25k instances from the MUST-C training set and construct three target-side variants with identical sources: the original references, salami-based references, and action-adapted references produced with our prompting strategies. Model validation is performed on the MUST-C tst-HE set and ACL60/60 evaluation (eval) set.

All LLM-based experiments are conducted on the ACL60/60 benchmark. The ACL60/60 development (dev) set is used to collect per-action translation quality and word-level monotonicity statistics and to select demonstration examples for prompting-based methods. The ACL60/60 evaluation (eval) set is used for all final evaluations of LLM-based approaches. Inference experiments cover three language pairs: En-Zh, En-De, and En-Ja.

### 4.2 Backbone models

We consider two types of backbone models in our experiments. For supervised encoder-decoder training, we use mBART50 (facebook/mbart-large-50-many-to-many-mmt) (Liu et al., 2020). For all LLM-based methods, including TransLLaMA-style fine-tuning, few-shot prompting, and step-wise action-based inference, we use Qwen3-8B (Yang et al., 2025) as the base decoder-only language model. In addition, GPT-4o (OpenAI, 2024) is used during development to generate action-based translations for collecting per-action quality and monotonicity statistics, but it is not used for final evaluation.

### 4.3 Evaluation metrics

Translation quality was measured by BLEU, chrF, TER, and neural-based metrics COMET-da and COMET-KIWI. Latency was evaluated using LAAL (Length-Adaptive Average Lagging).

### 4.4 Baseline systems

To evaluate the effectiveness of the proposed action-based framework, we compare against several representative baselines that cover supervised encoder-decoder training, LLM-based supervised adaptation, and prompting-based inference. All baseline systems are evaluated under the same text-only simultaneous translation setting and using identical

evaluation metrics. Detailed experimental settings for all baselines are provided in Appendix A.6.

#### 4.4.1 mBART50 fine-tuning

As a supervised encoder-decoder baseline, we fine-tune mBART50 on parallel data derived from the MUST-C training set. Separate models are fine-tuned under three reference styles: (1) the original MUST-C reference translations, (2) salami-based translations generated using salami prompting, and (3) action-based translations generated with our full action prompt. All mBART50 models share identical source inputs and training configurations, allowing us to isolate the effect of reference adaptation. These models are evaluated on the MUST-C tst-HE set and the ACL60/60 evaluation set.

#### 4.4.2 TransLLaMA

We include TransLLaMA as a strong LLM-based baseline that adapts decoder-only language models to simultaneous translation through supervised fine-tuning. Following Koshkin et al. (2024), training data are causally aligned by inserting <WAIT> tokens into the target sequences, enabling the model to implicitly learn when to generate or wait without relying on an explicit action-selection mechanism. TransLLaMA is fine-tuned on the ACL60/60 development set under three reference styles: (1) ACL60/60 dev set reference translations, (2) salami-based translations, and (3) action-based translations. Results are evaluated on the ACL60/60 evaluation set.

#### 4.4.3 Few-shot prompting

Few-shot prompting is used as a parameter-free baseline that leverages the in-context learning capability of large language models. We construct prompts by providing three demonstration examples selected from the ACL60/60 development set, covering reference, salami-based, and action-adapted translations. These demonstrations guide the model toward different simultaneous translation behaviors without updating model parameters. Few-shot prompting is performed using the same backbone model as other LLM-based methods and evaluated on the ACL60/60 evaluation set. Detailed prompt construction is shown in Appendix A.7.

### 4.5 Main results

#### 4.5.1 mBART50 fine-tuning

Table 1 reports the performance of mBART50 under different supervision styles on the English-

Chinese (En-Zh) and English-German (En-De) language pairs, evaluated on both MUST-C tst-HE and the ACL60/60 eval set. Training on action-based targets consistently outperforms the original MUST-C references and salami-based translations across most metrics, indicating that reference adaptation is beneficial for supervised encoder-decoder models in simultaneous translation settings. In particular, action-based supervision yields the strongest overall performance on both benchmarks, achieving higher BLEU and COMET scores while also reducing latency as measured by LAAL. This suggests that action-style targets encourage more monotonic and semantically faithful generation without sacrificing fluency.

#### 4.5.2 LLM-based model experiments

We report inference results on the ACL60/60 eval set for three language pairs (En-Zh, En-De, and En-Ja), comparing TransLLaMA-style supervised fine-tuning, few-shot prompting, salami technique, and step-wise action-based inference under different reference settings in Table 2. The results show that the choice of supervision or prompting style (reference, salami-based, or action-based) has a consistent impact across methods. For both TransLLaMA and few-shot prompting, action-based references generally lead to higher semantic scores and lower latency compared to reference and salami-based settings, indicating that action-aware supervision better aligns model behavior with the requirements of simultaneous translation.

When comparing different inference paradigms, step-wise action-based inference consistently achieves a better overall quality-latency trade-off than TransLLaMA and few-shot prompting. In particular, the step-wise variants obtain higher BLEU and COMET scores while maintaining lower or comparable latency across all three language pairs. This suggests that explicitly selecting actions during inference provides finer-grained control over incremental generation than relying solely on supervised fine-tuning or static prompting. We further verify that the LLM adapts its action choices to the provided per-action statistics; see Appendix A.12. While TransLLaMA remains competitive in latency-oriented configurations and few-shot prompting achieves strong semantic performance in some cases, both methods are generally outperformed by step-wise inference when quality and latency are considered jointly.

Across three language pairs, inference guided

Table 1: mBART50 supervised adaptation under different training data and evaluation benchmarks. *Reference* denotes the original offline MUST-C translations; *salami-based* and *action-based* refer to targets adapted for simultaneous translation using salami segmentation and our proposed action-based prompting, respectively. LAAL (s) denotes TTS-based LAAL measured on a real-time axis using the latency-aware TTS pipeline, while LAAL (tok) is the conventional token-level LAAL evaluated by SimulEval (Ma et al., 2020).

(a) English-Chinese (En-Zh): Evaluation on MUST-C tst-HE							
Training data	BLEU	chrF	TER↓	COMET-da	COMET-KIWI	LAAL (s)↓	LAAL (tok)↓
MUST-C (reference)	21.29	14.34	101.80	0.7525	0.7292	2.650	4.572
MUST-C + salami-based	23.03	15.35	109.48	0.7903	0.7653	2.656	4.573
MUST-C + action-based	<b>24.34</b>	<b>16.39</b>	<b>100.17</b>	<b>0.8002</b>	<b>0.7900</b>	<b>2.438</b>	<b>4.544</b>

(b) English-Chinese (En-Zh): Evaluation on ACL60/60 eval							
Training data	BLEU	chrF	TER↓	COMET-da	COMET-KIWI	LAAL (s)↓	LAAL (tok)↓
MUST-C (reference)	40.57	26.29	137.77	0.8092	0.7377	2.587	4.523
MUST-C + salami-based	38.80	26.84	169.21	0.8260	0.7564	2.568	4.501
MUST-C + action-based	<b>45.07</b>	<b>29.97</b>	<b>113.54</b>	<b>0.8436</b>	<b>0.7801</b>	<b>2.533</b>	<b>4.443</b>

(c) English-German (En-De): Evaluation on MUST-C tst-HE							
Training data	BLEU	chrF	TER↓	COMET-da	COMET-KIWI	LAAL (s)↓	LAAL (tok)↓
MUST-C (reference)	24.90	51.17	65.45	0.7834	0.7879	1.251	4.061
MUST-C + salami-based	25.47	51.72	66.12	0.7865	0.7943	1.223	3.681
MUST-C + action-based	<b>25.69</b>	<b>51.84</b>	<b>65.20</b>	<b>0.7887</b>	<b>0.7944</b>	<b>1.215</b>	<b>3.648</b>

(d) English-German (En-De): Evaluation on ACL60/60 eval							
Training data	BLEU	chrF	TER↓	COMET-da	COMET-KIWI	LAAL (s)↓	LAAL (tok)↓
MUST-C (reference)	30.57	56.90	54.70	<b>0.7890</b>	0.7646	1.210	4.006
MUST-C + salami-based	35.58	60.21	50.81	0.7884	0.7669	1.191	3.681
MUST-C + action-based	<b>36.02</b>	<b>60.74</b>	<b>50.09</b>	<b>0.7890</b>	<b>0.7687</b>	<b>1.189</b>	<b>3.605</b>

by action choices consistently outperforms salami segmentation, TransLLaMA fine-tuning and few-shot prompting in most quality metrics, and even yields COMET-KIWI scores comparable to the reference translations. Notably, the action combination “**Drop + Sentence\_Cut**”, which achieved the lowest latency on the dev sets, also leads to the best trade-off on the eval sets. For En-Zh and En-Ja, it delivers the highest BLEU and COMET-KIWI while substantially reducing LAAL. For En-De, it improves all quality scores simultaneously while maintaining competitive latency.

These results suggest a shared trend across three language pairs: action-aware inference is better at balancing semantic fidelity and latency than other inference methods. The consistent advantage of “**Drop + Sentence\_Cut**” can be explained by their complementary effects: **Drop** removes redundant or filler material, reducing delay, while **Sentence\_Cut** alleviates long-distance reordering by segmenting complex clauses. Together, they enable translations that are both more fluent and faster, aligning with human interpreter strategies.

For example, given the source sentence “*In*

*other words, it cannot be used for many projects on GitHub*”, the translation obtained with salami segmentation was “换句话说,它不能被使用在许多项目中在GitHub上” (BLEU 49.25), where “GitHub上” was placed at the end, resulting in an expression that does not conform to natural Chinese usage and providing little benefit in latency reduction. In contrast, inference with **Sentence\_Cut+Drop** generated “换句话说,它不能用于GitHub上的许多项目” (BLEU 100.00), which is both fluent and faithful.

#### 4.6 Analysis of TTS-based LAAL as a Proxy for Word Monotonicity

To assess whether TTS-based LAAL provides a stable and interpretable signal of word-level monotonicity, we analyze its system-level behavior and compare the induced rankings with those from existing monotonicity and latency proxies.

We compare system rankings produced by TTS-based LAAL, token-level LAAL, and alignment-based Spearman correlation, which has been used as a proxy for word monotonicity in prior work (Makinae et al., 2024). Table 3 summarizes the

Table 2: Overview of evaluated systems and inference results on the ACL60/60 eval set. (a) Definition of systems evaluated in this work; each system differs in model backbone, tuning strategy, supervision data, and inference-time prompting. (b) English-Chinese (En-Zh), (c) English-German (En-De), and (d) English-Japanese (En-Ja) inference results, reported for systems ①-⑩ defined in (a).

(a) System definition				
ID	System	Tuning method	Tuning data	Inference prompt
①	TransLLaMA (Salami)	Fine-tuning	Dev set (Salami refs)	–
②	TransLLaMA (Reference)	Fine-tuning	Dev set (Reference)	–
③	TransLLaMA (Action)	Fine-tuning	Dev set (Action-adapted refs)	–
④	Few-shot (Salami)	Prompt tuning	Dev set (Salami refs)	Few-shot demonstrations
⑤	Few-shot (Reference)	Prompt tuning	Dev set (Reference)	Few-shot demonstrations
⑥	Few-shot (Action)	Prompt tuning	Dev set (Action-adapted refs)	Few-shot demonstrations
⑦	Salami	–	–	Salami prompt
⑧	Step-by-step (Actions)	–	–	Incremental action selection
⑨	Step-by-step (Drop / Cut)	–	–	Incremental <b>Drop / Cut</b> selection
⑩	Reference	–	–	–

(b) English-Chinese (En-Zh)							
	ID	BLEU	chrF	TER↓	COMET-da	COMET-KIWI	LAAL (s)↓
<b>TransLLaMA</b>	①	57.66	41.36	97.16	0.8768	0.7950	2.376
	②	57.66	41.27	97.16	0.8852	0.8000	2.934
	③	58.50	41.61	<b>96.72</b>	0.8826	0.8053	2.360
<b>Few-shot</b>	④	55.49	50.11	106.11	0.8779	0.7984	2.456
	⑤	55.79	39.11	110.77	0.8856	0.8040	2.945
	⑥	55.80	<b>50.31</b>	105.90	0.8843	0.8079	2.395
⑦	Salami	57.21	40.57	110.04	0.8705	0.7846	2.521
⑧	Step-wise (Actions)	62.44	46.80	126.20	<b>0.9002</b>	0.8020	2.281
⑨	Step-wise (Drop / Cut)	<b>62.84</b>	44.06	104.80	0.8891	<b>0.8080</b>	<b>2.118</b>
⑩	Reference	100.00	100.00	0.00	0.9582	0.8029	2.848

(c) English-German (En-De)							
	ID	BLEU	chrF	TER↓	COMET-da	COMET-KIWI	LAAL (s)↓
<b>TransLLaMA</b>	①	39.12	59.67	46.68	0.8125	0.7798	1.187
	②	39.52	59.61	47.09	0.8198	0.7839	1.193
	③	40.31	60.43	45.87	0.8235	0.7842	<b>1.173</b>
<b>Few-shot</b>	④	38.62	62.74	47.83	0.8245	0.7859	1.192
	⑤	38.28	62.33	48.00	0.8301	0.7876	1.196
	⑥	38.92	63.17	47.85	0.8241	0.7856	1.180
⑦	Salami	47.48	69.87	38.94	0.8534	0.8102	1.182
⑧	Step-wise (Actions)	47.80	70.08	38.72	0.8541	0.8108	<b>1.173</b>
⑨	Step-wise (Drop / Cut)	<b>49.97</b>	<b>70.96</b>	<b>37.38</b>	<b>0.8594</b>	<b>0.8111</b>	1.180
⑩	Reference	100.00	100.00	0.00	0.9511	0.8048	2.507

(d) English-Japanese (En-Ja)							
	ID	BLEU	chrF	TER↓	COMET-da	COMET-KIWI	LAAL (s)↓
<b>TransLLaMA</b>	①	53.12	30.11	98.35	0.8735	0.8313	2.938
	②	54.88	30.48	98.58	0.8789	0.8429	3.085
	③	54.08	30.69	<b>97.72</b>	0.8837	0.8351	2.927
<b>Few-shot</b>	④	52.27	34.93	108.25	0.8926	0.8379	3.049
	⑤	52.91	35.02	104.72	0.8934	0.8423	3.146
	⑥	52.78	<b>35.08</b>	107.31	0.8913	0.8428	2.915
⑦	Salami	50.16	31.98	106.37	0.8854	0.8295	2.955
⑧	Step-wise (Actions)	55.31	34.33	101.42	<b>0.8998</b>	0.8439	<b>2.524</b>
⑨	Step-wise (Drop / Cut)	<b>55.33</b>	34.49	99.06	0.8913	<b>0.8447</b>	2.613
⑩	Reference	100.00	100.00	0.00	0.9721	0.8447	3.298

Table 3: System-level ranking induced by different word monotonicity proxies, computed from mBART50 inference results. Lower is better for LAAL-based metrics, while higher is better for Spearman correlation. The table reports relative system ordering rather than absolute scores.

Metric	En-Zh	En-De
TTS-based LAAL (s)	Action < Salami < MUST-C	Action < Salami < MUST-C
Token-level LAAL	Action < Salami < MUST-C	Action < Salami < MUST-C
Spearman correlation	Salami < Action < MUST-C	Salami < Action < MUST-C

results across language pairs.

Across both En-Zh and En-De, TTS-based LAAL yields identical system rankings to token-level LAAL, indicating that projecting decoding-based latency onto a real-time axis preserves relative monotonicity while expressing delay in seconds. In contrast, Spearman correlation sometimes produces different rankings, suggesting that it captures complementary word-order effects beyond emission timing. Additional analysis relating TTS-based LAAL to end-to-end delay metrics is provided in Appendix A.10.

## 5 Conclusions

We presented a LLM-based framework for simultaneous machine interpretation that extends the traditional READ/WRITE paradigm with a set of interpreter-inspired actions. By enabling new actions **Sentence\_Cut**, **Partial\_Summarization**, **Drop**, and **Pronominalization**, the framework allows large language models to generate translations that better reflect human interpreting strategies under simultaneous constraints. We also introduced a latency-aware TTS pipeline that supports time-based evaluation of word-level monotonicity using TTS-based LAAL.

Experiments across supervised encoder-decoder training and LLM-based inference show that action-aware supervision and inference consistently lead to better trade-offs between translation quality and monotonicity than reference-based or salami-style approaches, with step-wise action selection yielding the strongest overall performance. More broadly, our findings suggest that simultaneous machine interpretation can be formulated as action-driven, human-like decision making at decoding time, offering a flexible alternative to policy-based incremental translation.

## 6 Limitations and Future Work

This work studies simultaneous machine interpretation from a text-to-text perspective, with a focus on action-based generation strategies. While speech

input is not explicitly modeled, the proposed framework is intended to operate as the decoding component of a full SiMT system, where speech signals can be incorporated through an upstream encoder. Extending the framework to an end-to-end SiMT setting that jointly models speech input and action-based decoding is left for future work.

## Acknowledgements

This paper is supported by Project W2531054 of the National Natural Science Foundation of China, and the Program for Guangdong Introducing Innovative and Entrepreneurial Teams.

## References

- 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](#). *CoRR*, abs/1906.05218.
- Jun-Kun Chen, Renjie Zheng, Atsuhito Kita, Mingbo Ma, and Liang Huang. 2020. [Improving simultaneous translation with pseudo references](#). *CoRR*, abs/2010.11247.
- Colin Cherry, George F. Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. 2018. [Revisiting character-based neural machine translation with capacity and compression](#). *CoRR*, abs/1808.09943.
- Kyunghyun Cho and Masha Esipova. 2016. [Can neural machine translation do simultaneous translation?](#) *CoRR*, abs/1606.02012.
- 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.
- Kosuke Doi, Katsuhito Sudoh, and Satoshi Nakamura. 2021. [Large-scale English-Japanese simultaneous interpretation corpus: Construction and analyses with sentence-aligned data](#). In *Proceedings of the 18th*

- International Conference on Spoken Language Translation (IWSLT 2021)*, pages 226–235, Bangkok, Thailand (online). Association for Computational Linguistics.
- Taisiya Glushkova, Chrysoula Zerva, and André F. T. Martins. 2023. BLEU meets COMET: Combining lexical and neural metrics towards robust machine translation evaluation. In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*, pages 47–58, Tampere, Finland. European Association for Machine Translation.
- Shoutao Guo, Shaolei Zhang, Zhengrui Ma, Min Zhang, and Yang Feng. 2024. Sillm: Large language models for simultaneous machine translation. *arXiv preprint arXiv:2402.13036*.
- Yasumasa Kano, Katsuhito Sudoh, and Satoshi Nakamura. 2023. [Average token delay: A latency metric for simultaneous translation](#). *Preprint*, arXiv:2211.13173.
- Roman Koshkin, Katsuhito Sudoh, and Satoshi Nakamura. 2024. [TransLLaMa: LLM-based simultaneous translation system](#). *Preprint*, arXiv:2402.04636.
- Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. [Multilingual denoising pre-training for neural machine translation](#). *Transactions of the Association for Computational Linguistics*, 8:726–742.
- 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. 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.
- Mana Makinae, Yusuke Sakai, Hidetaka Kamigaito, and Taro Watanabe. 2024. [Simul-MuST-C: Simultaneous multilingual speech translation corpus using large language model](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22185–22205, Miami, Florida, USA. 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.
- Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. [Syntax-based simultaneous translation through prediction of unseen syntactic constituents](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 198–207, Beijing, China. Association for Computational Linguistics.
- OpenAI. 2024. Hello gpt-4o. <https://openai.com/index/hello-gpt-4o/>. Accessed: 2026-04-15.
- Sara Papi, Marco Gaido, Matteo Negri, and Marco Turchi. 2022. Over-generation cannot be rewarded: Length-adaptive average lagging for simultaneous speech translation. *arXiv preprint arXiv:2206.05807*.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. [BLEU: A method for automatic evaluation of machine translation](#). In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, page 311, Philadelphia, Pennsylvania. Association for Computational Linguistics.
- Maja Popović. 2015. [chrF: Character n-gram F-score for automatic MT evaluation](#). In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.
- Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. [Robust speech recognition via large-scale weak supervision](#). *Preprint*, arXiv:2212.04356.
- Ricardo Rei, Craig Stewart, Ana C. Farinha, and Alon Lavie. 2020. [COMET: A neural framework for MT evaluation](#). *CoRR*, abs/2009.09025.
- Ricardo Rei, Marcos Treviso, Nuno M. Guerreiro, Chrysoula Zerva, Ana C Farinha, Christine Maroti, José G. C. de Souza, Taisiya Glushkova, Duarte Alves, Luisa Coheur, Alon Lavie, and André F. T. Martins. 2022. CometKiwi: IST-unbabel 2022 submission for the quality estimation shared task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 634–645, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.
- Masoud Jalili Sabet, Philipp Dufter, François Yvon, and Hinrich Schütze. 2020. Simalign: High quality word alignments without parallel training data using static and contextualized embeddings. *arXiv preprint arXiv:2004.08728*.
- Elizabeth Salesky, Kareem Darwish, Mohamed Al-Badrashiny, Mona Diab, and Jan Niehues. 2023.

Evaluating multilingual speech translation under realistic conditions with resegmentation and terminology. In *Proceedings of the 20th International Conference on Spoken Language Translation (IWSLT 2023)*, pages 62–78, Toronto, Canada (in-person and online). Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. 2006. *A study of translation edit rate with targeted human annotation*. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, Cambridge, Massachusetts, USA. Association for Machine Translation in the Americas.

Minghan Wang, Thuy-Trang Vu, Jinming Zhao, Fatemeh Shiri, Ehsan Shareghi, and Gholamreza Haffari. 2024. *Simultaneous machine translation with large language models*. In *Proceedings of the 22nd Annual Workshop of the Australasian Language Technology Association*, pages 89–103, Canberra, Australia. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. *Qwen3 technical report*. *Preprint*, arXiv:2505.09388.

Jinming Zhao, Philip Arthur, Gholamreza Haffari, Trevor Cohn, and Ehsan Shareghi. 2021. *It is not as good as you think! evaluating simultaneous machine translation on interpretation data*. *CoRR*, abs/2110.05213.

## A Appendix

### A.1 New action examples

We provide concrete English-German (En-De) and English-Chinese (En-Zh) examples to demonstrate the functions of each new action. The source sentences and reference translations are selected from the ACL60/60 development set, and the action-based translations are generated using GPT-4o.

#### A.1.1 Sentence\_Cut

Source sentence: At the pool party with Barack Obama, we got a *graph* with the right nodes on the person and the event subject, but guessed the timing information wrong.

- **De.** Reference: Bei der Poolparty mit Barack Obama bekamen wir einen mit den richtigen Knoten zur Person und zum Ereignisthema versehenen *Graphen*, schätzten die zeitlichen Informationen aber falsch ein.

Sentence\_Cut: Bei der Poolparty mit Barack Obama bekamen wir *einen Graphen*. Er hatte die richtigen Knoten zur Person und zum Ereignisthema, aber wir schätzten die zeitlichen Informationen falsch ein.

- **Zh.** Reference: 在与Barack Obama的泳池派对上，我们得到了一张关于人物和事件主题正确节点的图表，但猜错了时间信息。

Sentence\_Cut: 在与Barack Obama的泳池派对上，我们得到了一个图，标示了人物和事件主题正确的节点，但猜错了时序信息。

#### A.1.2 Drop

Source sentence: These are the morphology level, *these are the morphology level* embeddings.

- **De.** Reference: Das hier ist die morphologische Ebene, *das hier sind die* Einbettungen auf der *morphologischen Ebene*.

Drop: Das hier sind die Einbettungen auf der morphologischen Ebene.

- **Zh.** Reference: 这些是形态学层面的，这些是形态学层面的嵌入。

Drop: 这些是形态学层面的嵌入。

#### A.1.3 Partial\_Summarization

Source sentence: And here you have the number of *spans that were labeled as English* and *the spans that were labeled as other borrowings* and how many of them were unique.

- **De.** Reference: Und hier sehen Sie die Anzahl der Segmente, die als Englisch gekennzeichnet wurden, und der Segmente, die als andere Entlehnungen gekennzeichnet wurden, sowie wie viele von ihnen einzigartig waren.

Partial\_Summarization: Und hier sehen Sie, wie viele Segmente als Englisch oder als andere Entlehnung gekennzeichnet wurden und wie viele davon einzigartig sind.

- **Zh.** Reference: 这包含了被标记为英语的跨度数量和标记为其他借词的跨度数量，以及它们中有多少是独一无二的。

Partial\_Summarization: 这包含了被标记为英语和其他借用词的跨度，以及其中有多少是唯一的。

### A.1.4 Pronominalization

Source sentence: Lexical borrowing is a type of linguistic borrowing um which is basically reproducing in one language patterns of other languages. There are however some differences between *lexical borrowing* and code-switching.

- **De.** Reference: Lexikalische Entlehnung ist eine Form der sprachlichen Entlehnung, die im Grunde darin besteht, in einer Sprache Muster anderer Sprachen zu reproduzieren. Es gibt jedoch einige Unterschiede zwischen *lexikalischer Entlehnung* und Code-Switching.

Pronominalization: Lexikalische Entlehnung ist eine Form der sprachlichen Entlehnung, die im Grunde darin besteht, in einer Sprache Muster anderer Sprachen zu reproduzieren. Es gibt jedoch einige Unterschiede zwischen *ihr* und dem Code-Switching.

- **Zh.** Reference: 词汇借用是一种语言借用，它基本上是在一种语言中再现其他语言的模式。不过，词汇借用和语码转换之间还是有一些区别。

Pronominalization: 词汇借用是一种语言借用，本质上是在一种语言中再现其他语言的模式。然而，它和语码转换之间存在一些差异。

### A.2 Prompt of actions

The following is an example of inference prompt letting the LLM choose from all the actions.

You are a simultaneous translation (En-Zh) agent. Your task is to read a source sentence word by word, and decide what action to take at each step to optimize the balance between translation quality and latency. Keep to the original meaning and word order of the sentence when doing translation You can choose from the following actions:

- READ: Wait for the next source word (default).
- WRITE: Output a target word or phrase.
- Drop: Remove previously read word(s) if they are meaningless fillers (e.g., “uh”, “um”), repetitions, false starts, or self-corrections. Use only when clearly justified.
- Partial\_Summarization: Merge or simplify redundant or equivalent expressions, while preserving the meaning and tone

(e.g., politeness, speculation). - Cut: Intentionally split the sentence into two shorter, independently translatable units. Use only when the sentence is long or syntactically complex. - PRONOUN: Replace a repeated noun phrase with a pronoun ONLY IF the referent is unambiguous. Keep to the original word order and meaning, and do the new actions only if it considerably improve the latency or quality of interpretation. Based on dev set evaluation: - Drop → AL ≈ 0.851s, BLEU ≈ 58.94 - Partial\_Summarization → AL ≈ 0.847s, BLEU ≈ 60.33 - Cut → AL ≈ 0.824s, BLEU ≈ 60.28 - PRONOUN → AL ≈ 0.858s, BLEU ≈ 60.85 Only use Drop, Partial\_Summarization, or Cut if they reduce latency without hurting translation quality. - You will receive the full source sentence. Your job is: 1. Simulate the step-by-step translation process internally; 2. Carefully choose the action to take at each step **\*\*strictly based on the statistics provided above\*\***; 3. Output: action sequence of every step, explanation of choosing each action, and the full translation of the sentence. 3. You are given only the prefix of the source. DO NOT use any information beyond the current prefix. If you find yourself relying on unseen future words, output the token <VIOLATION> and stop. Source sentence:<input sentence>

To verify that our setting does not exploit unseen future tokens, we conducted a prefix-feeding sanity check. For a source sentence  $x = (x_1, x_2, \dots, x_n)$ , we iterate  $t = 1$  to  $n$ . At step  $t$ , the model receives only the prefix  $x_{1:t}$  under the same instruction template as our main prompt; it outputs one action from READ, WRITE, DROP, CUT, PARTIAL\_SUMMARIZATION, PRONOMINALIZATION. If an outputting action is chosen (e.g., WRITE or PARTIAL\_SUMMARIZATION), the model must emit an *incremental* target fragment. Crucially, previously emitted target tokens are immutable: later steps may append but never revise earlier output, i.e., the target stream is prefix-monotonic.

**Instruction.** We append the following constraint to the end of the main prompt:

You are a simultaneous translation(En-Zh) agent. Your task

is to read a source sentence word by word, and decide what action to take at each step to optimize the balance between translation quality and latency. Keep to the original meaning and word order of the sentence when doing translation. You can choose from the following actions:

- READ: Wait for the next source word (default).
- WRITE: Output a target word or phrase.
- Drop: Remove previously read word(s) if they are meaningless fillers (e.g., “uh”, “um”), repetitions, false starts, or self-corrections. Use only when clearly justified.
- Partial\_Summarization: Merge or simplify redundant or equivalent expressions, while preserving the meaning and tone (e.g., politeness, speculation).
- Cut: Intentionally split the sentence into two shorter, independently translatable units. Use only when the sentence is long or syntactically complex.
- PRONOUN: Replace a repeated noun phrase with a pronoun ONLY IF the referent is unambiguous. Keep to the original word order and meaning, and do the new actions only if it considerably improve the latency or quality of interpretation. Based on dev set evaluation: - Drop  $\rightarrow$  AL  $\approx$  0.851s, BLEU  $\approx$  58.94 - Partial\_Summarization  $\rightarrow$  AL  $\approx$  0.847s, BLEU  $\approx$  60.33 - Cut  $\rightarrow$  AL  $\approx$  0.824s, BLEU  $\approx$  60.28 - PRONOUN  $\rightarrow$  AL  $\approx$  0.858s, BLEU  $\approx$  60.85 Only use Drop, Partial\_Summarization, or Cut if they reduce latency without hurting translation quality. - You will receive **\*\*a word at one time\*\*** Your job is: 1. Simulate the step-by-step translation process internally; 2. Carefully choose the action to take at each step **\*\*strictly based on the statistics provided above\*\***; 3. Output: At each step, output the action you chose and the incremental translation. If you choose READ or other actions that don't yield a translation, do not output the translation. Just give me the action. When given the complete sentence, output the whole sentence based on previous incremental translations. You are not allowed to modify or overwrite your previous output, only incremental

translations are allowed.

**One-sentence demonstration.** Source: “The method works well for the cases where long inputs are considered.” t=4 (prefix “The method works well”): action=WRITE  $\rightarrow$  “该方法运行良好”; t=6 (“The method works well for the cases”): action=READ (no output); t=8 (“... for the cases where long”): action=Cut  $\rightarrow$  append “, 尤其适用于”; t=11 (“... where long inputs are considered”): action=WRITE  $\rightarrow$  append “长输入的情形。” Final concatenation (end of sentence): “该方法运行良好, 尤其适用于长输入的情形。”

**Finding.** Running this prefix-feeding procedure with the same template and decoding settings produces translations that are nearly identical to those obtained with the single-shot prompt used in our main experiments (differences are limited to minor punctuation or phrasing). We did not observe evidence of future-token leakage: the incremental fragments at step  $t$  remain stable when we randomize the unseen suffix  $x_{1:n}$ , and the final full-sentence outputs match the single-shot results up to negligible surface variation.

### A.3 Adequacy Analysis of Drop and Partial\_Summarization

One concern with interpreter-inspired actions such as **Drop** and **Partial\_Summarization** is the potential loss of semantic adequacy. To examine this risk more directly, we conduct a targeted analysis on outputs produced with these two action types on the ACL60/60 English-Chinese evaluation set. In total, the analysis covers 936 instances in which either **Drop** or **Partial\_Summarization** was applied.

We evaluate adequacy from two perspectives. First, we measure the degree of compression introduced by each action using the hypothesis/reference length ratio. Second, we assess content preservation using reference 3-gram coverage and a set of hand-crafted critical-error checks designed to capture meaning loss that may not be fully reflected by surface-overlap metrics.

The critical-error analysis focuses on two especially sensitive phenomena: number preservation and negation preservation. Concretely, we define the following error types: (i) **Number Missing**, where a number present in the source or reference is absent from the hypothesis; (ii) **Negation Flip**, where the source expresses negation but the hypothesis does not; and (iii) **Total Critical Errors**, a composite indicator that is triggered if any of the following holds: (a) a number present in the

Table 4: Targeted adequacy analysis of **Drop** and **Partial\_Summarization** on ACL60/60 eval (En-Zh). Lower is better for critical-error rates.

Metric	Drop	Partial_Sum
Compr. Ratio (hyp/ref)	0.9704	0.9759
Ref 3-gram Coverage	0.8668	0.8053
Number Missing	0.0%	0.0%
Negation Flip	0.2%	0.2%
Total Critical Errors	2.99%	2.78%

source is missing in the hypothesis, (b) a negation expressed in the source is missing in the hypothesis, (c) a number present in the reference is missing in the hypothesis, or (d) a negation expressed in the reference is missing in the hypothesis.

Negation flips are detected using simple cue matching. We first check whether the English source contains a negation cue (e.g., *not*, *no*, *never*) from a manually defined English negation list, and then check whether the Chinese hypothesis contains a corresponding negation cue (e.g., “不”, “没”, “无法”) from a Chinese negation list. A flip is flagged when negation is present in the source but absent from the hypothesis.

Table 4 summarizes the results. Both **Drop** and **Partial\_Summarization** maintain high compression ratios, indicating that these actions do not aggressively shorten translations. At the same time, critical errors remain below 3% for both actions. In particular, missing numbers are not observed in our sample, and negation flips occur only rarely (0.2% for both actions). These results suggest that, although both actions intentionally simplify or omit parts of the input, they do not substantially compromise adequacy in practice under our prompting setup. We note that these checks are heuristic and target only a subset of adequacy failures, but they provide a direct diagnostic of the types of content loss most likely to arise from omission- or compression-oriented actions.

#### A.4 Formal Definition of Step-wise Action-Guided Inference

To make the decoding procedure in Section 3.2 fully explicit, we provide a formal algorithm for our step-wise action-guided inference. The algorithm follows the same prefix constraint used in our main experiments: at decision step  $i$ , the model only receives the visible source prefix  $x_{1:i}$ , the previously committed target output, and the action statistics included in the prompt. Previously emitted target tokens are immutable: later steps may

append new target material, but never revise past output.

In Algorithm 1, the model never observes future source tokens beyond the current prefix. READ postpones commitment, while WRITE and the new actions append a new target segment to the committed output. **Drop** does not add target content, but removes filler, repetition, or self-correction from the currently visible unresolved buffer. This formulation makes explicit how our method extends the conventional READ/WRITE setting with additional interpreter-inspired actions while preserving incremental decoding.

#### A.5 Batch generation from GPT-4o

We generate action-controlled SiMT outputs under a unified, reproducible pipeline. Inputs are line-delimited English sentences that are trimmed, deduplicated, and split into `.jsonl` shards; each sample is assigned a non-reversible hash as `custom_id` for idempotency and result alignment. Each `.jsonl` line specifies a `/v1/chat/completions` call with an identical prompt template that enforces online-style translation (final translation only, with only specified actions allowed, minimal long-distance reordering), fixed decoding and randomness controls (seed, temperature/top- $p$ , max tokens), and `response_format=json_object` for structured parsing. Shards are submitted as independent batch jobs. Determinism is maintained by fixing seeds, model and dependency versions, the prompt template, and a stable write order after deduplication; outputs are merged and deduplicated by `custom_id` before scoring. All methods share the exact same inputs, prompt, decoding parameters, and post-processing, ensuring that any systematic bias from the measurement pipeline is constant across systems and suitable for reliable *relative* latency and quality comparison.

#### A.6 Experimental setup details

**LLM-based SiMT Training.** We use Qwen3-8B as the base model for decoder-only simultaneous translation. Following the TransLLaMA-style supervised fine-tuning (SFT) paradigm (Koshkin et al., 2024), the training data are causally aligned by inserting explicit `<WAIT>` tokens into the target sequences to enforce monotonic decoding constraints. We apply QLoRA for parameter-efficient fine-tuning, with LoRA rank set to 16, scaling factor  $\alpha = 32$ , and dropout 0.1. The model is trained for 2 epochs using AdamW with a learning rate

of  $5 \times 10^{-5}$  and an effective batch size of 16. All LLM-based experiments are conducted on a single NVIDIA A100 GPU with 40GB memory.

**mBART Training.** For encoder-decoder baselines, we fine-tune mBART-large-50 using the HuggingFace Transformers library. The source and target languages are specified using the corresponding mBART50 language codes. Training data are provided in CSV format and split into training and development sets based on predefined split labels.

Input sequences are truncated to a maximum length of 256 tokens on both the source and target sides. We use a per-device batch size of 8 for both training and evaluation, with optional gradient accumulation to control the effective batch size. Models are trained using AdamW with a learning rate of  $3 \times 10^{-5}$  for up to 10 epochs. A linear warmup schedule with 1,000 warmup steps is applied. Mixed-precision training (FP16) is enabled when supported by the hardware.

**Evaluation and Model Selection.** During training, evaluation is performed every 500 steps on the development set. We report SacreBLEU as the primary validation metric and select the best checkpoint based on development BLEU scores. All generation-based evaluations use greedy decoding with a maximum generation length of 256 tokens. Unless otherwise specified, hyperparameters are fixed across experiments and follow common practice in prior work, and no extensive hyperparameter search is conducted.

### A.7 Prompts of few-shot prompting

The following is the prompt for few-shot prompting. Real examples are manually selected in the datasets (ACL60/60 dev references, salami-based translations, and action-based translations).

```
[System] You are a professional English-to-(tgt lang) simultaneous interpreter. Translate the given English sentence into Chinese. Do not output any thinking steps, only output the final result. Output only JSON: {"source": "...", "translation": "..."}.
```

{FEW\_SHOT\_EXAMPLES} [User] Sentence: {SOURCE\_SENTENCE}

### A.8 Different choices of actions

The results in Table A.5 show that individual actions like **Sentence\_Cut**, **Drop**, and **Partial\_Summarization** each brings moderate im-

provements in fluency or latency, but combining them leads to more significant gains. The BLEU and LAAL data of each individual action are added into step-wise inference prompt, which we discussed in the main content 3.2.

### A.9 Latency-aware TTS pipeline

This appendix describes the latency-aware TTS pipeline used to support the computation of TTS-based LAAL in text-only simultaneous translation settings. The pipeline is not a modeling contribution and is introduced solely for evaluation purposes. Figure A.3 shows the TTS procedure.

1. Apply Whisper large-v2 (Radford et al., 2022) to get timestamps of each English word in the source sentence.
2. Find best word level alignment between source and target sentences with *SimAlign* (Sabet et al., 2020).
3. Insert <WAIT> tokens before the target words if it appears before corresponding source words. In this way, we form causal alignment where the target words are never spoken before the source words.
4. Get segment timetables for target sentences. Specifically, <WAIT> tokens divide the sentences into segments, and the starting time to say each segment is decided by the starting time of the source word corresponding to the first word in this segment (represented by W). Two situations may happen: the source word is spoken *before* or *after* the previous word in target sentence was spoken. In the former case, the succeeding segment can be merged to the previous segment, while in the latter case, the succeeding segment should be spoken when W is spoken.
5. Synthesize speech using the segment timetables and merge them into a whole sentence with Cozyvoice 2 for Chinese and Japanese or Tacotron 2 for German.

### A.10 Computation Details of TTS-based LAAL

#### A.10.1 Detailed computation procedure

The formal definition of TTS-based LAAL is given in Section 3.3. In this appendix, we describe the detailed computation procedure used in our text-only setting, including timestamp extraction, causal alignment, interpolation on the source time axis, and the implementation of the final metric.

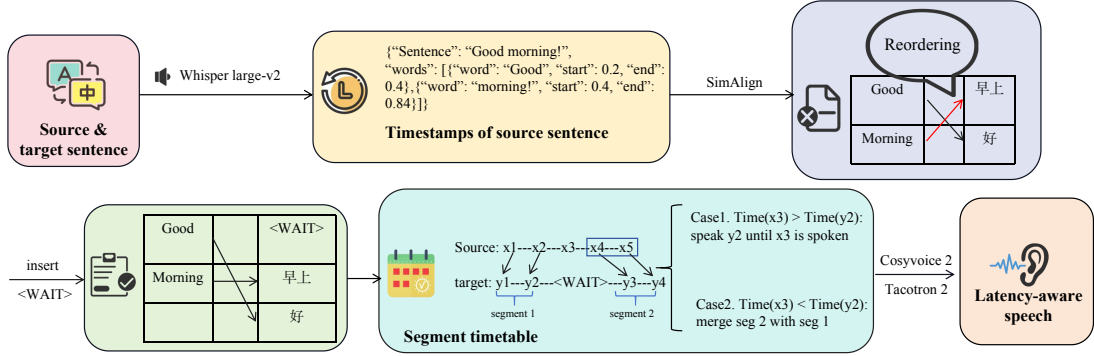


Figure 3: Latency-aware TTS. The system first obtains source word timestamps with Whisper and aligns source–target words using SimAlign. Special `<WAIT>` tokens are inserted to enforce causal alignment, which divides the target into segments. Each segment is then scheduled according to the corresponding source word timing and synthesized with CosyVoice2 (Chinese and Japanese) or Tacotron 2 (German). This process produces speech outputs that reflect realistic latency for evaluation.

Since target-side speech timing is unavailable in text-only simultaneous translation, we first construct a latency-aware target speech signal from the generated text. Source-word timestamps  $\{t_1, t_2, \dots, t_{|X|}\}$  are obtained from Whisper word-level alignment. We then align source and target units using SimAlign and insert `<WAIT>` tokens whenever necessary to ensure causal consistency, i.e., no target unit is scheduled before its aligned source content becomes available. These `<WAIT>` markers divide the target sequence into segments, which are then synthesized according to a segment timetable determined by the aligned source-word timings. The resulting synthesized speech is re-aligned with Whisper to recover target emission timestamps  $\{\tau_1, \tau_2, \dots, \tau_{|Y|}\}$ .

Given the source-word timestamps and recovered target emission timestamps, we compute the time-based LAAL defined in Section 3.3 by projecting both the policy index and the adaptive diagonal onto the source time axis via linear interpolation. Specifically, for a fractional source index  $x$ , we define  $\text{TIMEATINDEX}(x)$  as its interpolated time on the source timeline. Algorithm 2 summarizes the full computation procedure.

### A.10.2 Additional discussion on monotonicity metrics

**Difference from ATD after TTS.** Although TTS-based LAAL is computed using a latency-aware TTS pipeline, it is conceptually different from Average Translation Delay (ATD) measured after speech synthesis. ATD reflects end-to-end system latency and conflates modeling behavior with speech duration and sentence length.

By contrast, TTS-based LAAL retains the length-

adaptive formulation of LAAL and measures word-level monotonicity by comparing the target emission timeline against an ideal adaptive diagonal defined by source word coverage. As a result, TTS-based LAAL should be interpreted as a time-based proxy for monotonicity rather than a measure of overall translation delay.

**Limitations of alignment-based Spearman correlation.** Although alignment-based Spearman correlation has been widely used as a proxy for monotonicity in simultaneous translation, it can be sensitive to surface-level lexical variation that is unrelated to word order or monotonicity.

Table A.7 illustrates several representative examples from the En–Zh setting. In the first two cases, the machine-generated translations differ from the reference only in minor lexical choices (e.g., “信息” vs. “消息”), while preserving identical word order and semantics. Despite these near-equivalent realizations, the resulting Spearman correlation coefficients for the machine outputs are noticeably lower than those of the references. In the third example, the machine translation is arguably closer to the source in meaning and fluency, yet still receives a lower Spearman score. These examples suggest that Spearman correlation may penalize benign lexical substitutions even when alignment structure and monotonicity are effectively unchanged.

This sensitivity arises because Spearman correlation is computed over discrete alignment positions and directly reflects word-level alignment permutations. As a result, it conflates lexical realization differences with monotonicity behavior, making it less robust as a proxy for simultaneous interpretation quality.

### A.11 Step-by-step generation of action sequence

We display examples of outputs of LLM that generate step-by-step action choices and explanation.

**Example 1** See action sequence in Table A.8.

*Source:* We design our experiment as leave one out ah evaluation where we train FeSTe over sixteen datasets and apply it to the seventeenth dataset.

*Translations:* (Ours) 我们将实验设计为留一法评估,其中我们在十六个数据集上训练FeSTe,并将其应用于第十七个数据集。 [BLEU: 61.31]

(Salami) 我们设计了我们的实验,作为留一法评估,在这里我们训练FeSTe在十六个数据集上,并将其应用于第十七个数据集。 [BLEU: 38.58]

**Example 2** See action sequence in Table A.9.

*Source:* Or in other words, we trained the language model to classify an abstract and class ah to abstract and class ah, if the abstract belongs to the class or not.

*Translations:* (Ours) 换句话说,我们训练了语言模型,将摘要和类别分类,无论摘要属于不属于类。 [BLEU: 53.67]

(Salami) 换句话说,我们训练了语言模型来分类一个摘要,并将类别ah分类为摘要和类别ah,判断该摘要是否属于该类别。 [BLEU: 32.16]

### A.12 Adaptive behavior

To examine whether the LLM adapts its action choices in response to the statistics provided in the inference prompt, we conduct a targeted intervention experiment focusing on high-latency cases. Specifically, we first identify the 100 sentences from the evaluation set that yield the largest AL under step-by-step inference with the original prompt. For these sentences, we then modify the inference prompt by artificially altering the reported statistics of a single action, while keeping all other settings unchanged.

In two separate experiments, we promote either **Partial\_Summarization** or **Sentence\_Cut** by increasing its reported BLEU score and decreasing its LAAL value in the prompt. This manipulation is designed to encourage the model to prefer the promoted action by making it appear more favorable in terms of the quality-latency trade-off. As shown in Table A.10, the model indeed invokes the promoted action more frequently during inference. As a result, the generated translations exhibit

a clear reduction in LAAL compared to the baseline, while semantic-based quality metrics degrade only marginally.

These results indicate that the LLM does not merely follow a fixed or static decision template. Instead, it actively conditions its action selection on the provided per-action statistics and dynamically rebalances its strategy to optimize the trade-off between translation quality and latency. This behavior suggests that the model is capable of internalizing external performance signals and using them to guide interpretation decisions at inference time. An illustrative example demonstrating how the model’s action choices change under different statistical configurations is provided below.

*Source sentence:* There has been a growing interest in the influence of English on other languages ah particularly ah related to English lexical borrowings, borrowings which sometimes have been called Anglicisms.

*Baseline:* 人们对英语对其他语言的影响的关注日益增加,尤其是与英语词汇借用有关——这种借用有时被称为“英语化”(Anglicisms)。

*Partial\_Summarization*↑: 人们日益关注英语对其他语言的影响,尤其是与英语词汇借用相关的方面,这些借用有时被称为“英语化”。

In the translation process of this sentence, the second version utilized **PARTIAL\_SUMMARIZATION** more often than the first version. As a result, the segment “There has been a growing interest in the influence of English on other languages” is translated more concisely with less word reordering. This helps improve the latency of this sentence remarkably.

### A.13 LLM usage

This section describes the precise roles of large language models (LLMs) in our work, in accordance with the conference policy. LLMs were used both as components of our SiMT system and as general-purpose assist tools; they are not authors, and the human authors take full responsibility for all content.

**Models and versions.** GPT-4o (gpt-4o-2024-05-13); Qwen3-8B; ChatGPT-5 (for editing).

#### Roles in experiments.

- **GPT-4o:** Generated salami-based and action-adapted translations on the ACL60/60

English→Chinese and English→German dev sets.

- **Qwen3-8B**: Served as the base model for TransLLaMA supervised fine-tuning, few-shot learning, and DICL; also used for inference with both salami-based and action-adapted prompts.

#### **Roles in writing.**

- **ChatGPT-5**: Used strictly for copy-editing (grammar, wording, and minor style/formatting). It did *not* draft sections, introduce claims, or restructure arguments. All technical content, experiments, analyses, and conclusions were written and verified by the authors.

All model outputs (translations and edited text) were reviewed for accuracy; any errors were corrected by the authors. The authors accept full responsibility for the submission, including any content assisted by LLMs. LLMs are not eligible for authorship.

---

**Algorithm 1: Step-wise Action-Guided Incremental Translation**

---

**Input:** Source tokens  $x = (x_1, \dots, x_N)$ ; action set  $\mathcal{A} = \{\text{READ}, \text{WRITE}, \text{DROP}, \text{CUT}, \text{SUM}, \text{PRONOUN}\}$ ; action statistics Stats; language model  $M$ .  
**Output:** Committed target sequence  $y$

```
1  $i \leftarrow 1$ 
2  $y \leftarrow \emptyset$ 
3  $B \leftarrow \emptyset$  // source buffer
4  $H \leftarrow \emptyset$  // action history
5 while  $i \leq N$  or  $B \neq \emptyset$  do
6   if  $i \leq N$  then
7      $B \leftarrow B \oplus x_i$ 
8    $p \leftarrow \text{Prompt}(x_{1:i}, B, y, H, \mathcal{A}, \text{Stats})$ 
9    $(a, z) \leftarrow M(p)$  // selected action and optional output
10  if  $a = \text{READ}$  then
11    if  $i \leq N$  then
12       $i \leftarrow i + 1$ 
13    else
14      break
15  else if  $a = \text{DROP}$  then
16     $B \leftarrow \text{ResolveDrop}(B)$ 
17     $H \leftarrow H \oplus (a)$ 
18  else if  $a = \text{WRITE}$  then
19     $y \leftarrow y \oplus z$ 
20     $B \leftarrow \text{ResolveWrite}(B)$ 
21     $H \leftarrow H \oplus (a, z)$ 
22    if  $i \leq N$  then
23       $i \leftarrow i + 1$ 
24  else if  $a = \text{CUT}$  then
25     $y \leftarrow y \oplus z$ 
26     $B \leftarrow \text{ResolveCut}(B)$ 
27     $H \leftarrow H \oplus (a, z)$ 
28    if  $i \leq N$  then
29       $i \leftarrow i + 1$ 
30  else if  $a = \text{SUM}$  then
31     $y \leftarrow y \oplus z$ 
32     $B \leftarrow \text{ResolveSummarization}(B)$ 
33     $H \leftarrow H \oplus (a, z)$ 
34    if  $i \leq N$  then
35       $i \leftarrow i + 1$ 
36  else if  $a = \text{PRONOUN}$  then
37     $y \leftarrow y \oplus z$ 
38     $B \leftarrow \text{ResolvePronoun}(B)$ 
39     $H \leftarrow H \oplus (a, z)$ 
40    if  $i \leq N$  then
41       $i \leftarrow i + 1$ 
42 return Concatenate( $y$ )
```

---

Table 5: Performance of different action combinations on the dev set. Note that we mark the best results of each metric using **bold** letters and the best results except for the reference translations using **pink highlight**.

En-Zh Translation Performance						
Action Combination	BLEU	chrF	TER↓	COMET-da	COMET-KIWI	LAAL (s)↓
Salami	57.26	38.83	104.73	0.8567	0.7727	2.300
Sentence_Cut	60.28	<b>53.99</b>	101.58	0.8765	0.7927	2.300
Drop	58.94	<b>52.69</b>	101.18	0.8733	0.7909	2.581
Partial_Summarization	60.33	53.67	<b>98.22</b>	0.8764	0.7923	2.553
Pronominalization	60.85	41.39	101.78	0.8738	0.7910	2.567
Sentence_Cut + Drop	60.67	41.88	102.37	0.8745	0.7911	<b>2.275</b>
Drop + Partial_Summarization + Pronominalization	59.91	53.43	<b>98.22</b>	0.8764	0.7924	2.347
All actions	<b>62.67</b>	46.28	99.80	<b>0.8944</b>	<b>0.7952</b>	2.331
ACL60/60 ref	<b>100.00</b>	<b>100.00</b>	<b>0.00</b>	<b>0.9549</b>	<b>0.7983</b>	2.648
En-De Translation Performance						
Action Combination	BLEU	chrF	TER↓	COMET-da	COMET-KIWI	LAAL (s)↓
Salami	<b>47.48</b>	69.86	<b>38.94</b>	0.8534	0.8102	1.116
Sentence_Cut	44.05	<b>69.87</b>	42.66	0.8525	0.8076	1.101
Drop	44.90	68.61	42.63	0.8442	0.7988	1.104
Partial_Summarization	45.05	69.31	41.73	<b>0.8581</b>	0.8086	1.104
Pronominalization	44.96	69.40	41.89	0.8505	0.8074	1.111
Sentence_Cut + Drop	44.74	69.18	41.93	0.8501	0.8068	<b>0.856</b>
Drop + Partial_Summarization + Pronominalization	44.95	69.19	42.08	0.8542	<b>0.8198</b>	1.108
All actions	44.88	69.11	42.05	0.8526	0.8082	1.107
ACL60/60 ref	<b>100.00</b>	<b>100.00</b>	<b>0.00</b>	<b>0.9511</b>	0.8048	1.418
En-Ja Translation Performance						
Action Combination	BLEU	chrF	TER↓	COMET-da	COMET-KIWI	LAAL (s)↓
Salami	49.67	34.81	100.96	0.8659	0.8078	2.924
Sentence_Cut	56.12	37.69	100.96	0.8949	0.8424	2.682
Drop	44.91	<b>68.60</b>	94.81	0.8442	0.8225	2.583
Partial_Summarization	<b>56.82</b>	37.98	100.77	0.8959	<b>0.8428</b>	2.675
Pronominalization	52.86	52.68	<b>77.88</b>	<b>0.9106</b>	0.8375	2.788
Sentence_Cut + Drop	56.17	37.87	100.77	0.8935	0.8421	<b>2.272</b>
Drop + Partial_Summarization + Pronominalization	56.41	37.39	101.92	0.8961	0.8405	2.657
All actions	56.65	37.63	100.58	0.8948	0.8421	2.824
ACL60/60 ref	<b>100.00</b>	<b>100.00</b>	<b>0.00</b>	<b>0.9721</b>	0.8447	3.196

Table 6: Model adaptation under three supervision settings. We compare **TransLLaMA** and **Few-shot (static)** across: (a) *Salami-based* translations, (b) *Action-adapted* translations generated with the full action set, and (c) *ACL60/60* dev-set reference translations. We report surface-overlap metrics (BLEU/chrF/TER), semantic metrics (COMET-da/COMET-KIWI), and latency (LAAL, seconds↓).

(a) Salami-based references						
Method	BLEU	chrF	TER↓	COMET-da	COMET-KIWI	LAAL (s)↓
TransLLaMA	<b>57.66</b>	41.36	<b>96.72</b>	0.8768	0.7950	<b>2.376</b>
Few-shot prompting	55.49	<b>50.11</b>	106.11	<b>0.8779</b>	<b>0.7984</b>	2.456

(b) Action-adapted reference generated with full action set						
Method	BLEU	chrF	TER↓	COMET-da	COMET-KIWI	LAAL (s)↓
TransLLaMA	<b>58.50</b>	41.61	<b>97.16</b>	0.8826	0.8053	<b>2.360</b>
Few-shot prompting	55.80	<b>50.31</b>	105.90	<b>0.8843</b>	<b>0.8080</b>	2.395

(c) ACL60/60 dev-set reference						
Method	BLEU	chrF	TER↓	COMET-da	COMET-KIWI	LAAL (s)↓
TransLLaMA	<b>57.66</b>	<b>41.27</b>	<b>96.72</b>	0.8852	0.8000	<b>2.934</b>
Few-shot prompting	55.79	39.11	110.77	<b>0.8856</b>	<b>0.8079</b>	2.945

---

**Algorithm 2:** Time-based Length-Adaptive Average Lagging (LAAAL<sub>sec</sub>)

---

**Input:** Source word end times $\{t_1, \dots, t_{|X|}\}$  (monotonic); target  
unit onset times  $\{\tau_1, \dots, \tau_{|Y|}\}$ **Output:** LAAAL<sub>sec</sub>

```
1 for  $u \leftarrow 1$  to  $|Y|$  do
2    $g(u) \leftarrow |\{j \mid t_j \leq \tau_u\}|$  // # source
   words finished by  $\tau_u$ 
3  $u^* \leftarrow \min\{u \mid g(u) = |X|\}$ 
4 if no such  $u$  then
5    $u^* \leftarrow |Y|$ 

6 Define TIMEATINDEX( $x; t_1, \dots, t_{|X|}$ ) as:
7   if  $x \leq 1$  then return  $t_1$ ;
8   if  $x \geq |X|$  then return  $t_{|X|}$ ;
9    $i \leftarrow \lfloor x \rfloor$ ;  $w \leftarrow x - i$ ;
10  return  $(1 - w)t_i + wt_{i+1}$ 

11  $s \leftarrow 0$ 
12 for  $u \leftarrow 1$  to  $u^*$  do
13    $x_{\text{pol}} \leftarrow \max(1, \min(|X|, g(u)))$ 
14    $x_{\text{diag}} \leftarrow$ 
      $\max(1, \min(|X|, \frac{g(u)}{u}(u - 1)))$ 
15    $\text{policyTime} \leftarrow$ 
     TIMEATINDEX( $x_{\text{pol}}; t_1, \dots, t_{|X|}$ )
16    $\text{diagTime} \leftarrow$ 
     TIMEATINDEX( $x_{\text{diag}}; t_1, \dots, t_{|X|}$ )
17    $s \leftarrow s + (\text{policyTime} - \text{diagTime})$ 
18 return LAAALsec  $\leftarrow s/u^*$ 
```

---

Table 7: Examples illustrating the sensitivity of alignment-based Spearman correlation to lexical variation. Machine translations differ only minimally from references, yet receive lower Spearman scores.

---

**Example 1** (Machine  $\rho = 0.7157$ , Reference  $\rho = 0.7409$ )

EN: The commit messages classified as other are discarded.

ZH (Machine): 被归类为“其他”的提交信息会被丢弃。

ZH (Reference): 被归类为“其他”的提交消息将被丢弃。

**Example 2** (Machine  $\rho = 0.6492$ , Reference  $\rho = 0.6938$ )

EN: Next is a commit message.

ZH (Machine): 接下来是一个提交信息。

ZH (Reference): 接下来是提交消息。

**Example 3** (Machine  $\rho = 0.7258$ , Reference  $\rho = 0.7447$ )

EN: Therefore, as a third measure for generating valid foils, we employ human annotators to validate the data used in VALSE.

ZH (Machine): 因此, 作为生成有效干扰项的第三项措施, 我们聘请了人工标注员来验证VALSE中使用的数据。

ZH (Reference): 所以, 作为生成有效干扰词的第三项措施, 我们使用人类注释者来验证VALSE中使用的数据。

Step	Source word	Action	Output (if have)	Explanation
1	We	READ	—	Wait for the predicate; subject alone is incomplete.
2	design	READ	—	Likely a “design ... as” frame; need object and complements.
3	our	READ	—	Part of the object NP.
4	experiment	WRITE	我们将实验设计	Core clause “we design our experiment” is complete; emit core without preempting “as”.
5	as	WRITE	为	“design ... as” is now confirmed; add “为”.
6	leave	READ	—	Part of a fixed term.
7	one	READ	—	Part of a fixed term.
8	out	WRITE	留一法	“leave-one-out” fully recognized; safe to emit early.
9	ah	Drop	—	Clear filler; drop to reduce latency.
10	evaluation	WRITE	评估	Completes the term “leave-one-out evaluation”.
11	where	WRITE	,其中	Clause linker; output for fluency.
12	we	READ	—	Wait for predicate and object.
13	train	READ	—	Wait for object and prepositional phrase.
14	FeSTe	READ	—	Needs the following PP before emitting.
15	over	READ	—	Preposition observed; needs its object.
16	sixteen	READ	—	Wait for the head noun.
17	datasets	WRITE	我们在十六个数据集上训练FeSTe	VP complete with PP; output in Chinese word order.
18	and	READ	—	Could coordinate a verb or NP; hold to avoid guessing.
19	apply	WRITE	,并	Seeing “apply” confirms coordinated verb; output “,并”.
20	it	PRONOUN	将其应用	Pronoun refers to FeSTe; emit core predicate “应用” only; defer the “于”.
21	to	WRITE	于	“to” arrives; add “于” to complete “应用于”.
22	the	READ	—	Determiner; no emission needed.
23	seventeenth	READ	—	Wait for the noun to complete the phrase.
24	dataset	WRITE	第十七个数据集。	Target NP complete; finalize the sentence.

Table 8: Simultaneous translation actions with Chinese outputs and English explanations. Textual output of LLM: Here’s my simultaneous translation decision trace (word-by-word), choosing special actions only when they improve latency/quality per the given stats (PRONOUN best BLEU  $\approx$ 60.85; PARTIAL\_SUM  $\approx$ 60.33 with low AL; Drop helps latency with minimal loss)

Step	Source word	Action	Output (if have)	Explanation
1	Or	READ	—	Connector starts a phrase; nothing emit-worthy yet.
2	in	READ	—	Functions as a preposition within a set phrase; unit still incomplete.
3	other	READ	—	Phrase remains incomplete; no stable unit.
4	words,	PARTIAL_SUMMARIZATION	换句话说,	Idiomatic connective is complete with the current token; concise Chinese connective is appropriate.
5	we	READ	—	Subject present; predicate absent.
6	trained	READ	—	Verb present; object absent.
7	the	READ	—	Determiner begins an NP; incomplete.
8	language	READ	—	Inside the NP; still incomplete.
9	model	WRITE	我们训练了语言模型	Head noun closes the NP; the SVO clause is complete and safe to emit.
10	to	READ	—	Infinitive marker introducing purpose; verb pending.
11	classify	READ	—	Infinitive verb identified; requires object(s).
12	an	READ	—	Article; NP not yet formed.
13	abstract	READ	—	First object candidate; object remains open.
14	and	READ	—	Coordinator indicates multiple conjuncts; still open.
15	class	WRITE	将摘要和类别分类	Object closes with the current head; combine with “classify” to emit.
16	ah	Drop	—	Filler; drop.
17	to	READ	—	Preposition starting a new phrase; attachment unresolved.
18	abstract	READ	—	Repetition of earlier content; potential redundancy.
19	and	READ	—	Coordinator within the repeated span; still unresolved.
20	class	Drop	—	Local repetition (“abstract and class”) established; drop as redundant.
21	ah,	Drop	—	Filler; drop.
22	if	READ	—	Introduces a condition; polarity and scope not yet determined, so no emission.
23	the	READ	—	Article; NP not yet formed.
24	abstract	READ	—	Head noun appears; complement still missing.
25	belongs	READ	—	Predicate present; complement pending.
26	to	READ	—	Preposition present; object missing.
27	the	READ	—	Article for the object NP; head not yet present.
28	class	READ	—	Object head present; condition’s polarity still unspecified; hold.
29	or	READ	—	Coordinator signals an alternative; construction not closed.
30	not.	WRITE	,无论摘要属不属于类	Polarity is explicit; realize the condition with the compact “无论...” construction.

Table 9: Step-by-step simultaneous translation actions for the sentence. Textual output: Here’s my step-by-step sim of the simultaneous translation, with actions chosen to balance latency and quality (using Drop for fillers, Partial\_Summarization to merge redundancy).

Table 10: Effect of boosting one action’s BLEU and lowering its LAAL in the inference prompt on the top-100 high-LAAL sentences.

Setting	BLEU	chrF	TER	COMET-da	COMET-KIWI	LAAL (s)
Baseline (default prompt)	<b>63.87</b>	<b>45.94</b>	139.66	<b>0.8886</b>	<b>0.7903</b>	4.120
Partial_Summarization (↑)	53.60	37.38	152.59	0.8745	0.7896	<b>3.269</b>
Sentence_Cut (↑)	49.29	34.52	<b>137.93</b>	0.8598	0.7707	3.322