Simultaneous Machine Translation with Large Language Models

Minghan Wang, Thuy-Trang Vu, Jinming Zhao,
Fatemeh Shiri, Ehsan Shareghi, Gholamreza Haffari
Department of Data Science & AI, Monash University
{minghan.wang,trang.vu1,jinming.zhao,
fatemeh.shiri, ehsan.shareghi, gholamreza.haffari}@monash.edu

Abstract

Real-world simultaneous machine translation (SimulMT) systems face more challenges than just the quality-latency trade-off. They also need to address issues related to robustness with noisy input, processing long contexts, and flexibility for knowledge injection. These challenges demand models with strong language understanding and generation capabilities which may not often equipped by dedicated MT models. In this paper, we investigate the possibility of applying Large Language Models (LLM) to SimulMT tasks by using existing incremental-decoding methods with a newly proposed RALCP algorithm for latency reduction. We conducted experiments using the `Llama2-7b-chat` model on nine different languages from the MUST-C dataset. The results show that LLM outperforms dedicated MT models in terms of BLEU and LAAL metrics. Further analysis indicates that LLM has advantages in terms of tuning efficiency and robustness. However, it is important to note that the computational cost of LLM remains a significant obstacle to its application in SimulMT.[1]

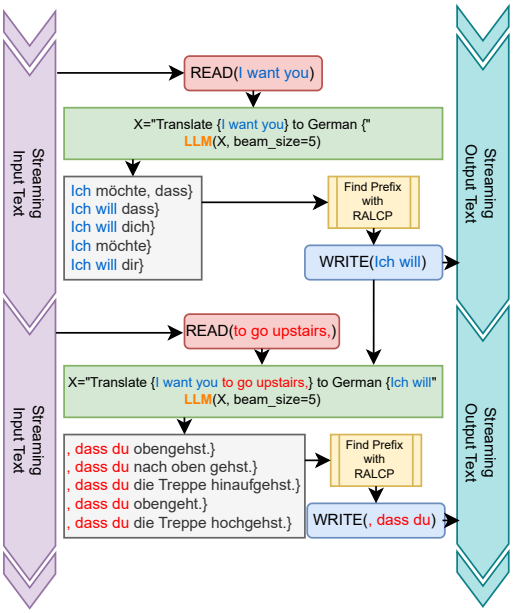

Figure 1: The illustration of the pipeline of our framework where the source texts are read from the streaming input buffer and incrementally added to the prompt. Target texts are written to the streaming output buffer and are also added to the prompt incrementally. RALCP denotes the Relaxed Agreement Longest Common Prefix algorithm proposed by us (§3.3).

1 Introduction

Simultaneous Machine Translation (SimulMT) is a highly challenging task, demanding both high quality and low latency (Gu et al., 2017a), while also confronting various real-world challenges. Since SimulMT systems are typically part of a Simultaneous Speech Translation (SimulST) system cascaded with an Automatic Speech Recognition (ASR) module, these challenges include, but are not limited to: (*i*) ASR outputs often contain errors, necessitating a degree of fault tolerance in the SimulMT model (Ruiz and Federico, 2014; Hu and Li, 2022); (*ii*) SimulMT is typically applied to nearly endless input streams, requiring translation content to maintain good contextual consistency (Radford et al., 2023); (*iii*) System needs to easily incorporate external knowledge for intervention in translation content, such as sensitive word blacklists or specific name translations.

Most existing work primarily focuses on building dedicated SimulMT models and policies to find the optimal balance between quality and latency (Ma et al., 2019a; Chiu and Raffel, 2017; Arivazhagan et al., 2019; Raffel et al., 2017; Gu et al., 2017a; Arthur et al., 2021a; Wang et al., 2022). Some efforts have successfully transformed offline Neural Machine Translation (NMT) models into SimulMT models to avoid the high cost of training from scratch (Liu et al., 2020; Nguyen et al., 2021a; Guo et al., 2023; Arivazhagan et al., 2020;

[1]Repository: https://github.com/yuriak/LLM-SimulMT

Proceedings of The 22nd Annual Workshop of the Australasian Language Technology Association, pages 89–103
December 2-4, 2024

Papi et al., 2022a), but they have not sufficiently explored the challenges mentioned above. Recently, the rapid development of large language models (LLMs) has demonstrated their multitasking and multilingual capabilities, offering new solutions for many complex NLP tasks (OpenAI, 2023; Touvron et al., 2023a,b; Bang et al., 2023). Research indicates that they also have certain advantages in offline translation tasks, specifically for high-resource languages (Hendy et al., 2023; Zhu et al., 2023; Robinson et al., 2023; Yang et al., 2023). Therefore, it is natural to consider whether the powerful understanding and generation capabilities of LLMs can be leveraged to address the challenges in SimulMT.

However, applying LLMs to SimulMT itself presents challenges, such as designing suitable read-write policies for LLMs and effectively handling incremental source and target states, along with their benefits or costs. Therefore, in this paper, we pose two research questions: *(1) whether we could effectively transform off-the-shelf open-source LLMs with light adjustments into SimulMT models?* and *(2) whether LLMs' application in SimulMT address some of the aforementioned challenges, and in doing so, are there any limitations?*

To address these questions, we first select the `Llama2-7b-chat` (Touvron et al., 2023b) as the backbone LLM. Then, considering the expensive training cost of LLM, we choose to find an approach that could endue LLM the ability of simultaneous decoding without training. Thus, we design the "read-n & incremental decoding" policy based on the approach proposed in (Liu et al., 2020; Nguyen et al., 2021a), namely the incremental-decoding with local agreement (LA), which could turn a sequence-to-sequence model that is trained specifically for offline decoding into a model supporting simultaneous decoding. Furthermore, to address the high latency issue caused by the Longest Common Prefix (LCP) algorithm used in the incremental decoding, we propose the Relaxed Agreement Longest Common Prefix (RALCP) algorithm to improve the selection of candidates to write during incremental decoding, resulting in a significant reduction of latency. We then conduct experiments on nine language pairs from the MUST-C (Gangi et al., 2019) dataset, comparing our LLM with dedicated NMT models such as Transformer (Vaswani et al., 2017). Our findings indicate that LLMs can outperform dedicated MT models using exactly the same decoding policy. Finally, we conduct a detailed analysis of different factors affecting the use of LLM for SimulMT, including its potential advantages (e.g. the improvement of data utilization efficiency, the robustness of noisy input) and limitations (e.g. the efficiency issue).

Our contributions can be summarized as follows:

- In this paper, we use the incremental decoding framework to turn an LLM into a simulMT model and propose RALCP to address the high latency issue caused by the LCP algorithm.
- We showcase the potential of applying LLMs to SimulMT tasks and demonstrate that LLMs, after undergoing supervised fine-tuning, can achieve comparable performance to dedicated SimulMT systems.
- Through our analysis, we discover that LLMs' prior knowledge is helpful for improving the efficiency of supervised fine-tuning on certain languages, and for the robustness of noisy input.
- We identify that the computational cost of LLMs during inference is a potential issue limiting their application in SimulMT.

2 Background

Simultaneous Machine Translation (SimulMT) is a task requiring the MT model to return translation content with the incremental source context in a real-time manner. It can be formalized as a Markov Decision Process (MDP), where the model can be considered as a policy function π. It receives the current state $\mathcal{S}_t$ at a specific time step t, and returns an action: $\mathcal{A}_t = \pi(\mathcal{S}_t)$, where $\mathcal{A}_t \in \{\mathbb{R}, \mathbb{W}\}$. Here, $\mathbb{R}$ represents continuing to `READ` the source context, and $\mathbb{W}$ signifies the action to `WRITE` the most recent translation segment. The state $\mathcal{S}_t$ generally encompasses the history of the already read source text and the translated target text $\mathcal{S}_t = \langle S_i^t, T_j^t \rangle$, where i and j are the length of the source and target history. Therefore, we can use $\mathbb{R}(i+1)$ to represent an action of reading one additional source token and use $\mathbb{W}(w, j+1)$ to represent the writing of a token w. The update of state $\mathcal{S}_t$ according to the action $\mathcal{A}_t$ can be denoted as:

$$\mathcal{S}_{t+1} = \begin{cases} \langle S_i^t \cup \{w\}, T_j^t \rangle & \mathcal{A}_t = \mathbb{R}(i+1) \\ \langle S_i^t, T_j^t \cup \{w\} \rangle & \mathcal{A}_t = \mathbb{W}(w, j+1) \end{cases}$$

where w represents any source or target word.

The evaluation of SimulMT systems not only considers translation quality but also accounts for latency, which measures the delay between target and source trajectory. Metrics used to measure latency include Average Lagging (AL) (Ma et al., 2020), Average Proportion (AP) (Cho and Esipova, 2016) or Length-Adaptive Average Lagging (LAAL) (Papi et al., 2022b). In this paper, we adopt LAAL (See Appendix C.1 for definition) because of its better calibration on the length difference between the hypothesis and the reference.

Large Language Model (LLM) leverage autoregressive decoding to conduct unsupervised language modeling on extensive text corpora, which equips them with language understanding and generation capabilities. Most LLMs nowadays are using the decoder-only Transformer architecture (Vaswani et al., 2017) composed of layers of self-attention and feed-forward blocks. In addition to unsupervised training, recent LLMs undergo supervised fine-tuning (SFT) and reinforcement learning from human feedback (RLHF) to align their behavior with human preferences (Ouyang et al., 2022). This allows these models to perform various NLP tasks through conversational interactions. More specifically, users construct prompts that include instructions and context and prompt the model to generate responses containing the desired results. In our paper, we mainly use beam search instead of top-p sampling to acquire more stabilized translations. Thus, we consider the calling of LLMs as a generative function g_θ with the prompt X sequence and the beam size B as input and the response sequences $\mathbf{Y}$ (for all beam candidates) as well as their probabilities $\mathbf{Pr}$ as the return values: $\mathbf{Y}, \mathbf{Pr} = g_\theta(X, B)$.

3 Adapting LLM to SimulMT

3.1 Prompt Design of Incremental States

While there are significant differences in the decoding process between SimulMT models and offline MT models, the fundamental approach to guiding LLMs in translation remains consistent. This approach continues to rely on constructing prompts composed of instructions + context as input, prompting LLMs to perform text completion. To elaborate further, in offline translation, we usually construct a prompt as follows: "**[INST] Translate the following sentence from English to German:** S **[/INST]**", where S is the source sentence. LLMs then provide the translation in the content completed after "[/INST]". The completed translation can be denoted as T.

In SimulMT, we keep the instruction unchanged and consider the source text as a time-dependent variable-length sequence S_i^t indicating at time step t, i source tokens have been read. Additionally, we treat the accumulated translation content as another variable-length sequence T_j^t, indicating j target tokens have been written at time step t. At this point, the model's input is also time-dependent, and we define X_t as the input to the model at time step t. X_t can be obtained through the prompting function $X_t = \text{create\_prompt}(S_i^t, T_j^t)$, which puts S_i^t and T_j^t in the same sequence starting with the instruction: "**[INST] Translate the following sentence from English to German:** S_i^t **[/INST]** T_j^t". By employing this approach, we can effectively manage the ongoing source and target content separately and structure them into standardized prompts (line 6 in Algo 1).

Algorithm 1 Read-n & Incremental Decoding π

Require: LLM : g_θ,
Cumulative Source Content: S_i,
Cumulative Target Content: T_j,
Variables Definition: Read-n: n, Beam-size: B, Agreement-degree: γ, Time step: t { t start from 0}, i and j {source and target length}

1: **if** NOT_FINISHED(S_i^t) **then**
2: **if** $i == 0$ **or** $i \mod n > 0$ **then**
3: **return** $\mathbb{R}(i+1)$
4: **end if**
5: **end if**
6: $X_t \leftarrow$ create_prompt(S_i^t, T_j^t)
7: //LLM only returns new tokens after X_t
8: $\mathbf{C}_t, \mathbf{Pr}_t \leftarrow g_\theta(X_t, B)$
9: //$\mathbf{C}_t$ and $\mathbf{Pr}_t$ are sets of beam candidates and their probabilities.
10: **if** NOT_FINISHED(S_i) **then**
11: $P_t \leftarrow$ RALCP($\mathbf{C}_t, B, \gamma$)
12: **else**
13: $b^* \leftarrow \arg\max_b \mathbf{Pr}_t$
14: $P_t \leftarrow C_t^{b^*}, C_t^{b^*} \in \mathbf{C}_t$
15: **end if**
16: **if** $P_t == \emptyset$ **then**
17: **return** $\mathbb{R}(i+1)$
18: **end if**
19: **return** $\mathbb{W}(P_t, j + |P_t|)$

3.2 Read-n & Incremental-decoding Policy

Given our goal of exploring the practical application of LLMs in SimulMT tasks in a straightforward and effective manner, our policy design adheres to two main principles. Firstly, we aim for the policy to rely primarily on LLMs' inherent text generation capabilities, avoiding the introduction of additional parameters for policy learning. Secondly, recognizing that invoking LLMs typically incurs substantial computational overheads and may result in additional processing delays, we seek to provide users with convenient control over the frequency of LLM invocation.

Building upon these principles, we introduce the **Read-n & incremental-decoding** policy. To determine the timing of taking `READ` action, we employ a straightforward approach: after each `WRITE` action, a fixed number of n tokens are read (line 2 in Algo 1). This method offers a convenient means of controlling the frequency of LLM invocation, as the decision-making process does not require LLM participation. Additionally, this approach aligns with the operational mode of many streaming ASR systems such as U2++ (Wu et al., 2021), which read speech chunks at fixed time intervals and predict multiple transcript tokens to feed into SimulMT system for translation.

For the decision of `WRITE` action, we directly employ the incremental-decoding method proposed in (Liu et al., 2020; Nguyen et al., 2021a). This entails invoking LLM based on the current incremental state to perform a complete beam search decoding (line 8 in Algo 1). Subsequently, we utilize the longest common prefix (LCP) algorithm to identify a prefix (line 11 in Algo 1) with local agreement (LA) in the word level (§3.3). If such a prefix is found, the policy triggers a `WRITE` action; otherwise, it proceeds to read n consecutive tokens (line 17 in Algo 1).

3.3 Latency Reduction with RALCP

Although the incremental-decoding algorithm has endowed LLM with the capability to perform SimulMT, there is a challenge when dealing with beam search candidates exhibiting significant diversity (See Figure 2 for an example). In such cases, the original LCP algorithm may struggle to promptly provide the longest prefix suitable for writing out. Consequently, the LLM invocation associated with the current incremental state goes to waste, resulting in a substantial increase in latency. To address this problem, we optimize the LCP algorithm and introduce the Relaxed Agreement Longest Common Prefix (RALCP) algorithm.

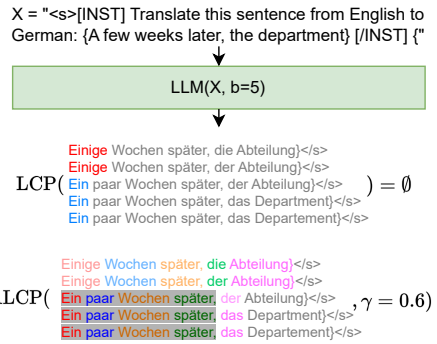

Figure 2: This example shows the scenario where the LCP algorithm fails to find a common prefix because of the difference of the first token, but RALCP successfully returns the prefix because of the relaxed constraints. For RALCP, words at the same position are annotated with the same color group, their votes are indicated by the darkness of the color. The selected prefix is annotated with gray background.

RALCP employs a voting mechanism to relax the constraints on identifying the common prefix. For example, if 80% of the candidates can propose the same token, then that token is accepted as a part of the prefix. We denote γ as the agreement threshold, which is considered as the threshold of accepting the most frequent token at the certain position. Specifically, in conventional LCP, the prefix with local agreement is located by matching the token at the same position i for all candidate sequences, if they are holding the same token, the token will be gathered into the prefix. In RALCP, we relax the criteria of selecting the token by employing the voting mechanism, i.e. if the token at i has the normalized votes (frequency) larger than γ, it will be accepted in the prefix. In our experiments, we explored γ ranging from 0.1 to 1.0 and found that 0.6 is an empirically balanced value toward performance and latency (See C.4 for detail).

3.4 SFT and Prefix Training

Due to the fact that 89.7% of the pretraining corpus of `Llama2` consists of English, we observed a significant limitation in its multilingual translation capabilities during our experiments (§4.2). In the one-shot setting, it still exhibited a considerable performance gap when compared to other baselines. To address this inherent disadvantage caused by the low coverage of non-English languages in its pretraining data, we further explored the use of

supervised fine-tuning (SFT) to explore the extent of achievable improvement.

However, due to the high computational cost associated with fine-tuning on a large dataset with full parameters, which is infeasible and not align with our aforementioned principles in §3.2. We placed restrictions on the SFT method to control the cost. Specifically, we used LoRA (Hu et al., 2022) for efficient fine-tuning, and frozen original LLM parameters. Furthermore, we conducted training for just **one** epoch on the fine-tuning set in the main experiment.

We explored two SFT strategies in total: (*i*) Pure Offline SFT, where we used full sentence source-target pairs to construct prompts and responses for training, and (*ii*) offline + Prefix, where we mixed full sentence source-target pairs with a small number of prefix-to-prefix pairs (introduced shortly) and conducted fine-tuning on this combined dataset.

Pure Offline SFT We mixed all the training data of MUST-C dataset for each selected language pair into a combined dataset. For each sample, to achieve better generalisation, we first sample a template from a list of 10 predefined templates to construct the prompt input as in sec §3.1. The predefined templates are shown in Appendix B. During the fine-tuning, we only compute loss on target response to avoid catastrophic forgetting as suggested in (Touvron et al., 2023b).

Offline + Prefix SFT Inspired by the approach of tuning the model on the prefix-to-prefix data described in (Niehues et al., 2018; Liu et al., 2020), which is aiming at solving the "fantasize" problem (the translation is often fantasized by the model to be a full sentence), we create our prefix-to-prefix dataset. However, instead of creating a 1:1 sized artificial prefix dataset with proportional-based truncating, we choose to use ChatGPT (`gpt-3.5-turbo`) to create a much smaller one for convenience. Specifically, we randomly sampled 1000 source sentences from the training set of each language pair and truncated them into 20% to 80% of the full length uniformly, resulting in 9000 source prefixes. We then used ChatGPT to translate these source prefixes into target prefixes. We checked the quality of the generated prefixes with a quantitative analysis to ensure the quality was reasonable. Further details are provided in Appendix A. These prefix pairs are mixed together with the full sentence dataset used in the pure offline SFT strategy for SFT in the same manner.

Language	Pretraining Coverage %	# SFT sample	# Test sample	Genus	Word Order
Czech	0.03	116.2k	2034	Slavic	SVO
German	0.17	206.9k	2640	Germanic	SOV
Spanish	0.13	240.3k	2501	Romance	SVO
French	0.16	247.9k	2631	Romance	SVO
Italian	0.11	228.3k	2573	Romance	SVO
Dutch	0.12	224.8k	2614	Germanic	SVO
Portuguese	0.09	186.8k	2501	Romance	SVO
Romanian	0.03	212.9k	2555	Romance	SVO
Russian	0.13	257.8k	2512	Slavic	SOV

Table 1: This table presents the statistic of the parallel dataset used in our experiments, including the coverage of each in `Llama2` pretraining corpus, the number of examples for SFT in our experiments, the number of test samples in the MUST-C test set, as well as the Genus of each target language. Note that all of these languages belong to the Indo-European family.

4 Experiments

4.1 Experimental Setup

Data and Evaluation We selected nine language pairs from the MUST-C (Gangi et al., 2019) dataset, which has been commonly used in the evaluation of the performance of speech and text translation systems. These nine language pairs all have English as the source language and consist of TED talk speech utterances. Detailed statistics of each language pair can be found in Table 1. During training, the combined training set has a total number of 2M samples with an additional 9000 prefix-to-prefix samples (§3.4) for the SFT+prefix training. We used the `tst-COMMON` test set for evaluation. For evaluation metrics, BLEU (Papineni et al., 2002) is used for evaluating quality, and LAAL (Papi et al., 2022b) is used for evaluating latency. All evaluations are conducted with the SimulEval toolkit (Ma et al., 2020), which follows the restriction of IWSLT evaluation (Agrawal et al., 2023) that the committed translation segments are not allowed to be updated.

LLM We used `Llama2-7B-chat`[2] as the LLM (Touvron et al., 2023b) in the experiments. It has been pretrained on 2B of tokens, and with a context length of 4K. The reason for choosing the 7B version in the experiment is that the model with this parameter size can perform inference on a single GPU, making it more suitable for real-world use cases.

During SFT, we use LoRA (Hu et al., 2022) to

[2]We choose to use the chat version of `Llama2` as it has better alignment with human preferences, and is a more realistic fit for a SimulMT use.

MODEL	EN-CS	EN-DE	EN-ES	EN-FR	EN-IT	EN-NL	EN-PT	EN-RO	EN-RU	AVG	BL/AL
OFFLINE BASELINES (I)											
Transformer	22.31	30.82	35.19	42.95	31.54	35.04	38	29.71	20.04	31.73	-
OFFLINE LLM (II)											
LLM-One-Shot	9.55	21.44	26.80	30.70	18.68	23.35	23.01	14.63	12.40	20.06	-
LLM-PFX-SFT	**20.27**	**30.88**	**36.65**	**42.68**	**32.04**	**33.11**	**37.63**	**27.27**	**21.15**	**31.30**	-
SIMULTANEOUS BASELINES (III)											
Transformer	21.10	29.24	33.67	42.09	30.13	33.87	36.77	29.40	19.15	30.60 (8.60)	3.544
Transformer⋆	17.19	24.20	29.34	35.84	25.67	29.37	30.45	24.42	16.38	25.87 (4.81)	**5.366**
SIMULTANEOUS ONE-SHOT-LLM (IV)											
LLM-One-Shot	10.31	21.34	27.54	30.74	19.25	**23.77**	23.50	14.95	12.79	20.47 (11.65)	1.768
LLM-One-Shot⋆	**11.19**	**22.03**	**27.59**	**31.27**	**20.32**	23.68	**24.13**	**15.48**	**13.70**	**21.04 (7.29)**	**2.903**
SIMULTANEOUS SFT-LLM (V)											
LLM-PFX-SFT	20.22	30.52	36.34	41.70	**31.88**	**34.11**	36.85	26.38	**21.28**	31.03 (12.23)	2.538
LLM-PFX-SFT⋆	**21.31**	**31.06**	**36.34**	**42.59**	31.53	33.92	**37.56**	**27.03**	20.66	**31.33 (7.62)**	**4.117**

Table 2: This table presents the overall results. They are classified into five groups, where the first two groups are offline results, and the rest three groups are simultaneous results. Models annotated with ⋆ are using RALCP ($\gamma = 0.6$), and others are with LCP ($\gamma = 1.0$). For LLM results, LLM-PFX-SFT stands for the model tuned with the combination of full sentences and prefixes (introduced in §3.4). The metrics are annotated as **BLEU** for offline results and **BLEU (LAAL)** for simultaneous results (Note that due to space limitation, we only present LAAL on the average column in this table, full results are presented in Table 7). The best results within each group are **bolded** (in terms of BLEU) and/or colored red (in terms of LAAL). The last column (BL/AL) is the normalized BLEU over LAAL obtained from the average (Avg) column, meaning the BLEU score acquired from each latency unit.

reduce the computation overhead, LoRA adapters were configured with $r = 64$ and $\alpha = 16$, thus having the total trainable parameters to be 33M. We set the learning rate to 2e-4, the batch size to 48, and employed 4-bit quantization. For all experiments involving an LLM, a single A100 GPU is used. SFT is done only for one epoch, except when stated otherwise.

Baselines We established a baseline model i.e. an offline NMT-Transformer(Vaswani et al., 2017) consists of 6 encoder and decoder layers, trained on full-sentence parallel data (but with source sentences prepended with a language tag for multilingual training) from scratch for 300K steps with 16k tokens per batch on 4 A40 GPUs, the parameter size of it is 48M. It used the same decoding policy as the LLM, but processed incremental source and target text with the encoder and decoder separately, similar to the implementation of (Polák et al., 2022; Guo et al., 2023).

4.2 Experimental Results

Table 2 presents our primary experimental results. Our experiments are divided into two scenarios and 5 groups, i.e. offline (group I and II) and simultaneous (group III-V). For each scenario, we evaluated the performance of baseline models, and the LLM under one-shot and SFT settings (we found that LLM under zero-shot setting often generates unexpected format in the response, the detail of the one-shot setting can be found in Appendix C.2). For each model in the simultaneous scenario, we evaluated them with both LCP ($\gamma = 1.0$) and RALCP ($\gamma = 0.6$, annotated with $\star$), the reason for choosing $\gamma = 0.6$ is discussed in Appendix C.4. We set $n = 6$ for all simultaneous models because of the moderate latency it leads to. For all models in both scenarios reported in Table 2, we set the beam size as 10. More results using different hyper-parameter configurations and evaluation metrics such as COMET (Rei et al., 2020) are reported in Appendix C.5. The following findings can be summarized in Table 2.

Offline scenario We observe a substantial performance gap between LLM's one-shot setting and the baseline model (an average difference of 10 points). Despite the fact that fine-tuning Llama2 achieved performance similar to that of the NMT-Transformer, it still fell short of our expectations, where we anticipated that a larger model would yield better results. We offer the following reasonable hypothesis for this outcome: according to findings by Allen-Zhu and Li (2024), LLMs primarily acquire knowledge during the pre-training phase, and the efficiency of learning additional knowledge in the SFT phase is quite limited. This could explain why, despite using a substantial amount of training data, the model was unable to further acquire multilingual knowledge, ultimately reaching a plateau in translation capability. Additionally,

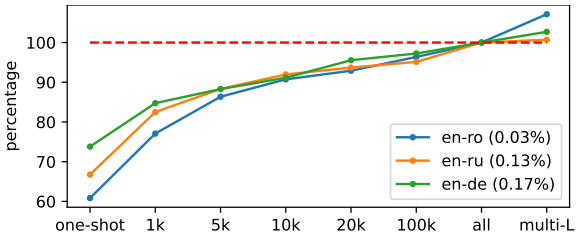

Figure 3: This figure illustrates how SimulMT performance (BLEU) is maintained (in %) with reduced data, in comparison to training on the full dataset (all): (*i*) one-shot, (*ii*) varying amount of training size from 1K to 100K and (*iii*) multilingual SFT on all data (multi-L). The legend shows the language pair and its coverage in `Llama2` pretraining data.

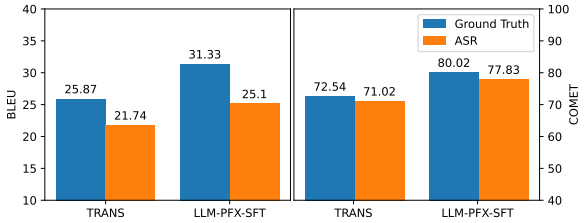

Figure 4: The performance in BLEU and COMET of baseline methods and LLM with ground truth or ASR transcripts as input. (Averaging across 9 language pairs)

since we performed SFT with LoRA for only one epoch, and the number of learnable parameters in LoRA is smaller than that of the NMT-Transformer, this further constrained the model's translation abilities.

Simultaneous scenario We found that both LLM-One-Shot's and LLM-PFX-SFT's remained on par with its offline scenario results indicating the robustness of the read-n & incremental-decoding approach on LLM.

Benefits of RALCP All simultaneous results demonstrated that RALCP effectively reduced latency (around 45%). In the case of baseline models, RALCP had a noticeable negative impact on BLEU. However, for LLM, it managed to keep BLEU unchanged. We speculate this is because LLM's decoder-only structure ensures a monotonic dependency on source context, guaranteeing higher consistency in beam candidates. Consequently, RALCP effectively reduces latency while maintaining prefix quality. For baseline models, the use of RALCP resulted in errors due to the inherent non-monotonic nature of bi-directional encoders, which led to higher uncertainty and diversity in beam candidates. This issue is also discussed in (Liu et al., 2020). In conclusion, our results indicate that RALCP is better suited for models with a monotonic dependency on source context.

5 Analysis

5.1 Data Utilization Efficiency

Figure 3 presents the percentage of performance retained after SFT using different data sizes ranging from 1k to 100k, compared to the performance achieved with full data (denoted as all) on three representative language pairs (en-de, en-ro, en-ru). We also provide the one-shot performance as the baseline and the best performance obtained by multilingual SFT (described in §3.4) denoted as multi-L. We can observe a high correlation between language coverage (see Table 1, column "Pretraining Coverage") in the pretraining corpus of `Llama2` and the retained translation performance in the one-shot setting. There are 2 interesting observations we can mention here to emphasise the benefit of LLM: (*i*) 1k samples can provide significant improvement compared to one-shot decoding, but still not sufficient for low-resource language. (*ii*) With only 10k samples, it retains 90% performance and closes the gap between low and high-resource language. Detailed experimental setup and results are shown in Appendix C.3.

5.2 Robustness of Noisy Inputs

To further investigate the potential advantages of LLM in the SimulMT task, we evaluated LLM's performance when using ASR transcripts as inputs. To ensure consistency in inputs for different methods, we did not directly use a streaming ASR system during inference. Instead, we first used Whisper-base (Radford et al., 2023) to generate transcripts (with an average WER of 17.31) for test sets of all 9 language pairs, which were then used as inputs for SimulMT, replacing the previous ground-truth inputs.

For this experiment, we employed both BLEU and COMET (Rei et al., 2020) as evaluation metrics. We included COMET because assessing model robustness in noisy input scenarios requires more than just n-gram matching in BLEU. Figure 4 displays the averaged BLEU and COMET scores for all 9 language pairs using three models with ground truth and ASR as inputs. For both BLEU and COMET scores, LLM outperforms dedicated NMT models by a large margin, indicating that LLM has better robustness on the noisy input.

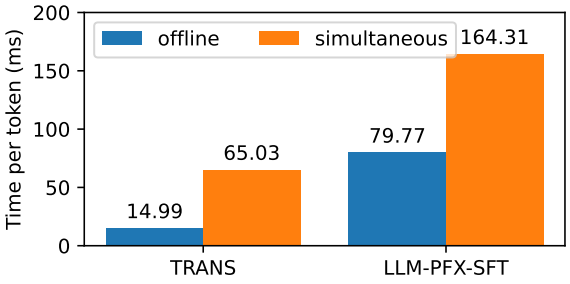

Figure 5: The average time of predicting one target token (in milliseconds) of baseline models and LLM under offline and simultaneous scenarios.

5.3 Inference Efficiency

Compared to the Transformer baseline, LLM has a larger number of parameters, which typically incurs higher inference costs. Figure 5 illustrates the average time it takes to predict a single token in both offline and simultaneous scenarios. This time is obtained by averaging the actual wall time across all hypothesis lengths for the three test sets (en-de, en-ro, en-ru), which also accounts for the time spent on model calls wasted due to RALCP failing to select a prefix during incremental decoding. As shown in the figure, LLM consumes more time in both scenarios compared to the other baseline methods. This suggests that in real-world usage, LLM must consider the additional latency brought about by computational expenses.

6 Related Works

Simultaneous Machine Translation (SimulMT) is the task to provide real-time translation of a source sentence stream where the goal is to minimize the latency while maximizing the translation quality. A common approach is to train a MT model on prefix-to-prefix dataset to directly predict target tokens based on partial source tokens (Ma et al., 2019b). Alternatively, Liu et al. (2020) proposed the incremental decoding framework to leverage the pretrained offline NMT and turn it into a SimulMT model without further training. A core component of SimulMT is a read-write policy to decide at every step whether to wait for another source token (`READ`) or to generate a target token (`WRITE`). Previous methods have explored fixed policy, which always waits for k tokens before generation (Ma et al., 2019b; Zhang et al., 2022) and adaptive policy, which trains an agent via reinforcement learning (Gu et al., 2017b; Arthur et al., 2021b). Re-translation (Arivazhagan et al., 2019) from the beginning of the source sentence at the `WRITE` step will incur high translation latency. Stable hypothesis detection methods such as Local Agreement (Liu et al., 2020), hold-n (Liu et al., 2020) and Share prefix SP-n (Nguyen et al., 2021b) are employed to commit stable hypothesis and only regenerate a subsequence of source sentence. The goal is to reduce the latency and minimize the potential for errors resulting from incomplete source sentence (Polák et al., 2022).

LLM for NMT Recent research has delved into the potential usage of LLMs in MT (Hendy et al., 2023; Zhu et al., 2023; Robinson et al., 2023). While LLMs do exhibit some level of translation capability, prior research has identified that they still lags behind the conventional NMT models, especially for low resource languages (Robinson et al., 2023). Additionally, the translation performance varies depending on prompting strategies (Zhang et al., 2023). Efforts have been made to enhance the translation performance of LLMs by incorporating guidance from dictionary (Lu et al., 2023), further fine-tuning (Zeng et al., 2023; Xu et al., 2023) and augmenting with translation memories (Mu et al., 2023). However, to the best of our knowledge, there is a lack of research exploring the simultaneous translation capability of LLMs.

7 Conclusion

In this paper, we focus on exploring the feasibility of applying LLM to SimulMT. We initially transformed the `Llama2-7B-chat` into a model that supports simultaneous translation using the existing incremental-decoding approach. We then introduced the RALCP algorithm to reduce inference latency. In our experiments, we found that the LLM after SFT could outperform the dedicated NMT model using the same decoding policy, showcasing the potential of LLM in this task. Additionally, we observed that LLM exhibited a degree of robustness against noisy input and could offer effective improvements through supervised fine-tuning with limited data. However, we also identified that the computational overhead of LLM is a significant challenge. In future work, we intend to propose policies more suitable for LLM and further explore the possible applications of various LLM capabilities in SimulMT tasks.

Limitations

We summarize the limitations of this study in three aspects:

Policy In this paper, we only explored a relatively simple policy, i.e. "read-n & incremental-decoding". Especially, the decision-making process for the READ action is almost naive. We recognize that the frequent LLM invocation for full-stop generation due to the inefficiency of the policy is a major factor for the high computational overhead. In future work, we aim to explore more adaptive and efficient policies.

Data Our evaluation was conducted solely on the MUST-C dataset, which has limited the domain and style diversity. We believe that richer datasets should be considered to allow for a more comprehensive evaluation of the approach.

Usage of LLM Currently, we only investigated the possibility of using LLM as a translation model in the entire SimulMT pipeline. However, LLM has capabilities beyond translation. In our future work, we plan to fully leverage LLM's multitasking capabilities and explore more diverse usage patterns in the pipeline.

These limitations provide directions for future research to further enhance the applicability and performance of LLM in the SimulMT task.

References

Sweta Agrawal, Antonios Anastasopoulos, Luisa Bentivogli, Ondrej Bojar, Claudia Borg, Marine Carpuat, Roldano Cattoni, Mauro Cettolo, Mingda Chen, William Chen, Khalid Choukri, Alexandra Chronopoulou, Anna Currey, Thierry Declerck, Qianqian Dong, Kevin Duh, Yannick Estève, Marcello Federico, Souhir Gahbiche, Barry Haddow, Benjamin Hsu, Phu Mon Htut, Hirofumi Inaguma, Dávid Javorský, John Judge, Yasumasa Kano, Tom Ko, Rishu Kumar, Pengwei Li, Xutai Ma, Prashant Mathur, Evgeny Matusov, Paul McNamee, John P. McCrae, Kenton Murray, Maria Nadejde, Satoshi Nakamura, Matteo Negri, Ha Nguyen, Jan Niehues, Xing Niu, Atul Kr. Ojha, John E. Ortega, Proyag Pal, Juan Pino, Lonneke van der Plas, Peter Polák, Elijah Rippeth, Elizabeth Salesky, Jiatong Shi, Matthias Sperber, Sebastian Stüker, Katsuhito Sudoh, Yun Tang, Brian Thompson, Kevin Tran, Marco Turchi, Alex Waibel, Mingxuan Wang, Shinji Watanabe, and Rodolfo Zevallos. 2023. Findings of the IWSLT 2023 evaluation campaign. In *Proceedings of the 20th International Conference on Spoken Language Translation, IWSLT@ACL 2023, Toronto, Canada (in-person and online), 13-14 July, 2023*, pages 1–61. Association for Computational Linguistics.

Zeyuan Allen-Zhu and Yuanzhi Li. 2024. Physics of language models: Part 3.1, knowledge storage and extraction.

Naveen Arivazhagan, Colin Cherry, Te I, Wolfgang Macherey, Pallavi Baljekar, and George F. Foster. 2020. Re-translation strategies for long form, simultaneous, spoken language translation. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*, pages 7919–7923. IEEE.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. 2019. Monotonic infinite lookback attention for simultaneous machine translation. In *ACL*, pages 1313–1323.

Philip Arthur, Trevor Cohn, and Gholamreza Haffari. 2021a. Learning coupled policies for simultaneous machine translation using imitation learning. In *EACL*, pages 2709–2719.

Philip Arthur, Trevor Cohn, and Gholamreza Haffari. 2021b. Learning coupled policies for simultaneous machine translation using imitation learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2709–2719, Online. Association for Computational Linguistics.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *CoRR*, abs/2302.04023.

Chung-Cheng Chiu and Colin Raffel. 2017. Monotonic chunkwise attention. *CoRR*, abs/1712.05382.

Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation? *CoRR*, abs/1606.02012.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 644–648. The Association for Computational Linguistics.

Mattia Antonino Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2019. Must-c: a multilingual speech translation corpus. In *NAACL-HLT*, pages 2012–2017.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O. K. Li. 2017a. Learning to translate in real-time with neural machine translation. In *EACL*, pages 1053–1062.

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

Jiaxin Guo, Daimeng Wei, Zhanglin Wu, Zongyao Li, Zhiqiang Rao, Minghan Wang, Hengchao Shang, Xiaoyu Chen, Zhengzhe Yu, Shaojun Li, Yuhao Xie, Lizhi Lei, and Hao Yang. 2023. The hw-tsc's simultaneous speech-to-text translation system for IWSLT 2023 evaluation. In *IWSLT@ACL*, pages 376–382.

Amr Hendy, Mohamed Abdelrehim, Amr Sharaf, Vikas Raunak, Mohamed Gabr, Hitokazu Matsushita, Young Jin Kim, Mohamed Afify, and Hany Hassan Awadalla. 2023. How good are GPT models at machine translation? A comprehensive evaluation. *CoRR*, abs/2302.09210.

Dongyang Hu and Junhui Li. 2022. Contrastive learning for robust neural machine translation with ASR errors. In *Natural Language Processing and Chinese Computing - 11th CCF International Conference, NLPCC 2022, Guilin, China, September 24-25, 2022, Proceedings, Part I*, volume 13551 of *Lecture Notes in Computer Science*, pages 81–91. Springer.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *ICLR*.

Danni Liu, Gerasimos Spanakis, and Jan Niehues. 2020. Low-latency sequence-to-sequence speech recognition and translation by partial hypothesis selection. In *Interspeech*, pages 3620–3624.

Hongyuan Lu, Haoyang Huang, Dongdong Zhang, Haoran Yang, Wai Lam, and Furu Wei. 2023. Chain-of-dictionary prompting elicits translation in large language models. *CoRR*, abs/2305.06575.

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. 2019a. STACL: simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *ACL*, pages 3025–3036.

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. 2019b. 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 Miguel Pino. 2020. SIMULEVAL: an evaluation toolkit for simultaneous translation. In *EMNLP*, pages 144–150.

Yongyu Mu, Abudurexiti Reheman, Zhiquan Cao, Yuchun Fan, Bei Li, Yinqiao Li, Tong Xiao, Chunliang Zhang, and Jingbo Zhu. 2023. Augmenting large language model translators via translation memories. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10287–10299, Toronto, Canada. Association for Computational Linguistics.

Thai-Son Nguyen, Sebastian Stüker, and Alex Waibel. 2021a. Super-human performance in online low-latency recognition of conversational speech. In *Interspeech*, pages 1762–1766.

Thai-Son Nguyen, Sebastian Stüker, and Alex Waibel. 2021b. Super-Human Performance in Online Low-Latency Recognition of Conversational Speech. In *Proc. Interspeech 2021*, pages 1762–1766.

Jan Niehues, Ngoc-Quan Pham, Thanh-Le Ha, Matthias Sperber, and Alex Waibel. 2018. Low-Latency Neural Speech Translation. In *Proc. Interspeech 2018*, pages 1293–1297.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

Sara Papi, Marco Gaido, Matteo Negri, and Marco Turchi. 2022a. Does simultaneous speech translation need simultaneous models? In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 141–153. Association for Computational Linguistics.

Sara Papi, Marco Gaido, Matteo Negri, and Marco Turchi. 2022b. Over-generation cannot be rewarded: Length-adaptive average lagging for simultaneous speech translation. *CoRR*, abs/2206.05807.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Peter Polák, Ngoc-Quan Pham, Tuan-Nam Nguyen, Danni Liu, Carlos Mullov, Jan Niehues, Ondrej Bojar, and Alexander Waibel. 2022. CUNI-KIT system for simultaneous speech translation task at IWSLT 2022. In *IWSLT@ACL*, pages 277–285.

Peter Polák, Ngoc-Quan Pham, Tuan Nam Nguyen, Danni Liu, Carlos Mullov, Jan Niehues, Ondřej Bojar, and Alexander Waibel. 2022. CUNI-KIT system for simultaneous speech translation task at IWSLT 2022. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 277–285, Dublin, Ireland (in-person and online). Association for Computational Linguistics.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 28492–28518. PMLR.

Colin Raffel, Minh-Thang Luong, Peter J. Liu, Ron J. Weiss, and Douglas Eck. 2017. Online and linear-time attention by enforcing monotonic alignments. In *ICML*, volume 70, pages 2837–2846. PMLR.

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 2020, Online, November 16-20, 2020*, pages 2685–2702. Association for Computational Linguistics.

Nathaniel R. Robinson, Perez Ogayo, David R. Mortensen, and Graham Neubig. 2023. Chatgpt MT: competitive for high- (but not low-) resource languages. *CoRR*, abs/2309.07423.

Nicholas Ruiz and Marcello Federico. 2014. Assessing the impact of speech recognition errors on machine translation quality. In *Proceedings of the 11th Conference of the Association for Machine Translation in the Americas: MT Researchers Track, AMTA 2014, Vancouver, Canada, October 22-26, 2014*, pages 261–274. Association for Machine Translation in the Americas.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 5998–6008.

Minghan Wang, Jiaxin Guo, Yinglu Li, Xiaosong Qiao, Yuxia Wang, Zongyao Li, Chang Su, Yimeng Chen, Min Zhang, Shimin Tao, Hao Yang, and Ying Qin. 2022. The HW-TSC's simultaneous speech translation system for IWSLT 2022 evaluation. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 247–254, Dublin, Ireland (in-person and online). Association for Computational Linguistics.

Di Wu, Binbin Zhang, Chao Yang, Zhendong Peng, Wenjing Xia, Xiaoyu Chen, and Xin Lei. 2021. U2++: unified two-pass bidirectional end-to-end model for speech recognition. *CoRR*, abs/2106.05642.

Haoran Xu, Young Jin Kim, Amr Sharaf, and Hany Hassan Awadalla. 2023. A paradigm shift in machine translation: Boosting translation performance of large language models. *CoRR*, abs/2309.11674.

Wen Yang, Chong Li, Jiajun Zhang, and Chengqing Zong. 2023. Bigtranslate: Augmenting large language models with multilingual translation capability over 100 languages.

Jiali Zeng, Fandong Meng, Yongjing Yin, and Jie Zhou. 2023. TIM: teaching large language models to translate with comparison. *CoRR*, abs/2307.04408.

Biao Zhang, Barry Haddow, and Alexandra Birch. 2023. Prompting large language model for machine translation: A case study. *ArXiv*, abs/2301.07069.

Shaolei Zhang, Shoutao Guo, and Yang Feng. 2022. Wait-info policy: Balancing source and target at information level for simultaneous machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2249–2263, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Wenhao Zhu, Hongyi Liu, Qingxiu Dong, Jingjing Xu, Lingpeng Kong, Jiajun Chen, Lei Li, and Shujian Huang. 2023. Multilingual machine translation with large language models: Empirical results and analysis. *CoRR*, abs/2304.04675.

Appendix

A Prefix Quality Evaluation

Method	EN-CS	EN-DE	EN-ES	EN-FR	EN-IT	EN-NL	EN-PT	EN-RO	EN-RU
RatioCut	18.64	13.90	22.05	19.80	19.38	19.34	20.59	19.71	14.68
ChatGPT	21.40	26.77	36.45	32.80	30.04	28.75	27.90	25.43	19.13

Table 3: This table presents the BLEU score of the created prefixes using length-ratio-based truncation or using ChatGPT.

To ensure the quality of the translation prefixes generated by ChatGPT (§3.4), we performed a basic evaluation on them. First of all, for each language, we use the `fast_align` (Dyer et al., 2013) toolkit to learn the alignment on full sentence pairs. Then, a golden prefix reference set is created based on the randomly truncated source text (the input for ChatGPT) and the learned alignment table. Finally, we evaluate the BLEU score of the hypothesis of ChatGPT. A baseline approach is also explored by directly using the length ratio to cut target text based on the source prefix length. Results in Table 3 demonstrate that the quality of ChatGPT is reasonable and better than the length-ratio-based truncation.

B Instruction Template for SFT

Translate the following sentence: {src_text} from {src_lang} to {tgt_lang}.
I need a translation from {src_lang} to {tgt_lang} for the text: {src_text}.
Please translate {src_text} from {src_lang} to {tgt_lang}.
Could you help me translate {src_text} from {src_lang} to {tgt_lang}?
I require a translation of {src_text} from {src_lang} to {tgt_lang}.
Take the sentence {src_text} in {src_lang} and translate it to {tgt_lang}.
Translate {src_text} from {src_lang} to {tgt_lang}.
Provide me with a translation from {src_lang} to {tgt_lang} for the text: {src_text}.
I'm looking for a translation of {src_text} from {src_lang} to {tgt_lang}.
Translate the sentence {src_text} from {src_lang} to {tgt_lang}.

Table 4: This table shows the ten prompt templates used in the SFT.

C Complementary Experimental Details

C.1 Latency Measurement

The computation of LAAL (Papi et al., 2022b) is defined as:

$$\text{LAAL} = \frac{1}{\tau}\sum_{i}^{\tau} d_i - (i-1)\frac{|S|}{max(|T|, |\hat{T}|)},$$

where $S, T, \hat{T}$ represent source, reference and hypothesis, $\tau = \arg\min_i(d_i = |S|)$ is the normalization factor, $d_i = j, j <= |S|$ is the delay of hypothesis T_i represented by the index j of the source word S_j at which T_i is predicted.

C.2 One-Shot Prompts

We follow the method introduced in (Touvron et al., 2023b) to perform one-shot inference by creating the prompt with a complete round of dialogue with a system message. Specifically, the example used in the prompt is "Good morning." in English as the source context and a translation in the target language. We consider this example as a complete dialogue history in the prompt with a system message placed before it, which looks like: "**<s><<SYS>>\nYou are a professional translator, you should try your best to provide translation with good quality, no explanations are required.\n<</SYS>>\n\n[INST] Translate the following sentence from English to German: {Good morning.} [/INST] {Guten Morgen.}</s><s>[INST] Translate the following sentence from English to German:** S_i^t **[/INST]** T_j^t", where S_i^t and T_j^t are incremental source and target text being processed.

C.3 Experimental Setup and Results for §5.1

Data Scale	Effective Batch Size	# Epoch	# Train step
1k	8	5	625
5k	8	1	625
10k	32	5	1563
20k	32	2	1250
100k	48	1	2084
BiL-all (*220k*)	32×4	1	*1800*
MultiL-mix (2M)	48×2	1	20.8k

Table 5: This table presents the detailed SFT hyper-parameters under different data scales. Values with italics represent an averaged value across languages. BiL-all stands for using all available bilingual training set for the specific language pair, and MultiL-mix stands for the mixed multilingual dataset (without prefix) introduced in §3.4. The effective batch size stands for the batch size times gradient accumulation steps. All models are trained using 1 A100 GPU.

Language Pair	One-shot	1k	5k	10k	20k	100k	all	Multi-L
EN-DE (0.17%)	22.03	25.30	26.36	27.21	28.52	29.03	29.85	30.66
EN-RO (0.03%)	15.48	19.61	21.97	23.09	23.64	24.52	25.44	27.26
EN-RU (0.13%)	13.70	16.93	18.13	18.88	19.23	19.53	20.52	20.67

Table 6: The BLEU score for all three language pairs under different data scales.

For the investigation of data utilization efficiency, we ensured fair comparisons by setting appropriate training parameters to guarantee that the models converge properly. Thus, based on the data size, we configured the hyper-parameters listed in Table 5 for SFT. The detailed BLEU scores are shown in

Table 6. We use $n = 6, \gamma = 0.6$, and beam size as 10 for all models during inference.

C.4 Ablation Study on Policy Hyper-parameters

We conducted a detailed ablation study on three hyperparameters: n, γ, and beam size. These experiments were primarily conducted on en-de, en-ro, and en-ru language pairs due to their distinct characteristics such as scripts, belonging to different Genus categories, and variations in pretraining language coverage, making them highly representative choices.

As shown in Figure 6, we separately illustrate the impact of different n, γ, and beam size settings on BLEU and LAAL. Regarding the exploration of n, we kept γ and beam size fixed at 0.6 and 10, respectively. The results show that n has a relatively minor influence on BLEU, typically achieving stable performance when $n > 3$. However, the impact of n on LAAL is linear, which aligns with the operational pattern of the policy itself.

For the investigation of γ, we set n to 6 and beam size to 10. It is observed that gamma has a certain effect on BLEU, but it is not linear. The better results tend to cluster around a value of approximately 0.6. This implies that when γ is too large, it leads to a significant increase in latency without necessarily improving the results. This observation underscores the effectiveness of RALCP, as it can reduce latency effectively without compromising quality.

In the exploration of beam size, we set n to 6 and γ to 0.6. Beam size exhibits a linear correlation with BLEU, though not highly significant. However, its impact on latency is more pronounced. This is mainly because a larger beam size makes it more challenging for RALCP to select common prefixes, resulting in more wasted LLM calls and increased latency. Additionally, we noticed that LAAL exhibits regular peaks at beam sizes of 5, 7, and 9. This phenomenon may be attributed to rounding errors during RALCP's voting process, reducing the chances of tokens being selected. It motivates us to explore improved mechanisms for local agreement identification.

C.5 Additional Details in the Main Experiment

In Table 7 and Table 8, we provide more experimental results evaluated with both of BLEU and COMET score (Rei et al., 2020), which are further divided into 10 groups compared to Table 2. These groups include the performance in offline decoding with two different beam sizes and the performance in simultaneous decoding under various latency degrees controlled by n. Specifically, for the simultaneous mode, we categorized the results into low-latency (beam size=5, n=3) and high-latency (beam size=10, n=6) configurations.

Consistent Effectiveness of RALCP Similarly, we also compared the results for each model using LCP and RALCP. Across different latency levels, RALCP exhibits similar latency reduction effects, consistent with the findings in section §4.2.

Ineffectiveness of Prefix data Furthermore, we also compared the results for LLM using SFT with and without the use of prefix data. We found that prefix data does not seem to have a positive impact on LLM in terms of quality and latency. The final results are almost identical to those without using prefix data. This may be related to the relatively small scale of the prefix data. However, due to cost constraints, we didn't construct a larger prefix dataset, so further exploration in this area is left for future work.

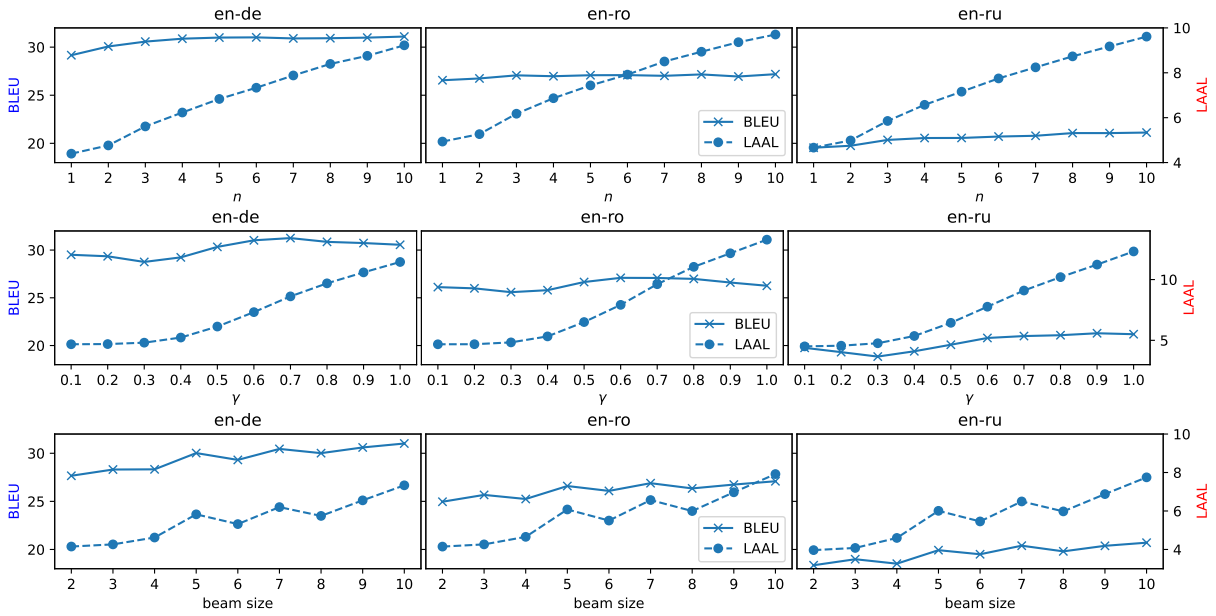

Figure 6: The correlation between BLEU and LAAL under different n, γ and beam size.

Model	en-cs	en-de	en-es	en-fr	en-it	en-nl	en-pt	en-ro	en-ru	Avg	BL/LA
Offline Baselines (b=5) (I)											
Transformer	22.29	30.65	35.08	42.91	31.46	34.91	38.05	29.58	20.09	31.669	-
Offline Baselines (b=10) (II)											
Transformer	22.31	30.82	35.19	42.95	31.54	35.04	38	29.71	20.04	31.733	-
Offline LLM (b=5) (III)											
LLM-One-Shot	10.37	21.79	27.4	31.25	19.71	23.8	23.87	15.44	13.4	20.781	-
LLM-SFT	20.47	30.73	36.43	42.77	**32.05**	**34.51**	37.58	**27.45**	20.65	**31.404**	-
LLM-PFX-SFT	**20.73**	**30.93**	**36.47**	**42.89**	31.91	33.87	**37.66**	27.15	**21.02**	31.403	-
Offline LLM (b=10) (IV)											
LLM-One-Shot	9.552	21.439	26.8	30.7	18.681	23.345	23.009	14.631	12.404	20.062	-
LLM-SFT	**20.405**	30.621	36.589	42.561	**32.14**	**33.648**	37.501	27.126	20.677	31.252	-
LLM-PFX-SFT	20.267	**30.88**	**36.653**	**42.682**	32.041	33.105	**37.633**	**27.296**	**21.153**	**31.301**	-
Simultaneous Baselines (Low-latency, b=5, n=3) (V)											
Transformer	19.45 (5.45)	27.48 (5.54)	32.54 (6.57)	40.10 (6.28)	29.23 (6.65)	32.43 (6.36)	35.07 (6.65)	28.00 (7.33)	18.10 (5.97)	29.156 (6.311)	4.610
Transformer⋆	14.11 (2.72)	19.73 (2.83)	25.37 (3.17)	30.50 (3.03)	21.83 (3.19)	25.41 (3.13)	25.79 (3.06)	20.60 (3.32)	13.52 (2.91)	21.873 (3.040)	7.163
Simultaneous Baselines (High-latency, b=10, n=6) (VI)											
Transformer	21.10 (7.72)	29.24 (7.93)	33.67 (8.71)	42.09 (8.60)	30.13 (8.87)	33.87 (8.71)	36.77 (9.27)	29.40 (9.29)	19.15 (8.34)	30.602 (8.604)	3.544
Transformer⋆	17.19 (4.58)	24.20 (4.61)	29.34 (4.88)	35.84 (4.78)	25.67 (4.95)	29.37 (4.87)	30.45 (4.91)	24.42 (4.95)	16.38 (4.78)	25.873 (4.812)	**5.366**
Simultaneous One-Shot-LLM (Low-latency, b=5, n=3) (VII)											
LLM-One-Shot	**11.70 (7.72)**	**22.38 (7.29)**	**27.75 (8.38)**	**31.89 (8.22)**	**20.43 (8.19)**	**24.02 (7.60)**	**24.32 (8.58)**	**15.80 (8.13)**	**13.65 (8.40)**	**21.327 (8.057)**	2.648
LLM-One-Shot⋆	10.63 (4.07)	19.10 (3.81)	24.48 (3.92)	28.57 (4.03)	17.12 (4.03)	20.89 (3.71)	21.86 (4.03)	14.21 (4.08)	12.63 (4.12)	18.832 (3.978)	**4.757**
Simultaneous One-Shot-LLM (High-latency, b=10, n=6) (VIII)											
LLM-One-Shot	**10.31 (11.66)**	21.34 (10.64)	27.54 (12.00)	30.74 (11.43)	19.25 (11.97)	**23.77 (10.93)**	**23.50 (11.99)**	14.95 (11.99)	12.79 (12.20)	20.466 (11.646)	1.768
LLM-One-Shot⋆	11.19 (7.41)	**22.03 (6.88)**	**27.59 (7.18)**	**31.27 (7.28)**	**20.32 (7.41)**	23.68 (6.91)	24.13 (7.43)	**15.48 (7.52)**	**13.70 (7.60)**	**21.043 (7.291)**	**2.903**
Simultaneous SFT-LLM (Low-latency, b=5, n=3) (IX)											
LLM-SFT	20.62 (7.69)	30.51 (7.94)	**36.66 (9.12)**	42.50 (8.64)	31.96 (9.02)	**34.28 (8.22)**	37.28 (9.48)	**27.19 (9.21)**	**20.86 (7.89)**	31.318 (8.579)	3.634
LLM-SFT⋆	19.09 (4.02)	28.31 (4.07)	33.82 (4.15)	41.23 (4.19)	29.46 (4.24)	30.87 (3.92)	35.05 (4.38)	25.67 (4.30)	18.29 (4.05)	29.088 (4.147)	**7.001**
LLM-PFX-SFT	**21.01 (8.16)**	**31.02 (8.58)**	36.63 (9.34)	**42.69 (9.15)**	**31.97 (9.47)**	34.03 (8.32)	**37.47 (9.68)**	27.11 (9.66)	20.80 (8.80)	**31.414 (9.018)**	3.476
LLM-PFX-SFT⋆	19.80 (4.21)	28.80 (4.15)	33.86 (4.40)	41.34 (4.29)	29.07 (4.36)	31.46 (3.99)	34.87 (4.41)	25.89 (4.40)	19.21 (4.29)	29.367 (4.278)	6.866
Simultaneous SFT-LLM (High-latency, b=10, n=6) (X)											
LLM-SFT	20.29 (11.49)	30.30 (11.57)	36.06 (12.73)	41.52 (12.14)	31.62 (12.62)	**34.19 (11.98)**	36.38 (13.40)	26.39 (13.00)	20.82 (12.09)	30.841 (12.336)	2.496
LLM-SFT⋆	**21.32 (7.29)**	30.66 (7.18)	**36.52 (7.67)**	42.20 (7.53)	31.68 (7.79)	34.09 (7.23)	37.40 (8.08)	**27.26 (7.97)**	20.67 (7.45)	31.311 (7.577)	**4.130**
LLM-PFX-SFT	20.22 (11.45)	30.52 (11.47)	36.34 (12.44)	41.70 (12.20)	**31.88 (12.53)**	34.11 (11.46)	36.85 (12.97)	26.38 (13.32)	**21.28 (12.28)**	31.031 (12.236)	2.538
LLM-PFX-SFT⋆	21.31 (7.38)	**31.06 (7.31)**	36.34 (7.72)	**42.59 (7.61)**	31.53 (7.72)	33.92 (7.08)	**37.56 (8.03)**	27.03 (7.91)	20.66 (7.82)	**31.333 (7.620)**	4.117

Table 7: This table is the full version of Table 2 which further includes results under different configurations. Results are further classified into 10 groups, with respect to offline/simultaneous mode, low latency (beam=5, $n = 6$), and high latency (beam=10, $n = 6$) mode. Models annotated with $\star$ are using RALCP ($\gamma = 0.6$), and others are with LCP ($\gamma = 1.0$). For LLM results, LLM-(PFX-)SFT stands for the model tuned with the pure offline full sentences w/wo prefixes (introduced in §3.4). The metrics are annotated as **BLEU** for offline results and **BLEU (LAAL)** for simultaneous results. The best results within each group are **bolded** (in terms of BLEU) and/or colored red (in terms of LAAL). The last column is the normalized BLEU over LAAL obtained from the average (Avg) column, meaning the BLEU score acquired from each latency unit.

MODEL	EN-CS	EN-DE	EN-ES	EN-FR	EN-IT	EN-NL	EN-PT	EN-RO	EN-RU	AVG	CM/LA
OFFLINE BASELINES (B=5) (I)											
Transformer	78.86	80.21	82.33	82.76	82.26	83.64	83.71	82.96	78.08	81.646	-
OFFLINE BASELINES (B=10) (II)											
Transformer	79.15	80.41	82.38	82.85	82.35	83.67	83.77	83.06	77.73	81.708	-
OFFLINE LLM (B=5) (III)											
LLM-One-Shot	69.38	77.85	81.92	81.06	78.06	79.47	81.45	75.74	73.8	77.637	-
LLM-SFT	**83.58**	**84.4**	85.13	**85.68**	85.45	**86.42**	**86.42**	85.46	**83.6**	**85.127**	-
LLM-PFX-SFT	83.49	84.3	**85.16**	85.66	**85.59**	86.31	86.34	**85.66**	83.57	85.120	-
OFFLINE LLM (B=10) (IV)											
LLM-One-Shot	68.41	77.43	81.71	80.76	77.37	79.2	81	74.68	72.19	76.972	-
LLM-SFT	**83.6**	**84.35**	**85.06**	85.58	85.48	**86.38**	**86.33**	85.35	**83.47**	**85.067**	-
LLM-PFX-SFT	83.49	84.29	**85.06**	**85.63**	**85.59**	86.23	86.27	**85.46**	83.4	85.047	-
SIMULTANEOUS BASELINES (LOW-LATENCY, B=5, N=3) (V)											
Transformer	76.14	77.79	81.29	81.11	81	82.38	82.38	81.98	76.69	80.084 (6.311)	12.690
Transformer⋆	67.38	68.64	75.79	73.64	74.91	76.05	75.4	75.62	70.35	73.087 (3.040)	**24.042**
SIMULTANEOUS BASELINES (HIGH-LATENCY, B=10, N=6) (VI)											
Transformer	77.73	79.24	81.82	82.08	81.72	83.28	83.19	82.7	77.57	81.037 (8.604)	9.418
Transformer⋆	72.27	74.31	78.64	78.11	78.13	79.61	79.25	78.66	73.78	76.973 (4.812)	**15.996**
SIMULTANEOUS ONE-SHOT-LLM (LOW-LATENCY, B=5, N=3) (VII)											
LLM-One-Shot	**69.48**	**77.61**	**81.62**	**81.06**	**78.36**	**79.42**	**81.51**	**76.04**	**74.1**	**77.689 (8.057)**	9.642
LLM-One-Shot⋆	66	73.31	78.59	77.46	74.01	75.05	78.16	72.28	71.36	74.024 (3.978)	**18.608**
SIMULTANEOUS ONE-SHOT-LLM (HIGH-LATENCY, B=10, N=6) (VIII)											
LLM-One-Shot	68.28	77.21	**81.55**	**80.76**	77.42	**79.05**	81.09	**75.26**	72.04	76.962 (11.646)	6.608
LLM-One-Shot⋆	**68.71**	**77.23**	81.4	80.6	**77.99**	78.93	**81.24**	75.15	**73.74**	**77.221 (7.291)**	**10.591**
SIMULTANEOUS SFT-LLM (LOW-LATENCY, B=5, N=3) (IX)											
LLM-SFT	**83.2**	**84.21**	84.86	**85.46**	85.23	**86.1**	**86.21**	85.23	**83.23**	**84.859 (8.579)**	9.891
LLM-SFT⋆	81.6	82.17	84.06	84.5	84.26	84.66	85.63	83.92	81.7	83.611 (4.147)	**20.162**
LLM-PFX-SFT	83.08	84.05	**84.91**	85.4	**85.28**	86	86.14	**85.36**	82.95	84.797 (9.018)	9.403
LLM-PFX-SFT⋆	81.47	82.26	83.97	84.35	84.21	84.77	85.3	84.31	81.78	83.602 (4.278)	19.542
SIMULTANEOUS SFT-LLM (HIGH-LATENCY, B=10, N=6) (X)											
LLM-SFT	83.1	**84.02**	84.71	85.14	85.19	**86.06**	85.95	84.86	83	84.670 (12.336)	6.864
LLM-SFT⋆	**83.44**	83.91	**84.92**	85.37	**85.29**	85.98	**86.18**	**85.24**	**83.19**	**84.836 (7.577)**	**11.196**
LLM-PFX-SFT	82.87	84	84.74	85.09	85.2	85.94	85.94	84.92	82.93	84.626 (12.236)	6.916
LLM-PFX-SFT⋆	83.1	83.76	84.79	**85.39**	85.15	85.89	86.11	85.15	83	84.704 (7.620)	11.116

Table 8: This table presents the COMET scores with the same structure as Table 7. LAAL results are only shown in the average column (Avg). The last column (CM/LA) is the normalized COMET score over LAAL obtained from the average (Avg) column. Best performed result (in terms of COMMET score) are **bolded**.