

VIP Cheatsheet:

Transformer 与大语言模型

Afshine AMIDI 与 Shervine AMIDI 著
翻译: Tao Yu (俞涛)、Binbin Xiong (熊斌斌)

2025 年 4 月 12 日

本速查表概述了《Super Study Guide: Transformer 与大语言模型》一书的核心内容，原书长达约 250 页，包含约 600 幅插图，深入讲解了以下核心概念。
更多详情请访问: <https://superstudy.guide>。

1 基础知识

1.1 词元

定义 – 词元 (token) 是文本中的最小不可分割单元，如单词、子词或字符，属于预定义词表中的元素。

注: [UNK] 表示未知文本片段，[PAD] 用于补齐输入，使序列长度保持一致。

分词器 – 分词器 (tokenizer) T 将文本按不同粒度切分为若干 token。

这只泰迪熊真是太太太可爱了 \rightarrow T \rightarrow [这只][泰迪][熊][真是][UNK][可爱了][PAD] ... [PAD]

主要分词器类型如下:

类型	优势	劣势	示意图
词	<ul style="list-style-type: none">易于理解序列较短	<ul style="list-style-type: none">词表庞大无法处理词形变化	[泰迪][熊]
子词	<ul style="list-style-type: none">利用词根嵌入直观	<ul style="list-style-type: none">序列变长分词复杂	[泰][##迪][熊]
字符 字节	<ul style="list-style-type: none">无 OOV 问题词表小	<ul style="list-style-type: none">序列极长因为粒度层级过低，难以理解其中的模式	[泰][迪][熊]

注: 字词对编码 (Byte-Pair Encoding, BPE) 与 Unigram 是常用的子词级分词方法。

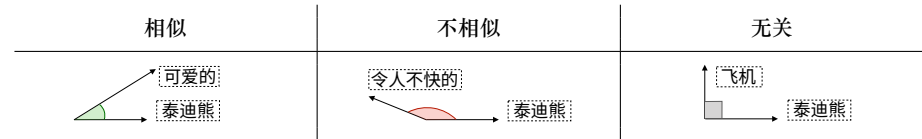
1.2 嵌入表示

定义 – 嵌入是元素 (如词元或句子) 的向量化表示，形式表示为向量 $\text{vector } x \in \mathbb{R}^n$ 。

相似度 – 两个词元 t_1, t_2 的余弦相似度定义为:

$$\text{相似度}(t_1, t_2) = \frac{t_1 \cdot t_2}{\|t_1\| \|t_2\|} = \cos(\theta) \in [-1, 1]$$

角度 θ 描述了两个词元 token 之间的相似程度:

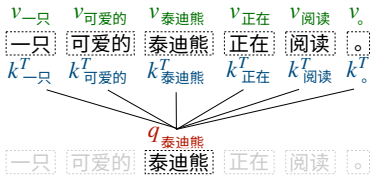


注: 近似最近邻 (Approximate Nearest Neighbors, ANN) 和局部敏感哈希 (Locality Sensitive Hashing, LSH) 是两种常用的方法，可在大规模数据库中高效近似计算相似度。

2 Transformer

2.1 注意力机制

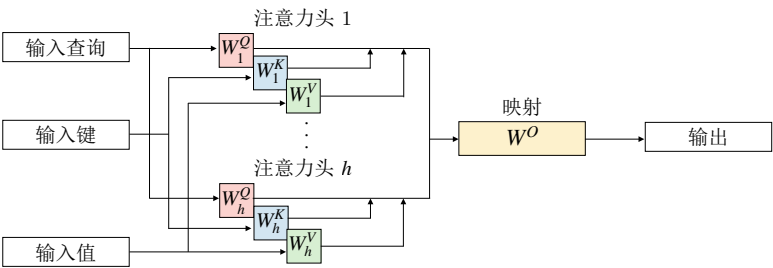
公式 – 给定一个查询向量 q ，我们希望知道它应该对哪个键 k 给予“注意”，从而获取与之相关联的值 v 。



注意力可以通过包含查询、键和值的矩阵 Q, K, V 高效计算，同时结合键的维度 d_k ，使用如下公式:

$$\text{注意力} = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

多头注意力 – 多头注意力 (Multi-Head Attention, MHA) 层会在多个注意力头 (head) 上并行执行注意力计算，然后将结果映射到输出空间。

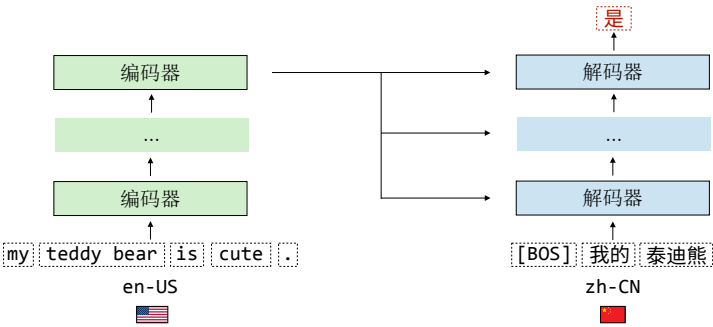


每个注意力头使用权重矩阵 W^Q, W^K, W^V 将输入映射为对应的查询 Q 、键 K 、值 V ，然后通过一个输出映射矩阵 W^O 生成最终输出。

注：分组查询注意力机制（*Grouped-Query Attention, GQA*）和多查询注意力机制（*Multi-Query Attention, MQA*）是 *MHA* 的变体，通过在多个注意力头之间共享键和值来降低计算开销。

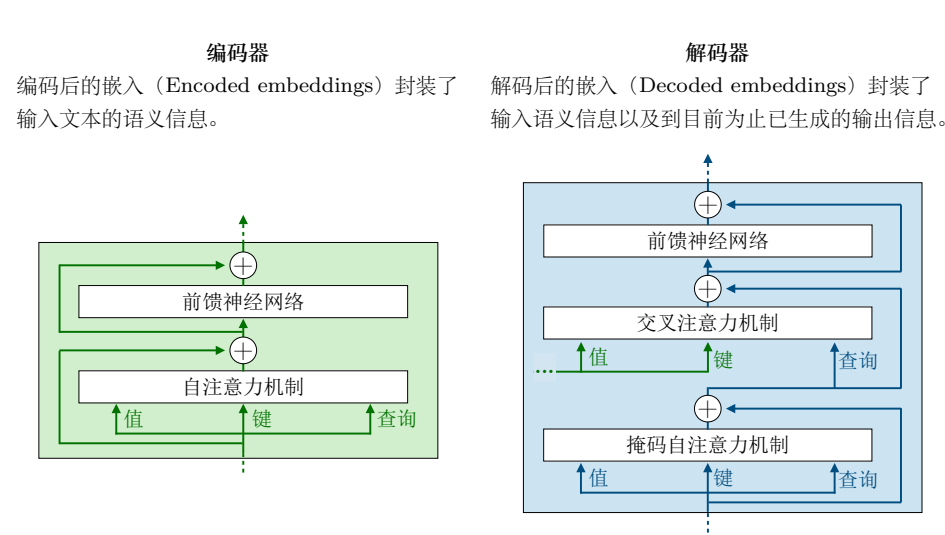
2.2 架构

总览 – Transformer 是一种具有里程碑意义的模型，基于自注意力机制（self-attention）构建，其核心结构由编码器（Encoder）和解码器（Decoder）组成。编码器负责对输入进行语义编码，生成有意义的嵌入表示，解码器则利用这些嵌入预测序列中的下一个词元。



注：尽管 *Transformer* 最初是为机器翻译任务提出的，但如今已广泛应用于多种自然语言处理任务中。

模块组成 – Transformer 的编码器和解码器是其两个基本组件，且各自承担不同的功能：

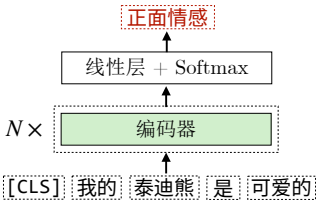


位置编码 – 位置编码（*Position Embeddings, PE*）用于告知模型每个次元在句子中的具体位置，其维度与次元的嵌入向量维度一致。位置编码可以是预定义的（固定位置编码），也可以是通过训练自动学习的（可学习位置编码）。

注：旋转位置编码（*Rotary Position Embeddings, RoPE*）是一种常用且高效的位置编码方式。它通过旋转查询向量和键向量的方式，将相对位置信息融入注意力计算中，提升了模型在处理长距离依赖上的能力。

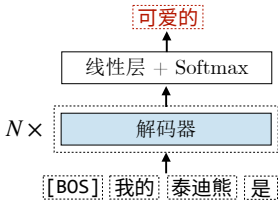
2.3 Transformer 变体

仅编码器结构 – BERT（*Bidirectional Encoder Representations from Transformers*，双向编码器表示）是一种基于 Transformer 的模型，由多个编码器堆叠而成。它以文本为输入，输出富有语义的嵌入向量，可用于下游的分类等任务。



在输入序列开头添加特殊的 [CLS] 词元，用于表示整个句子的语义。该词元的编码结果常用于下游任务，例如情感分析。

仅解码器结构 – GPT（*Generative Pre-trained Transformer*，生成式预训练 Transformer）是一种自回归的 Transformer 模型，由多个解码器堆叠组成。与 BERT 及其衍生模型不同，GPT 将所有问题统一视为“文本到文本”的生成问题。



目前多数主流的大语言模型（LLMs）都采用仅解码器架构，例如 GPT 系列、LLaMA、Mistral、Gemma、DeepSeek 等。

注：编码器-解码器模型（如 *T5*）同样具备自回归能力，并在结构和行为上与仅解码器模型有许多相似之处。

2.4 优化方法

注意力机制近似 – 注意力计算的时间复杂度为 $O(n^2)$ ，当序列长度 n 增加时，计算成本也迅速上升。常见的近似方法包括：

- 稀疏注意力：注意力不在整个序列中进行，而只在更相关的 token 之间进行计算。



- 低秩近似：将注意力公式近似为低秩矩阵的乘积，从而显著降低计算负担。

□ **Flash Attention** – Flash Attention 是一种精确的注意力计算优化方法。它通过充分利用 GPU 硬件，在快速的静态随机存取存储器 (*Static Random-Access Memory*, SRAM) 中执行矩阵操作，再将结果写入较慢的高带宽内存 (*High Bandwidth Memory*, HBM)，从而实现更高效的计算。

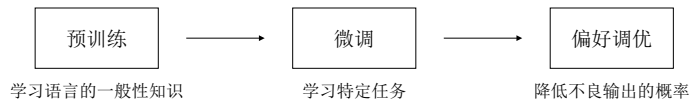
注：在实际应用中，Flash Attention 可有效减少内存占用并显著加速注意力计算过程。

3 大语言模型

3.1 概览

□ **定义** – 大语言模型 (*Large Language Model*, LLM) 是一类基于 Transformer 架构的模型，具备强大的自然语言处理能力。所谓“大”，是指它们通常包含数十亿甚至数千亿个参数。

□ **生命周期** – LLM 的训练通常分为三个阶段：预训练 (pretraining)、微调 (finetuning)、偏好调优 (preference tuning)。

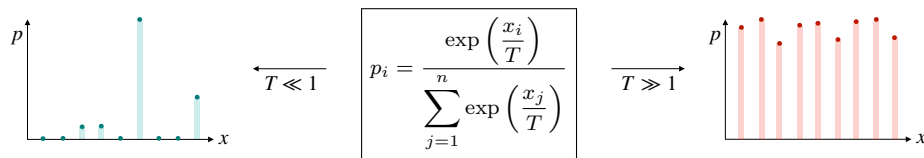


其中，微调与偏好调优属于后训练 (post-training) 阶段，目的是进一步对齐模型行为以完成目标任务。

3.2 提示工程

□ **上下文长度** – 上下文长度 (context length) 指模型在输入中能处理的最大次元数量，通常从几万到数百万不等。

□ **解码采样** – 生成次元时，从预测的概率分布 p_i 中进行采样，采样过程受超参数温度 (temperature) T 控制。



注：高温 (度 T 较大)：输出更具创造性，更多样化；低温 (度 T 较小)：输出更确定，更可控

□ **思维链** – 思维链 (*Chain-of-Thought*, CoT) 是一种推理过程，模型将复杂问题分解为一系列中间步骤，从而生成更准确的最终答案。思维树 (*Tree of Thoughts*, ToT) 是 CoT 的扩展版本，支持更复杂的推理结构。

注：自洽性 (*self-consistency*) 是一种方法，通过聚合多个 CoT 推理路径的结果来提升答案的稳定性和准确性。

3.3 微调

□ **监督式微调** – *Supervised FineTuning* (SFT) 是一种后训练方法，用于使模型行为与特定任务对齐，依赖于高质量的输入-输出对作为监督信号。

注：若训练数据是指令格式，该过程也称为指令微调 (*instruction tuning*)。

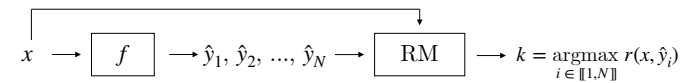
□ **参数高效微调** – *Parameter-Efficient FineTuning* (PEFT) 是一类用于高效执行 SFT 的方法。其中，低秩适配 (LoRA) 通过以下方式近似可学习权重矩阵 W ，其中， W_0 是固定的原始权重，仅需训练低秩矩阵 A 、 B 。

$$d \times k \text{ matrix } W \approx d \times k \text{ matrix } W_0 + d \times r \text{ matrix } B \times r \times k \text{ matrix } A$$

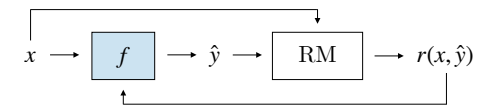
注：其他参数高效微调技术还包括前缀微调 (*prefix tuning*) 和适配器层插入 (*adapter layer insertion*)。

3.4 偏好调优

□ **奖励模型** – 奖励模型 (*Reward Model*, RM) 是一种模型，用于预测在给定输入 x 的情况下，输出 \hat{y} 与期望行为之间的契合程度。*Best-of-N* (BoN) 采样，又称拒绝采样 (*rejection sampling*)，是一种基于奖励模型的方法，在模型生成的 N 个候选输出中，使用奖励模型选择得分最高的那个作为最终输出。



□ **强化学习** – 强化学习 (*Reinforcement Learning*, RL) 使用奖励模型引导模型 f 的更新，以提高生成输出的质量。若奖励信号来源于人类反馈，该过程称为：基于人类反馈的强化学习 (*Reinforcement Learning from Human Feedback*, RLHF)。

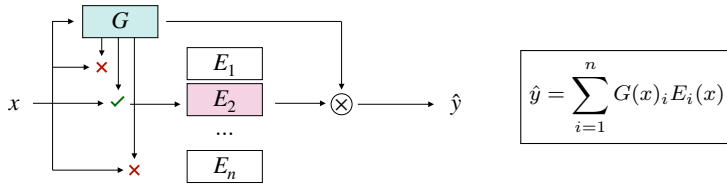


其中，*Proximal Policy Optimization* (PPO) 是一种常用算法，既鼓励高奖励输出，又限制模型偏离原始参数，避免奖励投机 (reward hacking)。

注：直接偏好优化 (*Direct Preference Optimization*, DPO)，它将奖励建模和强化学习结合为一个统一的监督训练步骤。

3.5 模型优化

□ **专家混合** – 专家混合 (*Mixture of Experts*, MoE) 模型是一种在推理阶段仅激活部分神经元的模型结构。其核心机制由一个门控网络 (gate) G 和多个专家网络 (experts) E_1, \dots, E_n 组成。



MoE 模型常在其前馈神经网络 (FFNN) 中使用这种门控机制，以在保持性能的同时降低推理计算量。

注：MoE 模型虽然在推理阶段高效，但训练过程极具挑战。如 *LLaMA* 论文所述，其作者因训练难度未采用该架构。

□ **蒸馏** – 模型蒸馏 (distillation) 是一种模型压缩技术，通过将大型“教师模型” (teacher) 预测的输出，用于训练一个小型的“学生模型” (student)。训练时使用的是 KL 散度损失 (Kullback-Leibler Divergence)：

$$\text{KL}(\hat{y}_T || \hat{y}_S) = \sum_i \hat{y}_T^{(i)} \log \left(\frac{\hat{y}_T^{(i)}}{\hat{y}_S^{(i)}} \right)$$

注：教师模型的输出称为软标签 (soft labels)，表示每个类别的概率，而非单一标签。

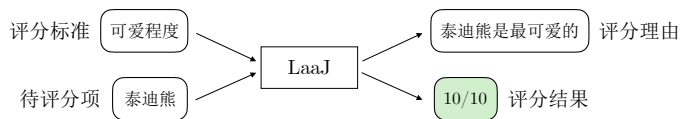
□ **量化** – 模型量化 (quantization) 是一类将模型参数的数值精度降低的技术（如从 FP32 降为 INT8 或 BF16），在尽可能保留模型性能的前提下，显著减小模型体积、降低内存使用、并加速推理速度。

注：QLoRA 是一种常见的 LoRA 量化变体。

4 应用

4.1 LLM 评审器

□ **定义** – LLM 评审 (*LLM-as-a-Judge*, LaaJ) 是一种评估方法，利用大语言模型 (LLM) 根据给定的标准对输出内容进行评分。值得注意的是，它不仅能打分，还能生成评分理由，从而提高评估的可解释性。



与传统的自动评估指标（如 *Recall-Oriented Understudy for Gisting Evaluation*, ROUGE）不同，LLM 评审不依赖参考文本，因此能被广泛应用于各种任务的评估。尤其在使用如 GPT-4 这类强大的大模型时，LLM 在评分结果上与人类评分高度相关，因为它具备推理能力。

注：LLM 非常适合快速迭代评估。但要注意评估结果是否与人类判断一致，以防出现偏差或误导性输出。

□ **常见偏差** – 模型在评估过程中可能会表现出以下几种偏差：

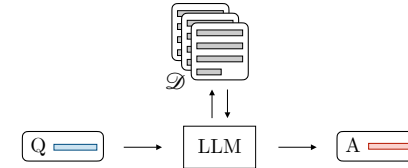
	位置偏差	冗长偏差	自我增强偏差
问题	在成对比较中偏好排在前面的项	偏好篇幅更长的内容	偏好由自身生成的内容
解决方案	随机打乱评估项顺序，并对多轮结果取平均值	对输出长度添加惩罚项	使用与被评估模型不同的基础模型构建评审器

为缓解这些问题，可以选择微调一个自定义的评审模型。但这通常需要大量的额外工作。

注：上述偏差仅为常见示例，实际中可能还存在其他类型的评估偏差。

4.2 检索增强生成

□ **定义** – 检索增强生成 (*Retrieval-Augmented Generation*, RAG) 是一种使大语言模型能够访问相关的外部知识来回答给定问题的方法。这在需要引入超出预训练截止日期之后的信息时尤其有用。



给定知识库 \mathcal{D} 和问题 Q ，检索器检索最相关文档，将检索内容注入上下文提示，由大模型输出答案 A 。

注：检索通常依赖于仅编码器模型生成的嵌入。

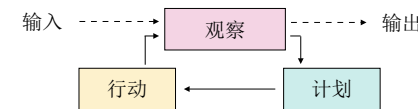
□ **超参数** – 知识库 \mathcal{D} 的初始化包括：将文档划分为大小为 n_c 的块 (chunk)，并将其嵌入为维度为 \mathbb{R}^d 的向量。



4.3 智能体

□ **定义** – 智能体是指能够自主完成任务、追求目标的系统，通常通过多轮 LLM 调用链执行任务。

□ **ReAct 框架** – *Reason + Act* (ReAct) 是一个支持通过多轮大型语言模型 (LLM) 调用来完成复杂任务的框架：



这种框架的流程如下：

- 观察：总结当前已知信息。
- 计划：规划待完成的任务与所需工具。
- 行动：调用 API 或检索知识执行操作。

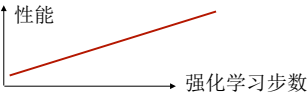
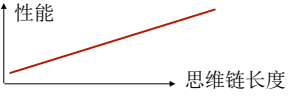
注：智能体系统的评估较复杂，但可通过组件级输入输出或完整链路进行评估。

4.4 推理模型

□ **定义** – 推理模型（reasoning models）是指依赖于思维链（CoT）推理轨迹来完成复杂任务（如数学、编程、逻辑推理）的模型。包括：OpenAI 的 o 系列，DeepSeek-R1，Google Gemini Flash Thinking。

注：DeepSeek-R1 会将其推理过程显式地输出在 <think> 标签之间。

□ **推理能力扩展** – 增强推理能力的两种方式：

	描述	示例图
训练时扩展	通过更长时间的强化学习训练，让模型学会输出 CoT 推理路径	 <p>A line graph with '性能' (Performance) on the y-axis and '强化学习步数' (Reinforcement Learning Steps) on the x-axis. A red line starts at a low point and trends upwards to the right, indicating that performance improves as the number of reinforcement learning steps increases.</p>
测试时扩展	在生成答案前通过提示词（如“Wait”）让模型“多思考”一些	 <p>A line graph with '性能' (Performance) on the y-axis and '思维链长度' (Chain of Thought Length) on the x-axis. A red line starts at a low point and trends upwards to the right, indicating that performance improves as the length of the chain of thought increases.</p>