Simul-LLM: A Framework for Exploring High-Quality Simultaneous Translation with Large Language Models

Anonymous ACL submission

Abstract

Large language models (LLMs) with billions of parameters and pretrained on massive amounts of data are now capable of near or better than state-of-the-art performance in a variety of downstream natural language processing tasks. Neural machine translation (NMT) is one such task that LLMs have been applied to with great success. However, little research has focused on applying LLMs to the more difficult subset of NMT called simultaneous translation (SimulMT), where translation begins before the entire source context is available to the model. In this paper, we address key challenges facing LLMs fine-tuned for SimulMT, validate classical SimulMT concepts and practices in the context of LLMs, explore adapting LLMs that are fine-tuned for NMT to the task of SimulMT, and introduce *Simul-LLM*[1], the first open-source fine-tuning and evaluation pipeline development framework for LLMs focused on SimulMT.

1 Introduction

Modern large language models (LLMs) contain at least several billion and up to trillions of parameters and are remarkably capable across a wide range of tasks. Pretrained on humongous amounts of unlabeled data, they have demonstrated incredible emergent capabilities. With minor prompt adjustments, such as including instructions and examples, LLMs are often capable of near state-of-the-art performance set by highly customized solutions. The performance of these models is further enhanced when fine-tuned for specialized downstream tasks, often exceeding the performance of previously cutting-edge solutions. Given their rapidly evolving capabilities, LLMs and their application have become a focused topic of research within NLP academia.

One popular downstream task for LLMs is text-to-text neural machine translation (NMT), which focuses on taking an input sequence in a given language and outputting a translation in another language. Typically, the entire source context is available at the start of translation for NMT. A particularly challenging subset of NMT is known as simultaneous translation (SimulMT), where the model begins translation without having access to the entire source sequence, and the translation progresses as the remaining source sequence is incrementally provided. For languages that are syntactically and structurally similar, near-NMT performance is fairly achievable, but for language pairs that differ significantly in structure, traditional models struggle to balance high-quality translations with delay for additional source context. This balance is typically achieved via a fixed or adaptive read-write schedule, with one of the most popular and longstanding fixed schedules being the *wait-k* policy (Ma et al., 2019), where the target translation hypothesis lags behind the incrementally available source sequence by k words or subwords.

While LLMs have been applied to and studied actively in NMT, their application to simultaneous translation has been lagging. This is in part due to a few challenges LLMs face when applied to SimulMT that are non-trivial to address. First and foremost, it is unclear how well LLMs, which are pretrained and usually fine-tuned under the assumption that the prompt is completely provided and static before generation, will adapt to an application space where the prompt dynamically changes as the simultaneous scheduler elects to read from the source sequence. Second, multiple approaches exist to enable LLMs for SimulMT and it is challenging to intuit which approach will perform best. For example, one could adapt LLMs fine-tuned for NMT (hereafter referred to as *NMT LLMs*) to SimulMT during inference, although how well such models will deal with the source context availabil-

[1] https://anonymous.4open.science/r/Simul-LLM-anonymous-E26F/

ity mismatch between fine-tuning (full sentence) and inference (partial sentence) is nebulous. Alternatively, one could fine-tune LLMs directly for SimulMT (hereafter referred to as *SimulMT LLMs*), but new prompt structuring is likely needed to match inference SimulMT behavior during fine-tuning exactly. Finally, it is also unclear how well previously understood concepts in existing SimulMT work, such as higher fine-tuning *wait-k* values increasing generalizability, will apply to SimulMT LLMs.

This paper seeks to address the above problems and contributes to the process of applying LLMs to SimulMT in the following major ways:

- We develop Simul-LLM, the first open-source fine-tuning and evaluation pipeline development framework for SimulMT LLMs, which seamlessly wraps around and interfaces with popular libraries for LLMs and SimulMT. This framework serves as a foundation for research on SimulMT LLMs that the community can employ and extend for a wide range of future work on LLM-based simultaneous translation.

- With the aforementioned framework, we explore the feasibility of adapting LLMs fine-tuned for NMT to SimulMT under a few decoding strategies and the classical *wait-k* fixed translation scheduler. Generally, we find that NMT LLMs demonstrate good performance during SimulMT inference which can be somewhat boosted by more complex decoding strategies.

- We propose an alternative prompt structuring approach to commonly employed NMT prompts that bridges the gap between the fine-tuning and inference environment, assuming a *wait-k* schedule, and we validate this via the Simul-LLM framework. We elaborate on counter-intuitive results that we observe and provide a base of exploration for future research to employ. Along these lines, we also validate that higher *wait-k* values employed during SimulMT fine-tuning do increase *wait-k* generalizability and boost translation quality across the board during SimulMT inference.

2 Background

While a range of work is relevant to this paper, we will only provide a focused and high level review of large language models applied towards machine translation and simultaneous translation as an application space. Readers interested in additional details should engage further with cited works in these areas.

2.1 Large Language Models for Neural Machine Translation

While LLMs are capable of effectively zero-shot sentence-to-sentence neural machine translation (NMT) (Vilar et al., 2023), their performance can still be improved via simple techniques. Prompt construction has been demonstrated to be critical to LLM performance, both before and after fine-tuning (Zhang et al., 2023). One-shot or few-shot performance via In-Context Learning (ICL) can produce near competitive results with fine-tuned LLMs for translation and can even be employed to enhance fine-tuned model performance (Vilar et al., 2023; Xu et al., 2023).

One particularly interesting area of study related to LLMs applied towards NMT remains whether or not to fully fine-tune a given model or engage in Parameter-Efficient Fine-Tuning (PEFT) (Mangrulkar et al., 2022). Early work in this area demonstrated some potential for smaller models (Üstün and Cooper Stickland, 2022), and the accessibility that PEFT provides designers in terms of fine-tuning on low-to-mid performance hardware setups renders it desirable. One of the most popular forms of PEFT freezes a LLM's weights and adds Low-Rank Adaptation (LoRA) (Hu et al., 2022) adapters between layers (other forms of PEFT exist, but adapter-based PEFT is extremely common so we refer to adapter-based PEFT simply as PEFT hereafter). While fully fine-tuned NMT LLMs tend to suffer from some level of catastrophic-forgetting (Kirkpatrick et al., 2017), intuitively, PEFT-based NMT LLMs should not suffer from any loss of off-task performance, as adapters can be loaded or detached depending on whether or not a given user is prompting for a translation. Given these factors, PEFT is an attractive option for NMT LLMs.

2.2 Simultaneous Translation

As a subset of typical NMT, simultaneous translation (SimulMT) focuses on engaging in translation (write decisions) while balancing the amount of available source context (read decisions) to reduce translation latency. This necessarily increases the difficulty of translating a sequence in one language to a sequence in another language, especially when

structural and/or syntactical differences exist in the language pair. As a brief example, we can consider translating from a subject-verb-object (SVO) language, such as English, to a subject-object-verb (SOV) language like German. Based on the available source context in English, we may have to guess at the translation in German without access to the necessary context to effectively make that prediction because of the aforementioned syntactical differences in the language pair.

There are two classical, high level approaches to scheduling the write and read decisions of SimulMT, those being static schedules like *wait-k* (Ma et al., 2019) or adaptive schedules which are flexible and learned, such as variants of monotonic multi-head attention, adaptive *wait-k* compositions, *wait-if-worse*, decision state assisted SMT, and others (Grissom II et al., 2014; Gu et al., 2017; Arivazhagan et al., 2019; Zheng et al., 2020). *Wait-k* remains a particularly popular baseline strategy given its ease of application during training and during inference. It functions by retaining a k-lagging factor between the source context (either in tokens or in words) $\mathbf{x}$ and the translation hypothesis $\mathbf{y}$. We can model a typical *wait-k* schedule's probability of generating a given output sequence, provided some source sequence, with Equation 1:

$$p(\mathbf{y}, \mathbf{x}) = \prod_{i=1}^{|\mathbf{y}|} p(\mathbf{y}_i | \mathbf{y}_{<i}, \mathbf{x}_{<min(i+k,|\mathbf{x}|)}) \quad (1)$$

Under circumstances or in environments where additional computational latency is acceptable, variations of beam search have been applied in simultaneous scenarios. Speculative Beam Search (SBS) (Zheng et al., 2019) is one such example where, at a high level, each translation step attempts to speculatively translate future steps for some number of beams, eventually selecting a single token or word (the first one) from the most likely beam for some beam length. When applied to *wait-k*, single token or word SBS can be modeled via Equation 2, where w is the length of the beam and $\hat{\mathbf{y}}_{i+1:w}$ represents the speculative beam of maximum joint probability:

$$p(\mathbf{y}, \mathbf{x}) = \prod_{i=1}^{|\mathbf{y}|} p(\mathbf{y}_i | \hat{\mathbf{y}}_{i+1:w}, \mathbf{y}_{<i}, \mathbf{x}_{<min(i+k,|\mathbf{x}|)}) \quad (2)$$

Chunk-wise variations are also possible, where for multiple consecutive translation steps, or write decisions, words or subwords from the last determined most-likely beam are employed to cut down on latency.

3 Simul-LLM: an Open-Source SimulMT LLM Fine-tuning Framework

SimulMT is an underexplored application space for LLMs, at the moment, and there is plenty of room to improve performance. To facilitate the rapid development of solutions for SimulMT LLMs (fine-tuned for SimulMT) or NMT LLMs (fine-tuned for NMT) adapted for SimulMT, we choose to develop and provide an open-source framework written in PyTorch for researchers to actively employ for future experiments. We call this framework *Simul-LLM*, and it bridges the gap between the development of fine-tuning agents via popular libraries and proper SimulMT evaluation. Simul-LLM is poised to support both selective classical SimulMT systems as well as fine-tuning and evaluation for a variety of LLM systems such as Falcon (Almazrouei et al., 2023), LLaMa (Touvron et al., 2023) and Mistral-based (Jiang et al., 2023) models. The high-level components of Simul-LLM include a fine-tuning wrapper and a SimulMT evaluation agent for every supported LLM; in the case of classical models, only the evaluation agent is supported in-framework.

3.1 Fine-tuning Wrapper and Features

The fine-tuning wrapper of Simul-LLM is constructed with a focus on simplicity and extensibility. It is depicted in Figure 1 and supports the following set of user-friendly features.

LLM Support and Extensibility: The proposed Simul-LLM currently supports Falcon and will imminently support Mistral and LLaMa for SimulMT fine-tuning. The fine-tuning wrapper is constructed with a focus on extensibility, allowing for rapid expansion to other LLMs assuming users hold a minimal level of LLM-specific knowledge.

Multiple Prompt Structures: Translation quality, as demonstrated by numerous prior works, varies significantly with prompt structure. As such, the fine-tuning wrapper of Simul-LLM supports multiple kinds of prompt structures, all of which we will validate later in this paper. In the interest of supporting adapting NMT LLMs to SimulMT, Simul-LLM supports NMT fine-tuning in addition to supporting prompt structures that allow for strict *wait-k* fine-tuning structures.

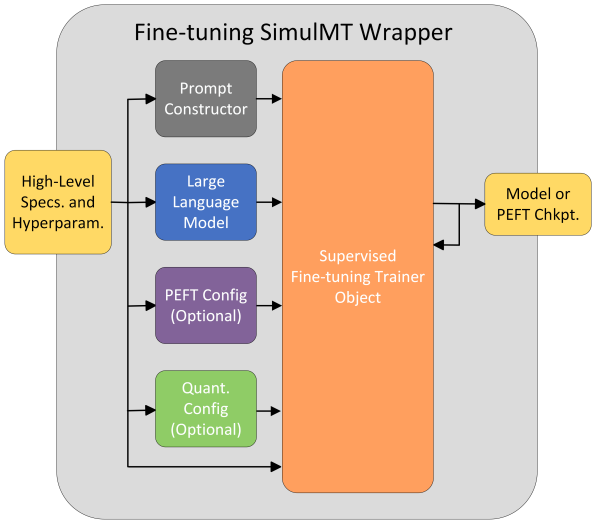

Figure 1: Depiction of the Simul-LLM fine-tuning wrapper framework. High level specifications and hyperparameters are passed to the wrapper on instantiation, which employs a specified prompt constructor, instantiates a specified LLM foundational model, optionally constructs a PEFT config, and optionally constructs a quantization config via `BitsAndBytes`.

Flexible Quantization and PEFT: Effectively fine-tuning LLMs is reliant on careful memory management for low-to-mid hardware setups. Given that, Simul-LLM quantizes LLMs via the `BitsAndBytes` library (seen in the Quant Config in Figure 1), which enables flexible fixed-point quantization. For most low-to-mid hardware setups, we recommend quantizing in 4-bit floating point via 4-bit NormalFloat (nf4). Additionally, Simul-LLM by default engages in PEFT (Mangrulkar et al., 2022) via a passed configuration, although it does also support full model fine-tuning.

Prompt Loss Filtering: Extremely basic supervised fine-tuning may inappropriately include portions of the prompt in loss calculations that are then backpropagated. The fine-tuning wrapper of Simul-LLM ensures that the model only learns from data it is intended to generate post-prompt via a `DataCollator` object and a response template.

3.2 Evaluation Agent and Features

Evaluation agents for Simul-LLM are similarly built for ease of use and extensibility in addition to customizability towards complex translation schedules and decoding strategies while seamlessly interfacing with the preeminent SimulMT evaluation framework, SimulEval (Ma et al., 2020). This is depicted in Figure 2 and includes the following features.

Classical SimulMT Translation Scheduler: In the interest of baseline accessibility, Simul-LLM evaluation agents support *wait-k* translation schedules for SimulMT, given their ease of application. Further adaptive or otherwise more involved translation schedulers can be quickly constructed and applied, assuming no reliance on fine-tuning.

Support for Multiple Decoding Strategies: Variable latency constraints for possible inference environments demand flexibility in decoding strategies. As such, evaluation agents for Simul-LLM support several decoding strategies, including greedy and naive decoding, subword-based beam search for single-word decoding, and variations on Speculative Beam Search (SBS) (Zheng et al., 2019), including single word-based and chunk-wise SBS.

Scoring and Latency via SimulEval: SimulEval (Ma et al., 2020) is the premier SimulMT evaluation framework. Simul-LLM evaluation agents interface seamlessly with SimulEval, which handles the incremental source context and manages translation scoring and latency tracking regarding the translation hypothesis.

4 Adapting NMT LLMs to SimulMT

Existing LLMs that have been fine-tuned for classical NMT may have the potential to be employed directly for SimulMT inference. This can be desirable under circumstances where a single deployed model is preferable to multiple, specialized models (e.g. avoiding fine-tuning costs for multiple models). However, exactly how to adapt such models is unclear in practice given the differences between prompts during NMT fine-tuning (full-sentence availability) and SimulMT inference (incremental source availability). This is especially problematic when engaging with short *wait-k* schemes and similar low-lagging schedules, where minimal source context is available during early translation steps.

To intuit why this is an issue, suppose that an NMT LLM is accustomed to receiving the entire source sequence $\mathbf{x}$ before outputting a word or token $\mathbf{y}_1$. If engaging in a low-lagging *wait-k* where $|\mathbf{x}| >> k$, then the output $\mathbf{y}_1$ can now only be based on source context up to $\mathbf{x}_k$. Assuming the NMT LLM would typically rely on some source context $\mathbf{x}_i$ where $i > k$, then critical information is missing from its translation decision. We have conducted preliminary quantitative studies on this front by employing the proposed Simul-LLM framework, and these results are presented in Section 7.

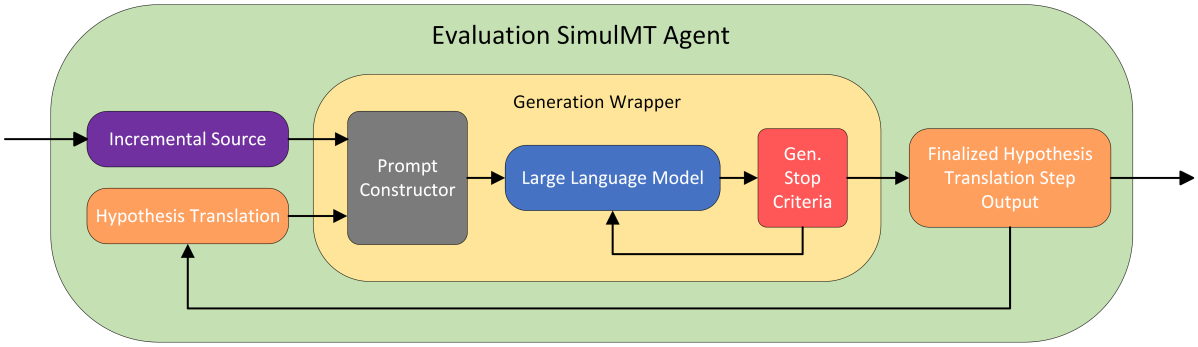

Figure 2: Depiction of the Simul-LLM evaluation agent framework. The SimulMT agent receives the incremental source from (left of the figure) and sends the finalized translation step hypothesis to SimulEval (right of the figure), which manages latency calculation and translation quality scoring.

5 Prompt Structure for SimulMT LLMs

Alternative to adapting NMT LLMs to SimulMT tasks, the SimulMT LLM approach aims to fine-tune LLMs directly for SimulMT, which requires new prompt structuring. Unlike classical encoder-decoder models for translation, source context for an LLM must be packaged within a prompt and structured appropriately. Some existing work has studied prompt structures for typical NMT (Zhang et al., 2023; Xu et al., 2023), but it is unclear whether such prompts are still optimal for SimulMT. This is because significant differences exist in source context availability between fine-tuning and inference, as discussed in Section 4.

To address this need, we propose a new prompting approach to construct the dataset for fine-tuning and evaluation of SimulMT LLMs. Instead of structuring each source-target sentence pair as a single example as typically done for NMT, we propose decomposing and expanding every sentence pair into a number of examples, where each example incrementally provides additional source and target tokens. When employed, the expanded examples directly mimic the behavior of simultaneous translation. We investigate two specific structures of materializing this prompt structuring approach, as analyzed and compared below.

5.1 Split Source-Target Prompt Structure

The first prompt structure follows the classical NMT prompts where the source sentence is included in the prompt and the target sentence is the output of the LLM. As illustrated in the first prompt structure in Figure 3, with the expanded examples, each example contains partial source along with some instruction in the prompt (starting with "<h>") and partial target that is k words behind the source in the model output (starting with "<a>"), where k is the intended inference *wait-k* value. While this *split source-target* prompt structure seems to be a natural and plausible way for model fine-tuning, it has exhibited difficulties when learning for simultaneous translation. The root of this stems from how fine-tuning with LLMs often works: when filtering the prompt from loss calculations during fine-tuning, a response template of some kind (e.g. "<a>:" or "»ANSWER«") is employed to ensure only the target word that is generated at the current step (i.e., $\mathbf{y}_i$) is scored. Unfortunately, with the split source-target prompt, the template allows for all the target words that have been generated from previous translation steps (i.e., $\mathbf{y}_1$ to $\mathbf{y}_i$) to be scored. Without employing a more complex loss filtering, this leads to an inappropriate level of context.

This lack of loss filtering can be especially problematic near the end of a given sequence's translation. As a simple example, suppose that a given source sequence is of length $|\mathbf{x}|$ and $|\mathbf{x}| >> k$. In the first prompt structure where up to i source words have been supplied and where $|\mathbf{x}| > i >> k$, the LLM is effectively being fine-tuned for varying *wait-k* values ranging from *wait-i* to *wait-k* in a single example (as $\mathbf{y}_i$ is predicted from $\mathbf{x}_1$ to $\mathbf{x}_{i+k}$ which is *wait-k*, $\mathbf{y}_{i-1}$ is predicted from $\mathbf{x}_1$ to $\mathbf{x}_{i+k}$ which is *wait-*$(k+1)$, and so so). If $i >> k$ to the point where it is close to normal NMT levels of context, where $i \cong |\mathbf{x}|$, then the LLM is no longer being effectively fine-tuned with an appropriate amount of source context for a given translation (write) decision schedule.

5.2 Single Output Word Prompt Structure

The above problem can be entirely side-stepped by our proposed second approach, *single output to-*

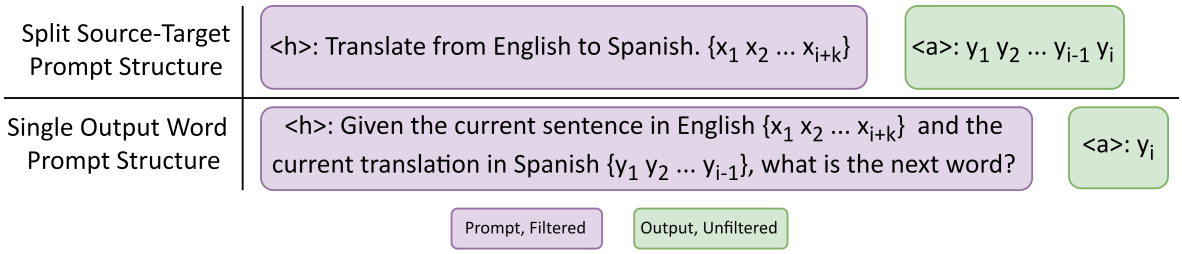

Figure 3: Example of English to Spanish translation prompt construction with an incremental source **x** and an incremental output **y** applied via our proposed expanded dataset. Without more complex loss filtering than is typical, the entire output sequence for the split source-target prompt structure would be scored and the model would learn for *wait-k* schedules ranging from *wait-i* to *wait-k* as opposed to just *wait-k*.

ken prompt structure, that embeds only the current target translation hypothesis within the model output. As illustrated in the second prompt structure in Figure 3, instead of allowing target translation hypotheses from previous time-steps to be incorporated into the loss, the proposed prompt structure shifts those previous translation hypotheses into the prompt. Combined with the expanded examples that form a rigorous *wait-k* curriculum in terms of the fine-tuning dataset (rigorous meaning a complete curriculum as opposed to a random subset), inference behavior can be copied exactly for every fine-tuning example, completely closing the context mismatch between fine-tuning and inference.

Between the two prompt structures, it is clear that the single output word prompt structure more closely replicates the relationship observed in Equation 1 between the source and target sequence during fine-tuning. For both structures, a source-target sentence pair is expanded up to $max(|\mathbf{x}| - (k - 1), |\mathbf{y}|)$ examples.

6 Evaluation Methodology

To validate our proposed solutions to the aforementioned challenges and to test the capabilities of Simul-LLM as the first SimulMT LLM open-source framework, we engaged in several experiments allowing for comparisons among classical non-LLM-based NMT (Vaswani et al., 2017) / SimulMT architectures (Ma et al., 2019), NMT LLMs adapted for SimulMT, and SimulMT LLMs. All mentioned LLMs are fine-tuned Falcon-7B models, but Simul-LLM features an easy to extend framework and we support both Llama and Mistral-based models as well (results in Appendix).

NMT LLMs were fine-tuned for one epoch, as overfitting was observed beyond that point, which we intuit to be possibly due to the well-documented ability of LLMs to quickly memorize training sets (Biderman et al., 2023). In contrast, SimulMT LLMs were fine-tuned for 2M random examples out of 5M examples on the expanded dataset due to computational constraints.

6.1 Dataset Selection and Preprocessing

No standardized dataset exists for SimulMT with LLMs. Due to its popularity in speech-to-text simultaneous translation (SimulST), we employ MuST-C for our experiments[2]. For the purposes of adapting MuST-C for text-to-text usage, we preprocess the dataset and filter out certain acoustic indicators (e.g., floating "-" characters representing pauses). In some cases, this resulted in significant changes to some samples of the test set, such as the removal of (Applause) acoustic indicators.

We employ MuST-C across two language pairs, those being English-to-German (en-de) and English-to-Spanish (en-es). Some additional experiments are provided for the en-es language pair that validate fundamental SimulMT concepts and display BLEU scores, gathered via sacreBLEU (Post, 2018), with respect to samples observed during fine-tuning. The original dataset contains roughly 270K training set samples and approximately 2.5-3K test set samples (tst-COMMON split) per language pair. The expanded version of this dataset for single output word fine-tuning contains approximately 5M training set samples per language pair.

6.2 Word or Token-Based *Wait-k* for LLMs

While classical encoder-decoder SimulMT systems usually engage in either word or token-based *wait-k*, they most typically engage with whichever is more suitable for their vocabulary (i.e. word versus sub-word vocabularies). In spite of the fact that LLMs function via sub-word vocabularies, we recommend, and employ for this work, word-based

[2]This allows for future work that explores multi-modal simultaneous LLMs, engaging in SimulST via a cascaded model structure with a transcription model for the source speech or a joint speech/text-to-text framework.

Grouped Explorations	Model and Decoding Scheme	en-de	en-es
Classical Baselines	NMT Transformer (non-simultaneous)	26.96 (22.6)	32.64 (23.1)
	Monotonic Transformer *Wait-5* (SimulMT)	22.01 (3.32)	24.90 (2.58)
NMT LLMs Adapted for SimulMT	NMT LLM	25.83 (3.65)	**30.06 (3.95)**
	NMT LLM Single SBS (k=3, b=5, c=1, w=6)	25.98 (4.12)	29.48 (4.64)
	NMT LLM Single SBS (k=3, b=5, c=1, w=10)	25.95 (4.25)	27.67 (4.82)
	NMT LLM Chunk SBS (k=3, b=5, c=2, w=10)	23.61 (4.63)	26.33 (5.26)
	NMT LLM Chunk SBS (k=5, b=5, c=3, w=15)	25.80 (5.61)	27.00 (5.90)
	NMT LLM Chunk SBS (k=7, b=5, c=4, w=20)	**27.32 (6.97)**	28.66 (7.09)
SimulMT LLMs with Proposed Prompt	*Wait-3* Fine-tuning LLM	19.99 (3.41)	23.68 (3.64)
	Wait-7 Fine-tuning LLM	20.82 (3.44)	25.18 (3.61)
	Wait-7 Fine-tuning LLM (k=7)	23.09 (6.71)	28.92 (6.87)

Table 1: Comparisons of peak performance for various models and decoding schemes during primarily *wait-3* evaluation (non-*wait-3* is specified via k) via detokenized BLEU. Non-LLM baselines are subword-based *wait-k* (standard) while LLMs are word-based *wait-k*. Best SimulMT quality results are **bolded**, second best results are underlined, and latency is provided in parentheses as LAAL (Papi et al., 2022). Speculative Beam Search (SBS) during inference is experimented with for NMT LLMs, which lend themselves towards SBS (k=*wait-k* value, b=beams, c=chunks/words, w=window size).

wait-k for SimulMT LLMs, as it more closely resembles the flow of engaging with a natural language interface. Moreover, supposing that the LLM is receiving a given sequence actively from a transcription system or something similar, it makes intuitive sense to wait for a word to be emitted from the system as opposed to a fragment.

7 Results and Analysis

7.1 Exploration of Adapting NMT LLMs to SimulMT

In Table 1, we provide a breakdown of the performance of several different models, decoding strategies, and *wait-k* schedules. Regarding our exploration related to adapting NMT LLMs to SimulMT, we also include results related to our implementation of Speculative Beam Search (SBS) (Zheng et al., 2019). As demonstrated by these results, compared with classical models, LLMs fine-tuned for NMT are very capable of SimulMT upon being adapted during inference (even exceeding the score of the classical NMT transformer on en-de that performs non-simultaneous translation). It is worth reiterating that classical architectures typically engage in subword-based *wait-k* whereas we employ word-based *wait-k* for LLMs, but the comparisons still serve as a useful reference.

SBS-based decoding strategies helped NMT LLMs in the en-de language pair, but lacked improvement for the en-es language pair. We noted that our implementation (and seemingly also the original implementation) was sensitive to both the window size and the number of committed chunks, with large values for either resulting in the speculative target translation getting too close to the size of the source context. In our tests, when reaching the same length as the source context (akin to context levels of *wait-1*), degenerate output began to appear that resulted in the output trailing off (e.g., final output of "que..." instead of correct output of "que"). Notably, too many committed chunks only explains performance gaps for chunk-wise SBS, not single SBS, which is normally a flat improvement upon greedy decoding (single SBS is still sensitive to window size). Future experiments can be conducted to utilize the proposed Simul-LLM framework to quantify these factors.

7.2 Exploration of SimulMT LLMs with Proposed Prompt Structure

In Table 1, we also provide a breakdown of the performance of our exploration of SimulMT LLMs with our proposed prompt structure in Section 5.2 that carefully manages source context availability for target translation generation. Two models are employed for this exploration, one fine-tuned for *wait-3* inference, and another fine-tuned for *wait-7* inference that produces noticeable better quality translations than the first.

While SimulMT LLMs are a more promising approach in achieving higher translation quality than NMT LLMs due to more direct task-specific fine-tuning and better context alignment, our ex-

Fine-tuning *Wait-k*	Inference *Wait-k*	BLEU
SimulMT LLMs Fine-tuned in *Wait-3*	*Wait-3*	23.68
	Wait-5	25.59
	Wait-7	26.31
SimulMT LLMs Fine-tuned in *Wait-7*	*Wait-3*	25.18
	Wait-5	28.19
	Wait-7	28.92

Table 2: Peak BLEU scores for various SimulMT LLMs fine-tuned with different *wait-k* values on en-es.

periment results suggest that, for the time-being, the performance of SimulMT LLMs is not advantageous to NMT LLMs adapted to SimulMT (and varies from slightly worse to barely better translation quality compared with the classical non-LLM SimulMT baseline). This is not completely unexpected as NMT LLMs have been optimized heavily in recent years whereas the exploration of SimulMT LLMs has just started. We provide some analysis below that point out several possible reasons for this observed performance gap and call for additional community efforts to investigate further.

First, due to computational constraints we were unable to fine-tune for an entire epoch of the training dataset (only 2M random samples out of 5M), which represents a major loss of lessons and a lack of rigorous *wait-k* curriculum for the fine-tuned model. Second, it is possible that the fine-tuning hyperparameters are ill-suited for this particular prompt. We consider this likely to be the most influential issue on our observed results, given the drastic differences between the original and expanded datasets. Third, at least one other work related to NMT LLMs (Chen et al., 2023) has demonstrated that relative positional embeddings can cause issues via attention dilution that ends up being unhelpful, suggesting that distancing the source context, running target hypothesis, and the current translation step hypothesis can be unexpectedly problematic. We posit that our proposed Simul-LLM can be leveraged to verify the above reasons.

7.3 Higher *Wait-k* Generalizability Comparisons

It is well documented that in typical SimulMT systems, training or fine-tuning with a slightly higher *wait-k* than intended during inference can boost translation quality and generalizability across slightly lower *wait-k* (Ma et al., 2019). While this likely applies to SimulMT LLMs, no existing work has validated that this behavior persists. We provide a brief comparison of two SimulMT LLMs fine-tuned via *wait-3* and *wait-7* context levels in Table 2. The results demonstrate that, generally, the expected behavior does hold, with all LLMs fine-tuned in *wait-7* outperforming their corresponding *wait-3* models for the same inference *wait-k* with up to a 2.6 BLEU improvement. We leave validating additional, previously understood SimulMT principles in SimulMT LLMs to future work.

7.4 Discussion on Other Works

We are aware of two other concurrent efforts at applying LLMs to simultaneous translation (Koshkin et al., 2024; Wang et al., 2023). The former focuses on high-resource settings (i.e. order-of-magnitude larger language models than we aim for in this work) and additionally touches on comparisons with GPT-4 and speech-to-text simultaneous translation efforts. Contrastingly, the latter work aligns with ours, approaching the problem of LLM simultaneous translation in a low-resource setting. Their work, however, is limited to a focused contribution: a variation of longest-common prefix (LCP) that they call relaxed agreement LCP (RALCP). Neither host open-source software to test their changes and both only test Llama-based models in addition to arguably shallower explorations of this subject matter. Nonetheless, we intend to add support for end-to-end speech-to-text translation soon and have already added support for RALCP, which we provide brief results for in our Appendix. Both works validate our choice to engage with MuST-C as a dataset, as both works also employ it.

8 Conclusion

In this work, we introduce Simul-LLM, the first open-source framework that enables rapid development of LLM fine-tuning and evaluation pipelines for simultaneous machine translation (SimulMT). Simul-LLM seamlessly integrates with the fine-tuning and generation tools of the popular `transformers` library as well as with SimulEval, the preeminent SimulMT evaluation framework. In addition to introducing Simul-LLM, we employ this framework to explore a wide range of topics in the LLM SimulMT space. Our proposed Simul-LLM framework enables multiple lines of future work that can be carried out to understand and optimize LLMs for simultaneous translation, and it will likely be a useful tool for the research community.

9 Limitations

This work is focused on providing an open-source framework for an interesting application of LLMs. While we propose a few novel techniques for engaging with LLMs in this application space that we support in Simul-LLM, the work is largely exploratory and we cannot assert that our proposed practices are guaranteed to work best. Additionally, the framework itself does not currently support speech-to-text exploration (although augmentation to this end should not be difficult), a popular simultaneous translation track. All of these points serve to slightly limit the impact of our contribution.

References

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. Falcon-40B: an open large language model with state-of-the-art performance.

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 *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1313–1323, Florence, Italy. Association for Computational Linguistics.

Stella Biderman, USVSN Sai Prashanth, Lintang Sutawika, Hailey Schoelkopf, Quentin Anthony, Shivanshu Purohit, and Edward Raff. 2023. Emergent and predictable memorization in large language models.

Yijie Chen, Yijin Liu, Fandong Meng, Yufeng Chen, Jinan Xu, and Jie Zhou. 2023. Improving translation faithfulness of large language models via augmenting instructions.

Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. 2014. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1342–1352, Doha, Qatar. Association for Computational Linguistics.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O. K. Li. 2017. Learning to translate in real-time with neural machine translation.

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 *International Conference on Learning Representations*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Roman Koshkin, Katsuhito Sudoh, and Satoshi Nakamura. 2024. Transllama: Llm-based simultaneous translation system.

M. Ma, L. Huang, H. Xiong, R. Zheng, K. Liu, B. Zheng, C. Zhang, Z. He, H. Liu, X. Li, H. Wu, and H. 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 (ACL).

Xutai Ma, Mohammad Javad Dousti, Changhan Wang, Jiatao Gu, and Juan Pino. 2020. Simuleval: An evaluation toolkit for simultaneous translation.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling.

Sara Papi, Marco Gaido, Matteo Negri, and Marco Turchi. 2022. Over-generation cannot be rewarded: Length-adaptive average lagging for simultaneous speech translation. In *Proceedings of the Third Workshop on Automatic Simultaneous Translation*. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

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. 2023. Llama: Open and efficient foundation language models.

Ahmet Üstün and Asa Cooper Stickland. 2022. When does parameter-efficient transfer learning work for machine translation? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7919–7933, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

David Vilar, Markus Freitag, Colin Cherry, Jiaming Luo, Viresh Ratnakar, and George Foster. 2023. Prompting PaLM for translation: Assessing strategies and performance. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15406–15427, Toronto, Canada. Association for Computational Linguistics.

Minghan Wang, Jinming Zhao, Thuy-Trang Vu, Fatemeh Shiri, Ehsan Shareghi, and Gholamreza Haffari. 2023. Simultaneous machine translation with large language models.

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.

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

Baigong Zheng, Kaibo Liu, Renjie Zheng, Mingbo Ma, Hairong Liu, and Liang Huang. 2020. Simultaneous translation policies: From fixed to adaptive.

Renjie Zheng, Mingbo Ma, Baigong Zheng, and Liang Huang. 2019. Speculative beam search for simultaneous translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1395–1402, Hong Kong, China. Association for Computational Linguistics.

A Appendix

A.1 Licensing Information

Fairseq (Ott et al., 2019) is MIT-licensed. SimulEval is licensed via CC BY-SA 4.0. MuST-C, one of the premiere speech-to-text datasets across many language pairs, is licensed under CC BY-NC-ND 4.0. Simul-LLM itself is MIT-licensed.

A.2 Training, Fine-Tuning, and Evaluation Hyperparameters and Hardware Details

All classical models were trained on two NVIDIA 32GB V100s and validated on a single V100. All LLMs were fine-tuned via PEFT (Mangrulkar et al., 2022) on a single NVIDIA 40GB A40 in bfloat16 and evaluated on a single V100 in float32. Simul-LLM seamlessly integrates with SimulEval (Ma et al., 2020) for the purpose of these evaluations. Classical transformer baselines were trained via Fairseq (Ott et al., 2019), an easily extensible sequence to sequence toolkit.

Classical models were trained with typical hyperparameters provided in Fairseq examples. All LLMs were fine-tuned with identical hyperparameters, employing a constant learning rate of 3e-4 and were optimized via Adam with around 4K warmup updates and batch sizes of 40 samples. LoRA adapter parameters were an α of 16 and a r value of 64, resulting in a total of around 40M added parameters during fine-tuning, with a dropout value of 0.1. For fair comparison, classical models were of a similar, although not quite identical, size in terms of parameter count.

Additionally, all LLMs were fine-tuned while quantized with NormalFloat4 (nf4) quantization. A small performance boost was observed when removing this quantization during inference, so all models did not engage with nf4 quantization during inference.

Decoding Scheme	en-de	en-es
Chunk SBS	25.8 (5.6)	27.0 (5.9)
Chunk SBS w/ RALCP	26.1 (5.3)	28.1 (5.5)

Table 3: Demonstration of RALCP support and effectiveness via BLEU and LAAL (provided in parentheses) with k=5, b=5, c=3, and w=15 as per Table 1. All models here employ the NMT LLM prompt structure mentioned in this paper.

A.3 Estimated Computational Costs

We estimate that, normalizing for 1 GPU, building Simul-LLM cost roughly 20 GPU days of experimentation and approximately 18 GPU days to generate the results in this paper.

A.4 RALCP and Policy Differences

While Simul-LLM supports RALCP (Wang et al., 2023), our application differs slightly from the specification of the original implementation. Unlike the authors of Wang et al. 2023, we do not engage in consecutive read decisions unless it is to maintain a k-lagging factor as specified by an inference *wait-k* value. This ensures, after a given set of outputted chunks via RALCP implemented on top of SBS (Zheng et al., 2020), that expected context

Foundational Model	BLEU (LAAL)
Falcon 7B LLM	30.06 (3.95)
Mistral 7B LLM	30.25 (3.66)
Llama-2 7B LLM	20.18 (3.84)[3]

Table 4: Brief comparison of Falcon, Mistral, and Llama-based models fine-tuned and evaluated in Simul-LLM. All models were fine-tuned with the NMT LLM prompt structure specified in this paper and are evaluated via a *wait-3* schedule with greedy decoding on en-es.

limitations are kept. This contrasts with the original implementation of RALCP on top of a more relaxed read-write schedule that does not always attempt to maintain a k-lagging factor after that lagging factor is initially established. Additionally, we assume that the authors applied a word-based simultaneous approach, as this is not specified in their original work.

Nonetheless, RALCP, which is fundamentally a form of beam rescoring, does improve translation quality based on our empirical results. We include a brief comparison in Table 3, where we observe that the addition of RALCP seems to result in small to moderate improvements in BLEU score with roughly equivalent latency behavior (slight reductions in our tests).

A.5 Supporting Various LLMs

Simul-LLM's easy to extend fine-tuning and evaluation framework supports Falcon, Llama, and Mistral-based models. While our explorations in this paper are rooted in Falcon-based models, we provide a brief set of results for other LLMs in Table 4. Generally, translation quality appeared similar across models, although the Llama-based model struggled. BLEU degradation for Llama models seemed mostly linked to degenerate output where the model would get stuck in low-quality generation (e.g. continuously outputting the same token).

[3] Some issues were observed during fine-tuning the Llama-based translation agent that seems to have resulted in poorer than expected translation quality. We are actively working to update these results.