**: A Framework for Exploring High-Quality Simultaneous**

**Translation with Large Language Models**

一个基于大语言模型探索实现高质量同时翻译的框架

背景

要讨论实时翻译，首先要从最经典的模型说起，

**神经机器翻译（NMT），同时也是** LLMs的一个热门下游任务，

特点：是文本到文本的模型，在翻译开始时可以获得完整的源上下文，因此他的翻译质量是很高的，但是他只能在得到全部信息后翻译，会消耗较多的时间

而今天我们要讨论的是同时翻译（SimulMT），他是NMT的一个特殊子集任务，他们两个的任务不同：传统的翻译任务通常是**后置翻译**，等到源语言的整句话输入完毕后，才开始生成目标语言的翻译。**但在实时场景下，比如会议翻译或直播翻译，延迟显然是无法接受的**。模型需要在尚未获得整个源序列时就开始翻译，也就是simulmt。

在同时翻译中，模型需要在源语言句子还未完全输入时，逐步生成目标语言的翻译。

这意味着我们必须面对一个关键的挑战：**在有限的上下文信息下生成高质量的翻译**。

而随着**大型语言模型（LLM）** 的迅速发展，我们可以看到在很多自然语言处理任务中看到了它们的强大能力（推理，翻译，回答问题），带来了新的可能性。

在这之前的研究者们提出了一种叫做**Wait-k策略**的方法，它允许模型在接收到源语言的前k个单词后就开始翻译。减少了翻译的延迟，但也带来了上下文不足的问题，特别是在源语言句子较长时，前k个单词往往不足以提供足够的信息。

那么，针对这些问题，**本篇论文的核心就是探讨如何通过大型语言模型（LLM）和Wait-k策略的结合，优化同时翻译的质量**。

**问题定义与挑战**

**同时翻译**的最大挑战就是在**低延迟**和**翻译质量**之间找到平衡。理想情况下，我们希望机器能够在源语言输入的过程中快速翻译目标语言，这样才能保证实时性。然而，当模型接收的源语言信息还不完整时，它往往无法做出准确的翻译决策。**这就引出了上下文不足的问题**。

**之前我们说了Wait-k策略** 是解决低延迟翻译。

然而，这种策略存在一个关键问题：当源语言句子较长，或者 k 值较小时，模型可能缺乏足够的上下文信息。比如，源语言中的某些重要信息可能位于句子后半部分，但由于模型在生成第一个目标词时只看到了前 k个单词，这些关键信息没有被纳入考虑。

另一个问题是**上下文不匹配**。这个问题体现在训练（微调）阶段与推理阶段的不一致性。模型在微调时往往会看到完整的源语言句子，但在推理时，由于Wait-k策略的限制，它只能看到源语言的一部分。这种上下文的不匹配会让模型在推理阶段难以适应，就会影响翻译质量。

所以论文提出的一个核心问题是：**如何减少或消除训练阶段和推理阶段的上下文不匹配**，让模型即使在不完整的上下文下，也能生成准确的翻译。

其实也就是，SimulMT模型必须根据现有的上下文和以往的经验推测最可能的下一个翻译词，而这正是LLMs所擅长的任务。

**我们可以通过微调**来减少训练和推理之间的上下文不匹配。

它们具备强大的生成能力，可以应对翻译过程中逐步生成目标语言的需求。

在了解实时翻译的任务和面临的问题之后，接下来我们将讨论论文中提出的**Simul-LLM框架**，它是如何通过设计有效的提示（prompts）和微调策略，来解决这些问题并提升翻译质量的。

**Simul-LLM框架：技术设计与实现**

在前面我们讨论了**Wait-k策略**的上下文不足问题和**上下文不匹配**的问题，

本文的目标是通过结合**大型语言模型（LLM）** 和 **Wait-k策略**，优化同时翻译的质量和效率。核心思想是通过**微调和高效的提示结构**，**调整模型对不完整上下文的适应能力**，确保在低延迟情况下保持较高的翻译质量。

**主要由三个核心模块构成：**

* **提示构造器（Prompt Constructor）**
* **大规模语言模型（LLM）**
* **微调与推理机制**

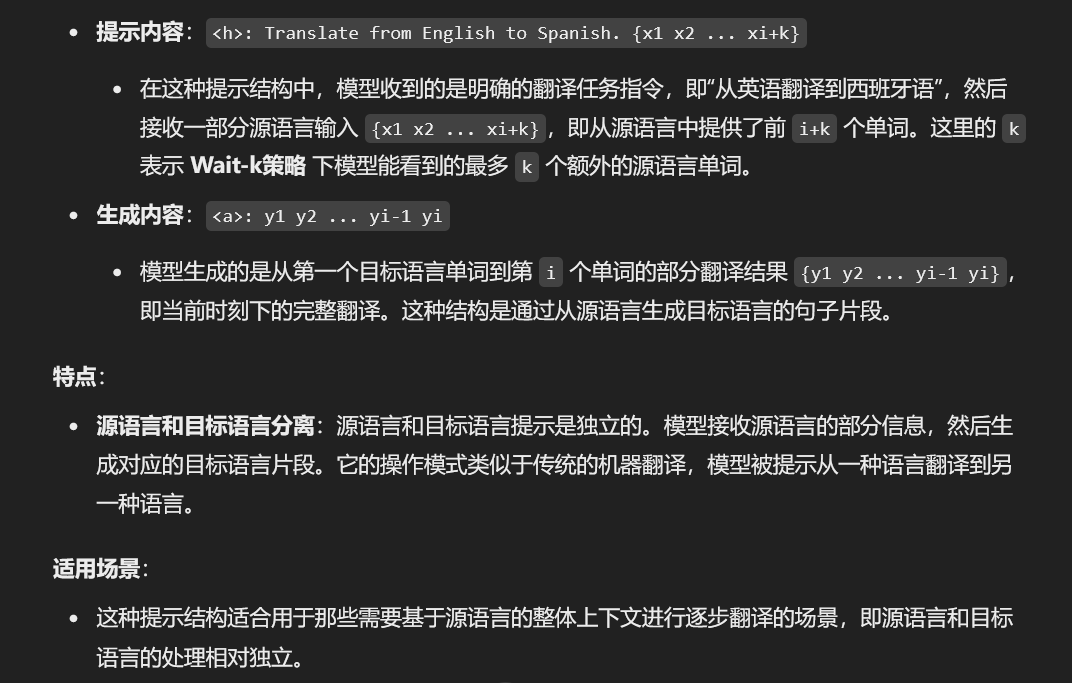
接下来我们逐个模块进行解析。

**提示构造器（Prompt Constructor）**

作用是为大规模语言模型提供有效的提示（prompts），以帮助模型生成准确的翻译。提示可以理解为模型接收的“上下文信息”，让模型知道如何从源语言生成目标语言。

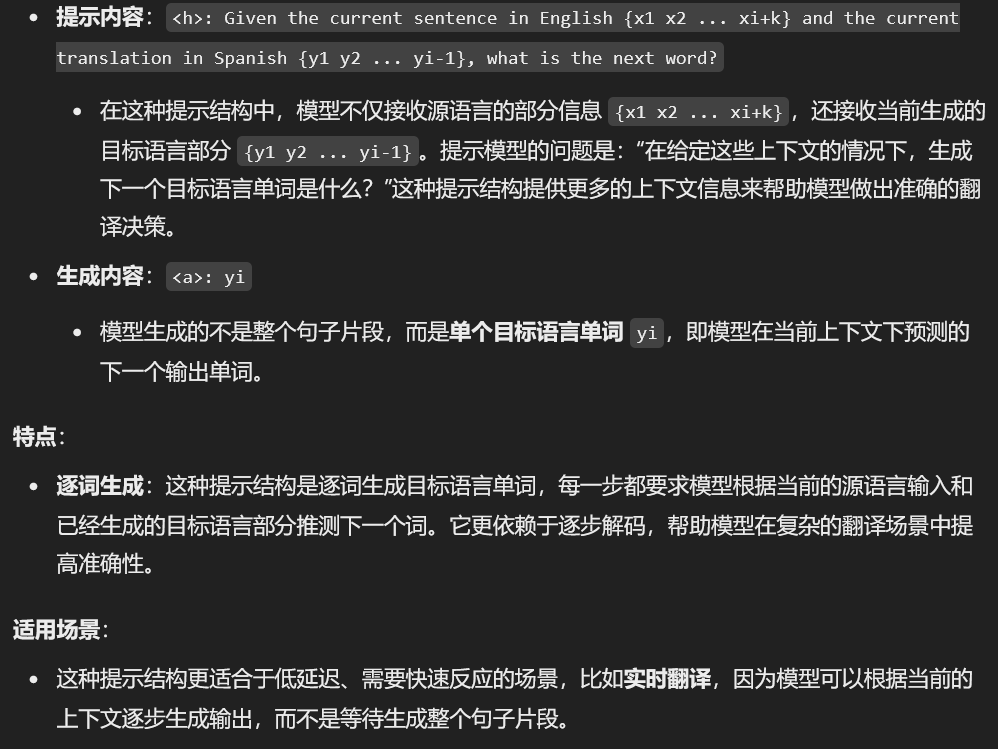
**两种提示结构**：

* **Split Source-Target Prompt Structure分割源-目标提示结构**：这种提示结构分别提供源语言和目标语言的片段，模型根据提示生成目标语言的翻译片段。提供x1-k,其中多出来的是wait-k策略下的额外原单词，直接翻译到目标语言的y1-i



操作模式类似于传统的机器翻译，模型被提示从一种语言翻译到另一种语言。基于源语言的整体上下文，源语言和目标语言的处理相对独立。

* **Single Output Word Prompt Structure单输出词提示结构**：更细粒度的提示结构，为模型提供源语言的上下文以及当前的部分翻译结果，并要求模型生成下一个单词。除了接受源语言的x1-k,同时接收当前生成的目标语言部分 {y1 y2 ... yi-1}提示模型的问题是：“在给定这些上下文的情况下，生成下一个目标语言单词是什么？” 生成的不是整个句子片段，而是**单个目标语言单词** yi



逐词生成目标语言单词，每一步都要求模型根据当前的源语言输入和已经生成的目标语言部分推测下一个词。它更依赖于逐步解码。更适合于低延迟、需要快速反应的场景，比如**实时翻译**，因为模型可以根据当前的上下文逐步生成输出，而不是等待生成整个句子片段。

通过上面讲解

**为什么提示构造器重要？**确保了模型在微调和推理时能接收到与推理阶段一致的上下文信息。这就解决了**上下文不匹配**问题。

**大规模语言模型（LLM）**

在提示构造器生成提示之后，LLM负责根据提示构造器生成的上下文提示，逐步翻译源语言。

Simul-LLM框架通过**参数高效微调（PEFT, Parameter-Efficient Fine-Tuning）**

来减少微调过程中的计算开销。在传统的微调方法中，模型的所有参数都会被更新，而PEFT只微调少量关键参数，极大减少了硬件资源的需求。这对于同时翻译任务中的实时性非常关键，因为我们不能承受耗费大量计算资源来等待模型更新。

设计一个**严格的Wait-k微调课程**，解决了微调和推理时的上下文不匹配问题。

**Wait-k微调课程**：

* **扩展的训练数据**：通过扩展微调数据集，让模型在微调时接触到与推理阶段相同的上下文输入。这意味着，模型在训练时不仅仅处理完整的源语言上下文，还会学习在源语言输入部分时如何生成高质量的翻译，从而在推理时表现更好。
* **严格课程设计**：这部分强调了扩展训练数据集的重要性。通过设计一个覆盖广泛的微调课程，模型在微调时可以见到各种上下文输入的情况，完全匹配推理时的实际场景。这样就避免了微调和推理时的上下文差异问题。

微调封装器框架的示意图

这张图展示了如何在**SimulMT**任务中使用大规模语言模型进行微调的完整过程。输入的高层次规格和超参数定义了整个过程的结构，提示构造器为模型提供任务提示，解决了**上下文不匹配，**PEFT(参数高效微调)和量化配置可以选择性地应用以降低计算开销。

整个微调过程由监督微调训练器对象管理，最终输出经过微调的模型或参数检查点。

**实验设计与结果**

接下来，我们会继续讨论论文的实验设计与结果，看看Simul-LLM框架在不同任务和数据集上的表现如何。

论文在多个同时翻译的实验设置下对Simul-LLM框架进行了测试

**1. 数据集：**

* 论文中使用了**MuST-C数据集**，这是一个常用于语音到文本翻译的大规模数据集，包含多种语言对。实验中的翻译任务主要集中在**英语到德语**和**英语到西班牙语**这两个语言对。

2. **对比方法**：

 **传统NMT模型**：完全依赖于源语言的完整输入进行翻译，作为基准。

 **Wait-k策略下的LLM微调**：使用不同的k值来测试翻译质量和延迟的平衡。

 **推测波束搜索（SBS）**：一种加速翻译过程的策略，通过提前推测未来目标词，减少生成时间。

**3.评估指标：**

* **BLEU分数**：这是评估翻译质量的标准指标，BLEU分数越高，表示翻译的准确性越好。
* **延迟（lagging）**：延迟越低，意味着翻译速度越快，适合实时场景。

**结果分析**

**2. 表格1：Simul-LLM与其他方法的对比**

传统NMT模型和推测波束搜索SBS在多个翻译任务中的性能对比。展示了不同模型、解码策略和wait-k调度的性能对比

**表格说明**：

* **Simul-LLM在BLEU分数上的表现**：在多数情况下，Simul-LLM的翻译质量接近或优于传统NMT模型，特别是在延迟控制得较好的情况下（在英语-德语对上超过了执行非同时翻译的经典NMT Transformer的得分），Simul-LLM的BLEU分数几乎与传统后置翻译方法相当。这说明框架能够在**不牺牲太多翻译质量**的前提下，显著降低延迟。

还展示了使用新的提示结构进行SimulMT LLMs探索的性能结果。使用了两个模型，一个是为wait-3推理微调的模型，另一个是为wait-7推理微调的模型，后者在翻译质量上明显优于前者。

**这很好理解**

**1. 表格2：不同k值下的Wait-k策略对比**

**表格说明**：

* 当k值较小时，模型可以更快开始翻译，**延迟显著降低**，但是随着k值减小，**BLEU分数**也有所下降。这是因为上下文不足，模型无法获得足够的信息进行准确的翻译。
* 当k值较大时，**翻译质量提高**，BLEU分数上升，但延迟也随之增加。这说明在Wait-k策略下，**k值的选择是翻译质量和延迟之间的一个重要平衡点**。

对比了两个SimulMT LLMs的简单比较，一个是在wait-3上下文下微调的，另一个是在wait-7上下文下微调的。结果显示，通常情况下，预期的行为确实存在，使用wait-7微调的所有LLMs在相同的推理wait-k情况下，性能均优于相应的wait-3模型

因此

**3. 上下文不匹配问题的解决效果**

论文的一个重要实验还展示了通过设计**严格的Wait-k微调课程**，Simul-LLM框架如何解决上下文不匹配的问题。

**实验结果展示**：

* 微调过程中使用了扩展的Wait-k训练集，确保模型在训练时已经学会处理不完整的上下文信息。在推理阶段，实验结果表明，与未使用严格微调课程的模型相比，使用该策略的Simul-LLM在推理时表现更加稳定，BLEU分数更高。
* 这部分实验结果证明了通过严格的微调课程，Simul-LLM框架能够在上下文不完整的情况下有效应对，同时消除了训练和推理阶段的上下文差异。

相关工作

创新点：

**1. 大型语言模型（LLM）与Wait-k策略的结合**

Wait-k策略主要是在传统的神经机器翻译（NMT）模型上实现的，论文首次探索了如何在LLM架构中应用Wait-k策略。借助llm擅长根据上下文和经验推理的特点，从而在提高翻译质量的同时，依然能够显著降低翻译延迟。

2. 设计一个**严格的Wait-k微调课程**，来解决训练和推理阶段的上下文不匹配问题。

3. 采用了**PEFT技术**，只微调部分模型参数，减少了微调时的计算成本。高效又灵活的微调方法，非常适合实时翻译任务。

优势：

1. 通过LLM提高了翻译质量，还在**低延迟**下保持了高水平的翻译准确性。通过调整Wait-k的k值，Simul-LLM框架能够在不同的场景中灵活地控制延迟和翻译质量的平衡。
2. 严格的微调课程设计，Simul-LLM框架在训练时模拟了推理时的上下文输入，解决上下文不匹配问题
3. LLM作为核心模型，这个框架不仅适用于当前的翻译任务，未来也可以扩展到其他类型的自然语言处理任务。无缝适配和微调现有的LLMs，如Falcon、Llama、Mistral等，并提供了参数高效微调（PEFT）的选项，降低了对硬件资源的要求，使得低性能设备也能够参与模型的微调和应用。

不足

**跨语言对的扩展性不足**：当前的实验主要集中在**英语到德语**和**英语到西班牙语**的翻译任务上。不同语言之间的语法结构差异可能会对LLM的翻译质量产生更大的影响。

**解码策略复杂度**：推测波束搜索（SBS）等复杂解码策略尽管在某些情况下提升了翻译质量，但在不同语言对（如英语到西班牙语）上表现不一致，策略对具体实现和参数选择较为敏感。

**计算资源的限制**：采用了参数高效微调技术，但使用LLM进行同时翻译的训练过程仍然对计算资源要求较高。特别是在实际部署时，如何在资源有限的环境中高效应用该框架是一个待解决的问题。

**MuST-C** 数据集最初是为**语音到文本**的同时翻译（SimulST）任务设计的，但论文将其修改为适用于**文本到文本**翻译任务。但在某些细节上可能仍与原本设计的语音翻译任务有一定的差异。

**上下文不匹配问题**：在低等待-k策略下，源语言输入不完整时会影响翻译准确性。这同样也是研究者们设计严格的WAIT-k策略的原因