



# Introduction to LLM

周赵嘉程

2025 年 11 月 30 日



上海交通大学  
SHANGHAI JIAO TONG UNIVERSITY



# 目录

1 Introduction

2 Overview

3 文本处理

4 注意力机制

5 GPT 实现

第 1 节

## Introduction



# Introduction



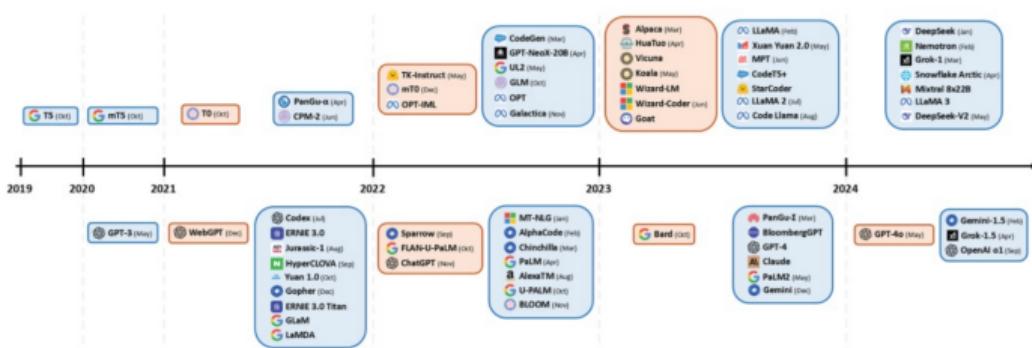
- 自 20 世纪 50 年代图灵测试提出以来，人类一直在探索如何让机器掌握语言智能。
- 语言本质上是一个复杂精妙的人类表达系统，受语法规则的约束。
- 开发能够理解和掌握语言的人工智能（AI）算法是一项巨大的挑战

# 什么是大语言模型 (What are LLMs?)



- 基于深度学习的大规模概率模型，参数量通常在数十亿 (Billions) 至万亿级别。
- 这里的“大”既指参数量巨大，也指训练数据的规模巨大 (万亿级 Tokens)。
- 绝大多数现代 LLM 基于 **Transformer** 架构 (特别是 Decoder-only 结构，如 GPT 系列)
- 利用自注意力机制 (Self-Attention) 捕捉长距离的文本依赖关系。
- **自监督学习 (Self-Supervised Learning):** 不需要人工标注，通过“预测下一个词”(Next Token Prediction) 进行训练。
- **知识压缩 (Knowledge Compression):** 模型不仅仅是在记忆文本，而是将世界知识、逻辑规则和语言模式压缩进了神经网络的权重参数中。

# 什么是大模型

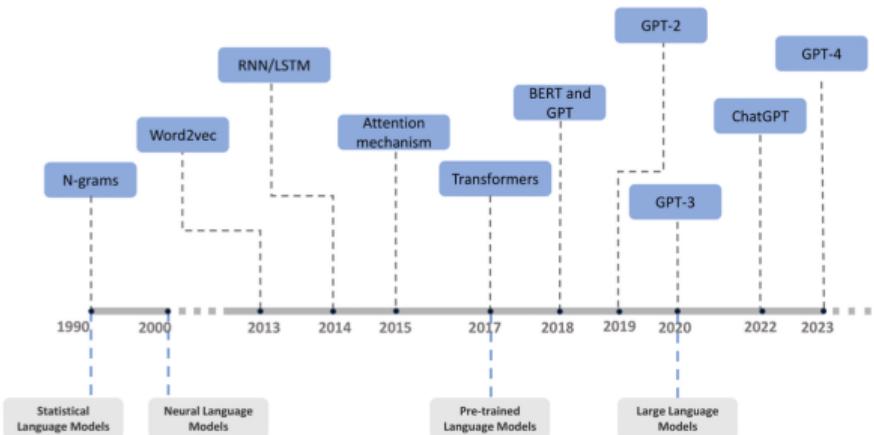


# Historical Timeline of LLMs



- SLM: 统计语言模型
- NLM: 神经语言模型
- PLM: 预训练语言模型
- LLM: 大语言模型

# Historical Timeline of LLMs



# 从 PLM 到 LLM 的范式转变



- PLM (Pre-trained Language Models) 时代 (e.g., BERT, RoBERTa):
  - **范式:** 预训练 + 微调 (Pre-train + Fine-tune)。
  - **局限:** 需要针对特定任务 (Task-specific) 收集标注数据并更新参数；模型往往是“专才”而非“通才”。
- LLM (Large Language Models) 时代 (e.g., GPT-3, GPT-4, Llama):
  - **范式:** 预训练 + 提示工程/上下文学习 (Pre-train + Prompt/In-Context Learning)。
  - **优势:** 冻结参数即可处理多种任务；通过指令微调 (Instruction Tuning) 实现通用意图理解。

# LLM 的核心进步 (Key Advancements)



- **扩展定律 (Scaling Laws):**
  - 随着参数量、数据量和计算量的增加，模型性能呈幂律提升，并未出现瓶颈 [2]。
- **指令微调 (Instruction Tuning):**
  - 通过在多任务指令数据集上微调，模型学会了“遵循指令”而非仅仅“预测下一个词”，显著提高了零样本泛化能力 [3]。
- **人类反馈强化学习 (RLHF):**
  - 将模型输出与人类价值观（有用性、诚实性、安全性）对齐，解决了 PLM 生成内容不可控的问题 [3]。

# LLM 的特殊能力 (Emergent Abilities)



当模型规模突破一定临界值（如  $10^{22}$  FLOPs）时，会出现小模型不具备的涌现能力 (**Emergent Abilities**) [4]：

- **上下文学习 (In-Context Learning, ICL):**
  - 模型无需梯度更新，仅通过 Prompt 中的几个示例即可学会新任务 [5]。
- **思维链 (Chain-of-Thought, CoT):**
  - 通过生成中间推理步骤，显著提升了处理复杂逻辑、数学和常识推理任务的能力 [6]。
- **指令遵循与泛化:**
  - 能够处理训练集中从未见过的任务描述，表现出真正的通用智能特征。

# 大语言模型的当前应用



- **编程智能体:** GitHub Copilot, Cursor, CodeT5, 以及基于 GPT 的编程助手。
- **对话式 AI:** ChatGPT, Claude, Gemini, 用于客户服务及个人助理。
- **内容生成 (Content Generation):** 自动化文章撰写、创意写作、营销内容生成。
- **科研辅助 (Research Assistance):** Alpha evolve, GPT Deepresearch, 文献综述加速、研究假设生成、辅助数据分析。

# Attention Mechanism



## Attention Is All You Need

**Ashish Vaswani\***  
Google Brain  
avaswani@google.com

**Noam Shazeer\***  
Google Brain  
noam@google.com

**Niki Parmar\***  
Google Research  
nikip@google.com

**Jakob Uszkoreit\***  
Google Research  
usz@google.com

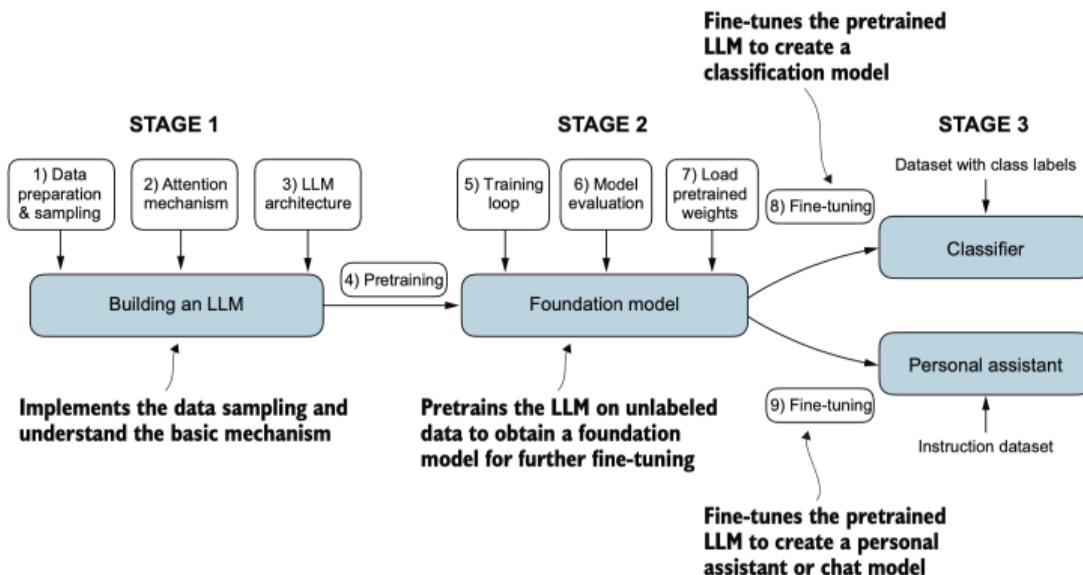
**Llion Jones\***  
Google Research  
llion@google.com

**Aidan N. Gomez\* †**  
University of Toronto  
aidan@cs.toronto.edu

**Łukasz Kaiser\***  
Google Brain  
lukaszkaiser@google.com

**Illia Polosukhin\* ‡**  
illia.polosukhin@gmail.com

# Building LLM



第 2 节

## Overview

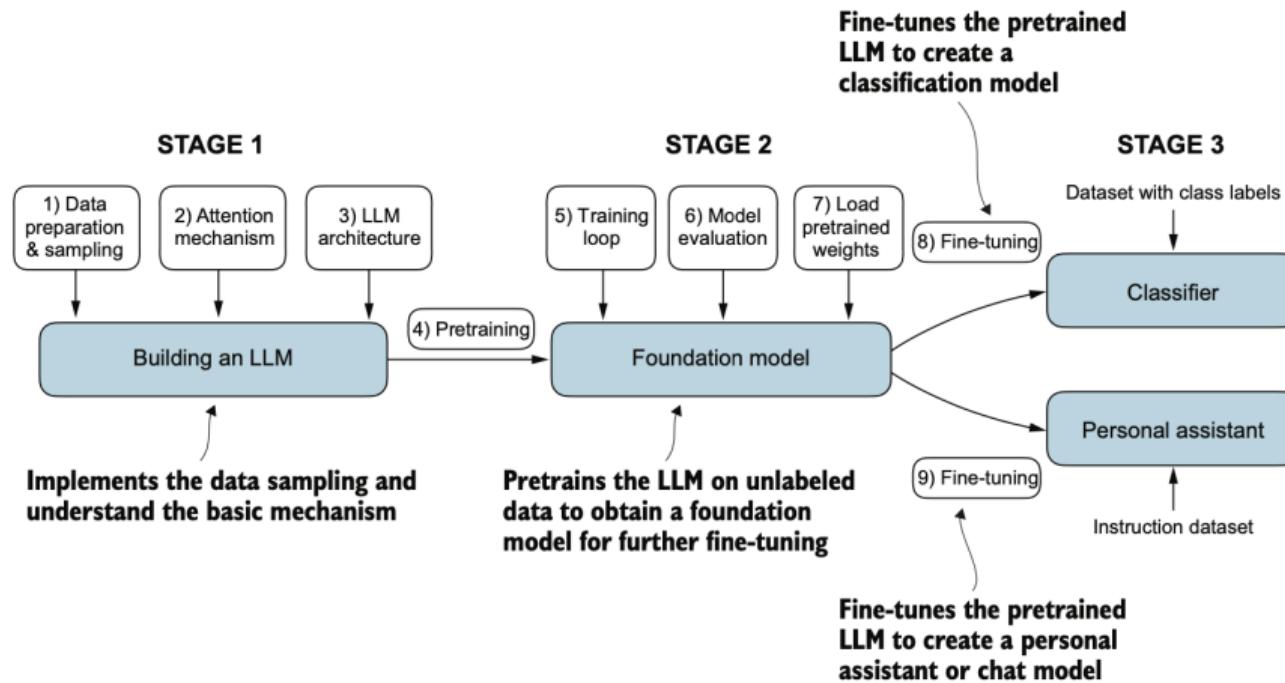


# Credit

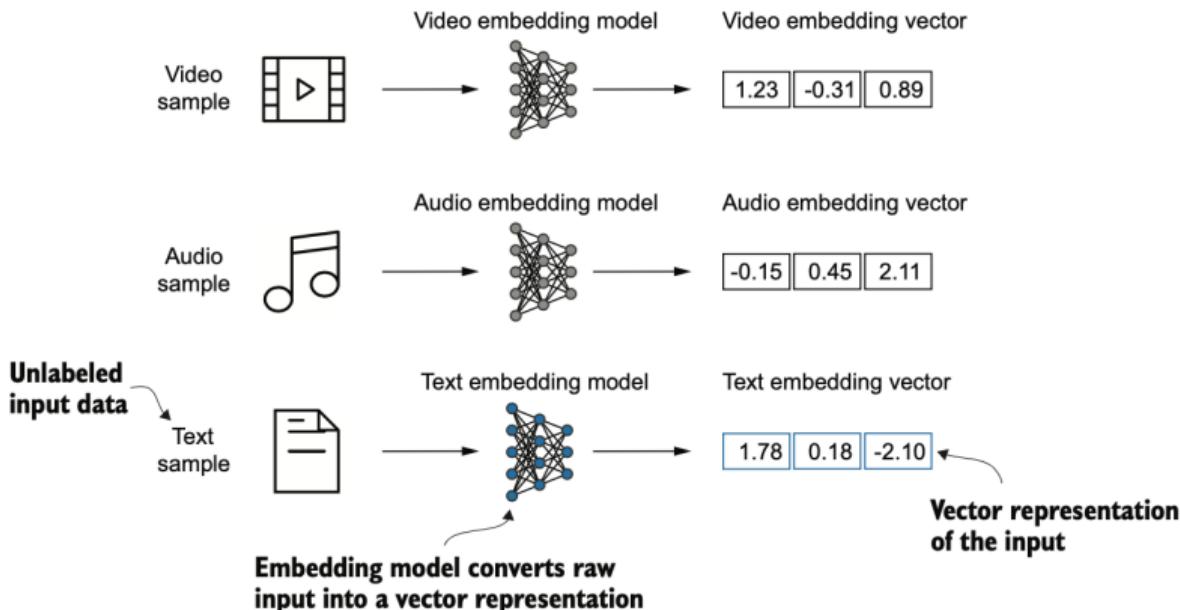


以下部分主要来自**build-llms-from-scratch-book**

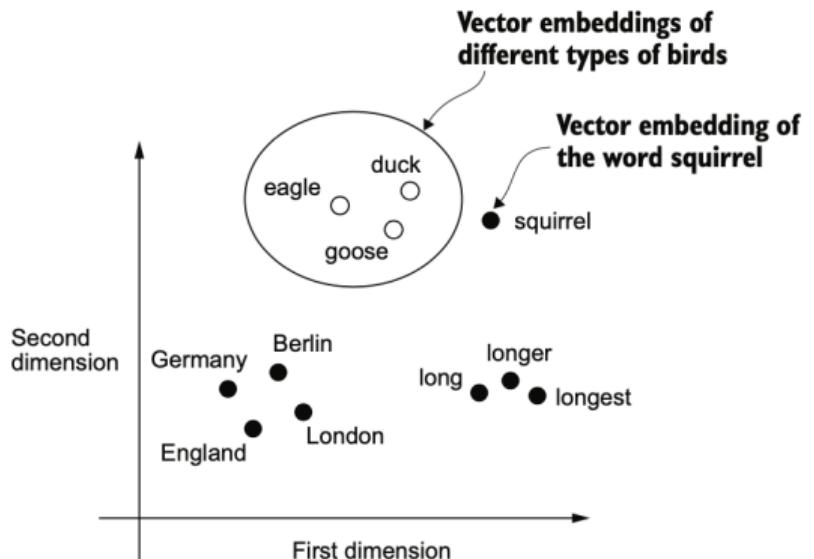
# Building Overview



# Overview



# Overview

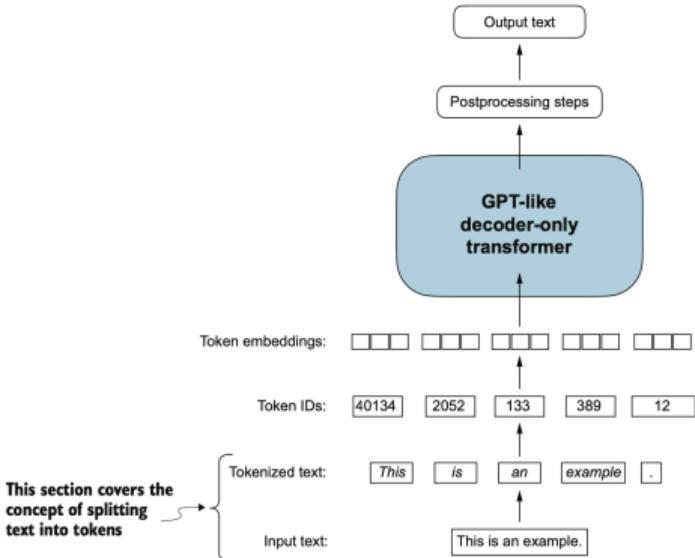


第 3 节

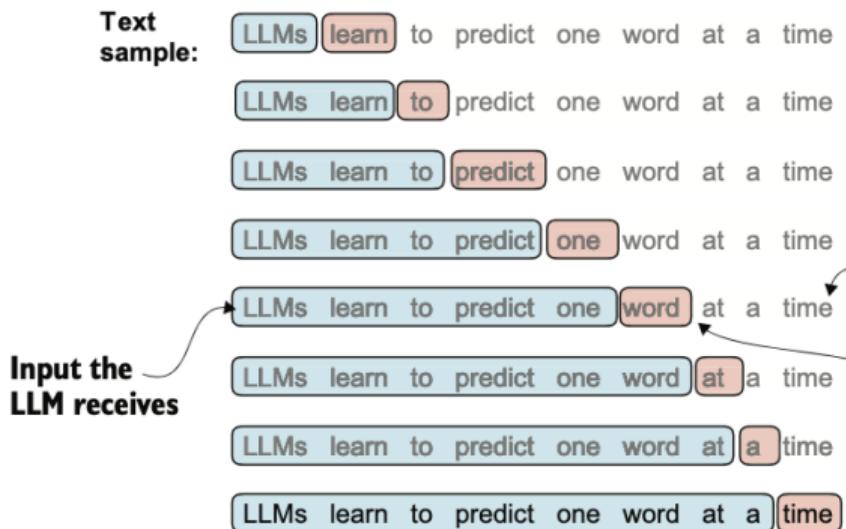
## 文本处理



# GPT Workflow



# GPT Workflow



# GPT Workflow

Sample text

"In the heart of the city stood the old library, a relic from a bygone era. Its stone walls bore the marks of time, and ivy clung tightly to its facade ..."

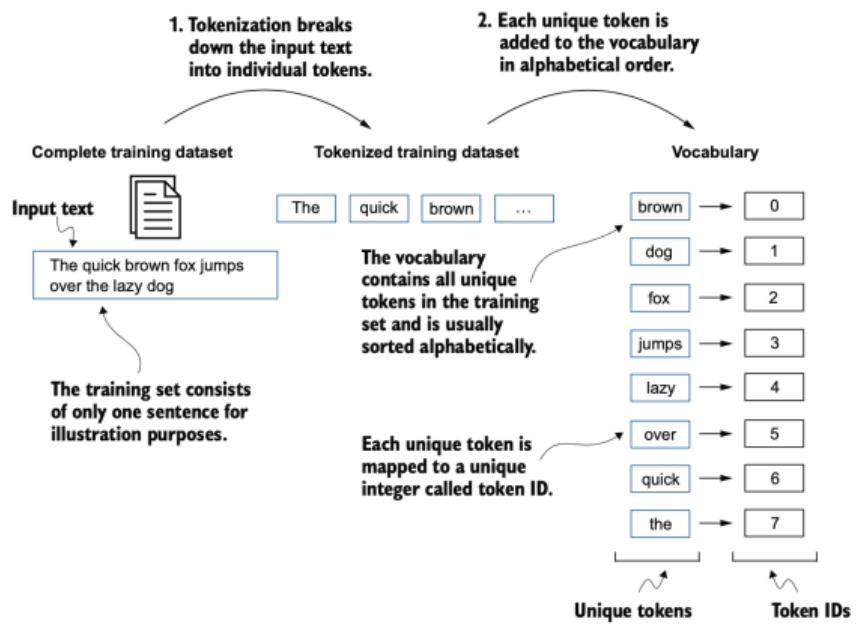
Tensor  
containing  
the inputs

```
x = tensor([[ "In",      "the",      "heart",     "of"  ],
           [ "the" ,     "city",     "stood",    "the"  ],
           [ "old",      "library",   "",         "a"    ],
           [ ... ]])
```

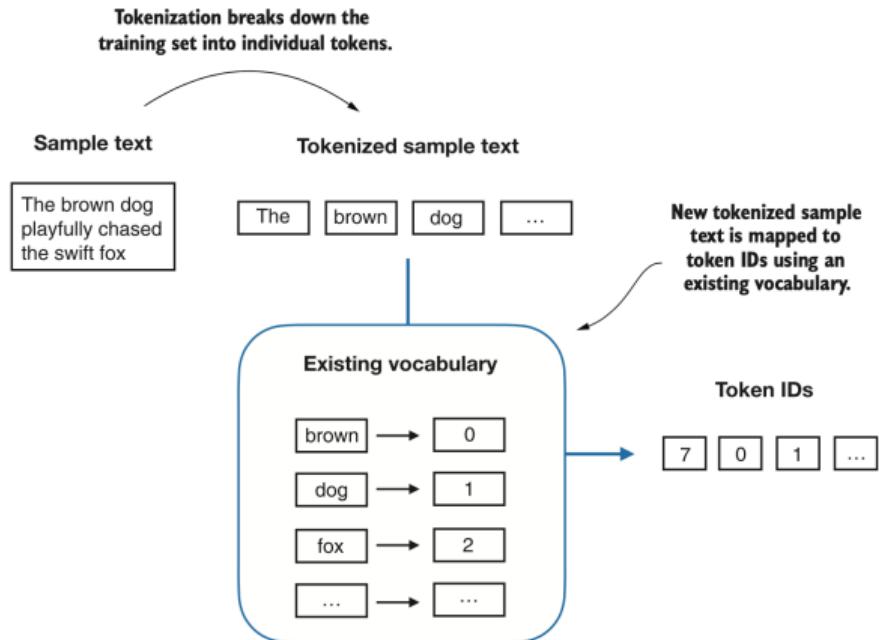
Tensor  
containing  
the targets

```
y = tensor([[ "the",      "heart",     "of",       "the"  ],
           [ "city",      "stood",     "the",     "old"  ],
           [ "library",   "",         "a",       "relic" ],
           [ ... ]])
```

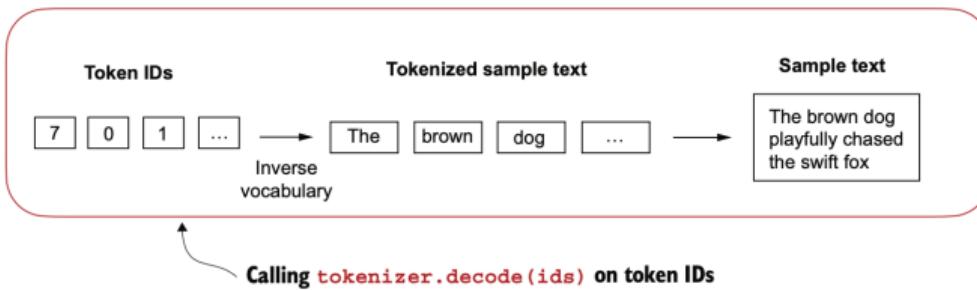
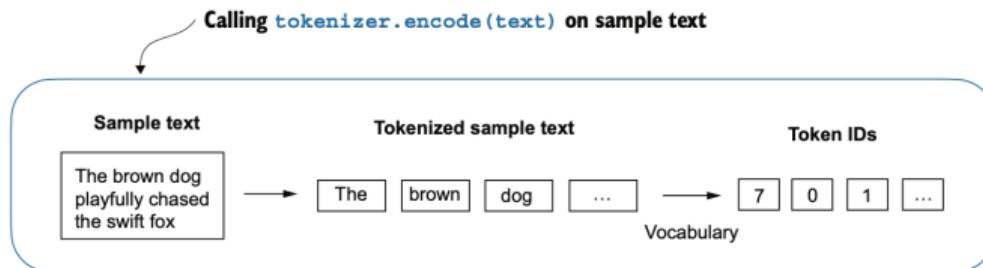
# Tokenizer



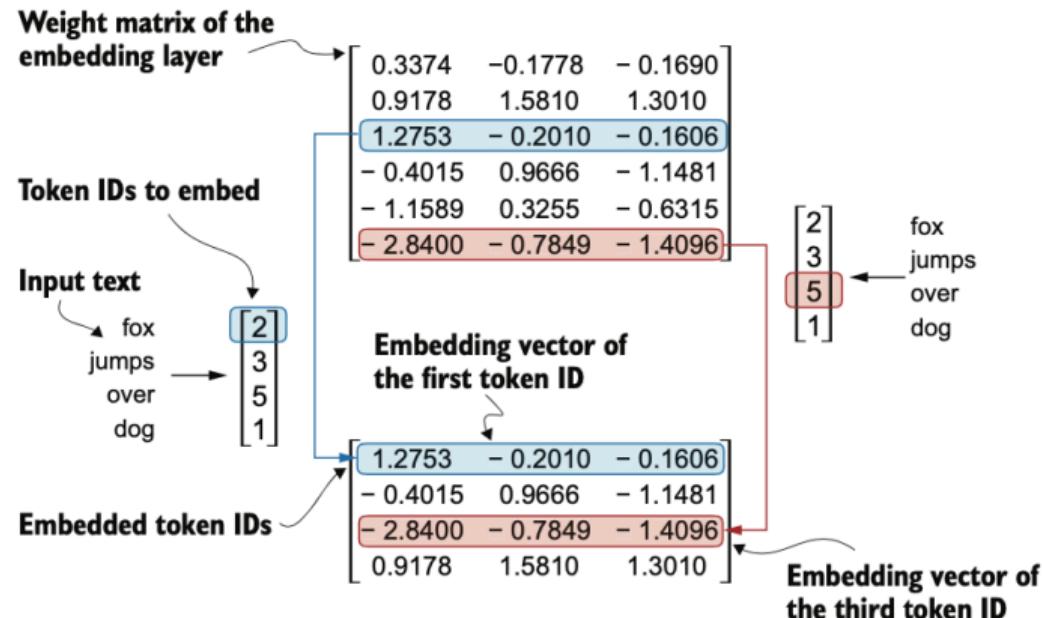
# Tokenizer



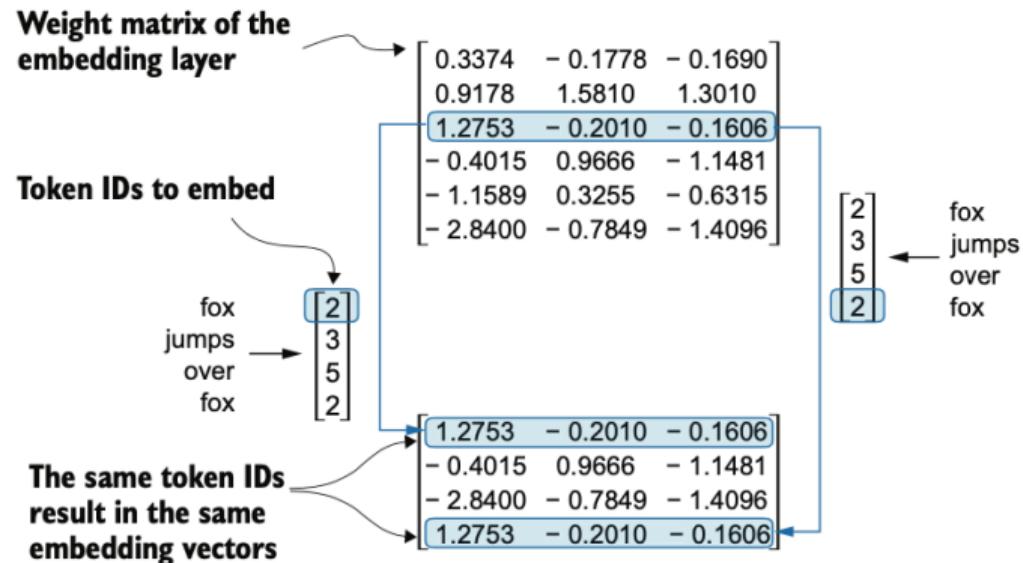
# Encoding & Decoding



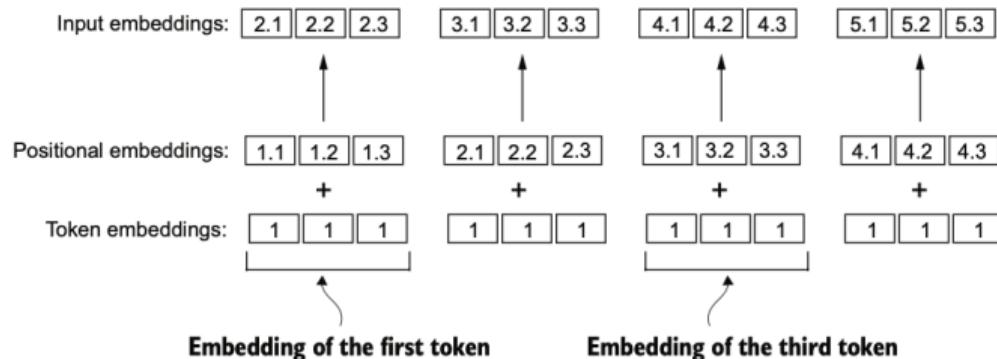
# Embedding



# Embedding



# Embedding

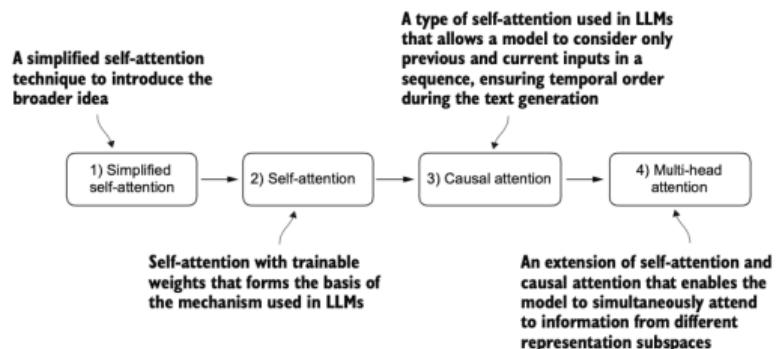


## 第 4 节

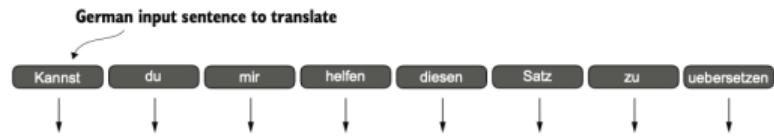
# 注意力机制



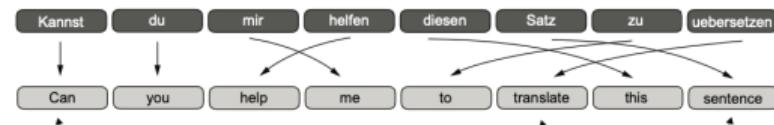
# Attention Evolution



# Problem

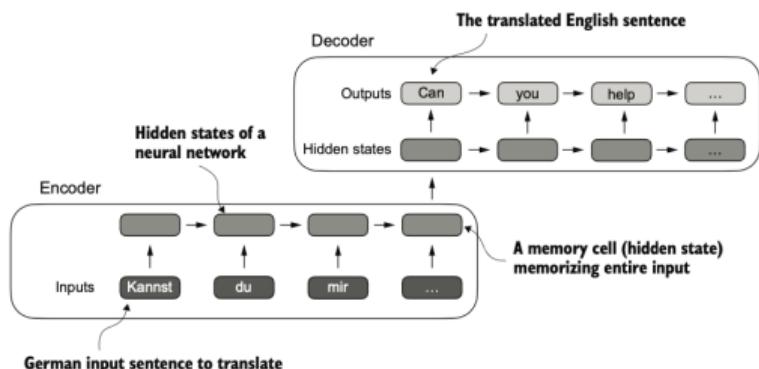


The word-by-word translation results  
in a grammatically incorrect sentence

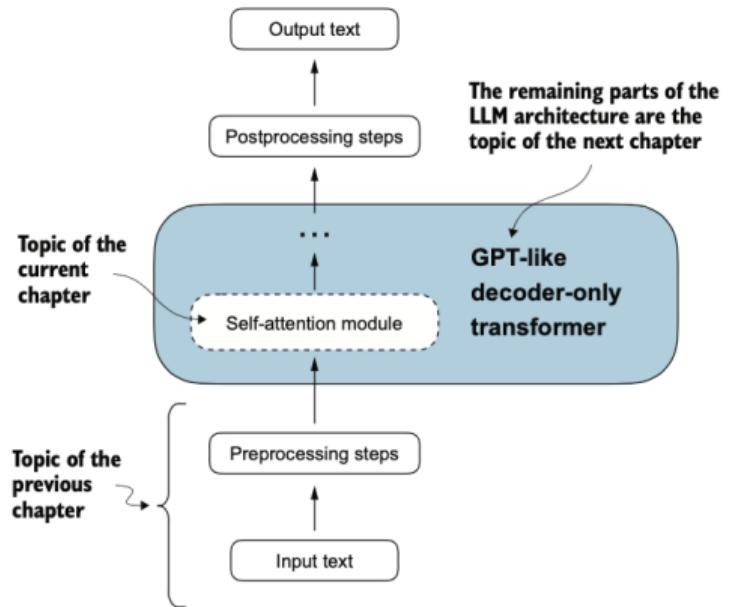


Certain words in the generated translation  
require access to words that appear earlier  
or later in the original sentence.

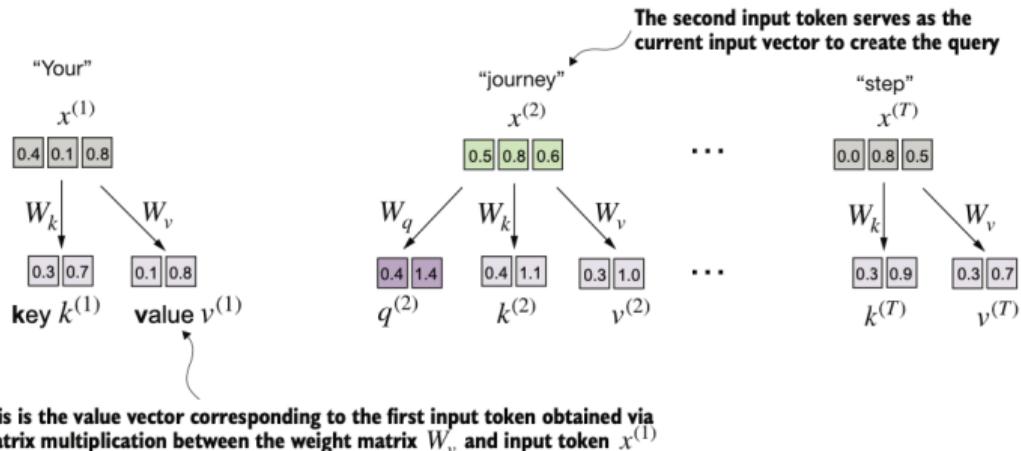
# Solution



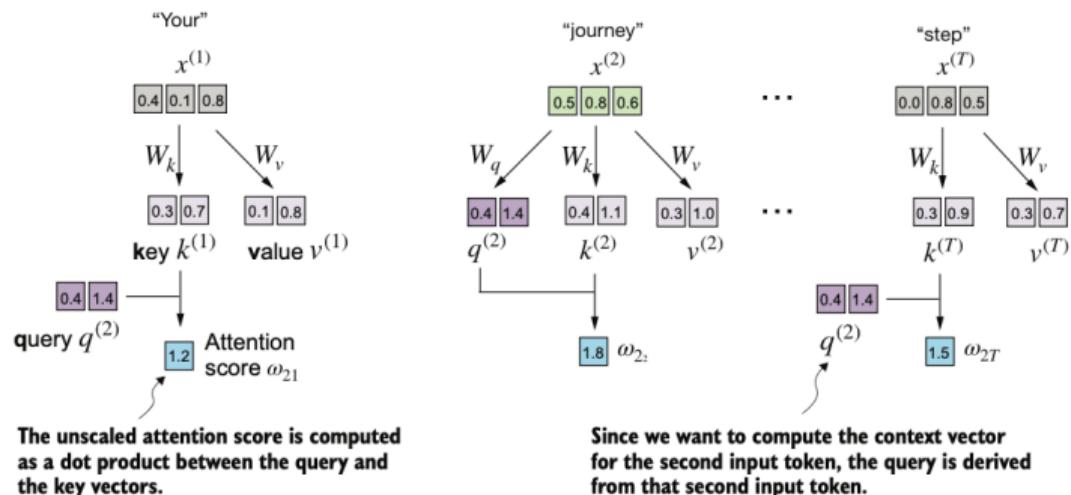
# Self Attention Module



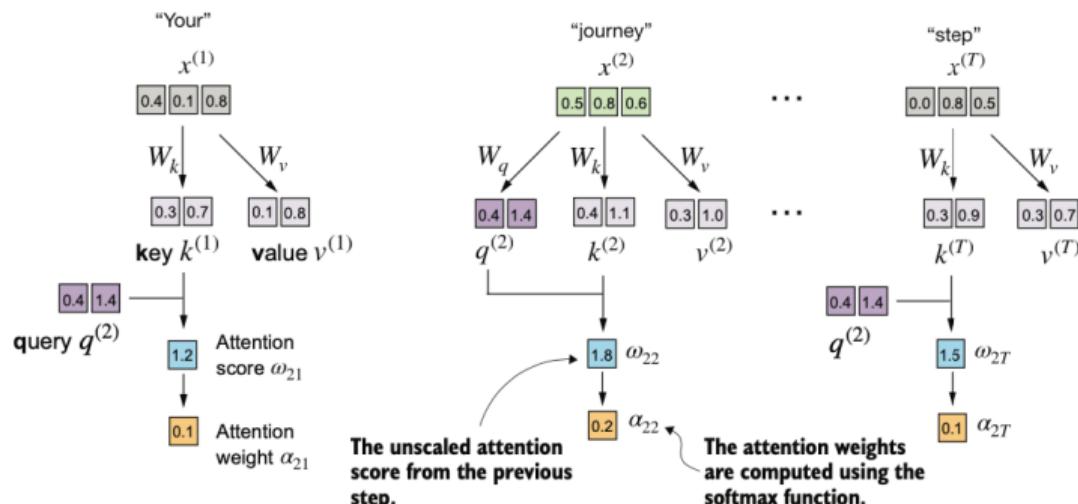
# Compute Self Attention



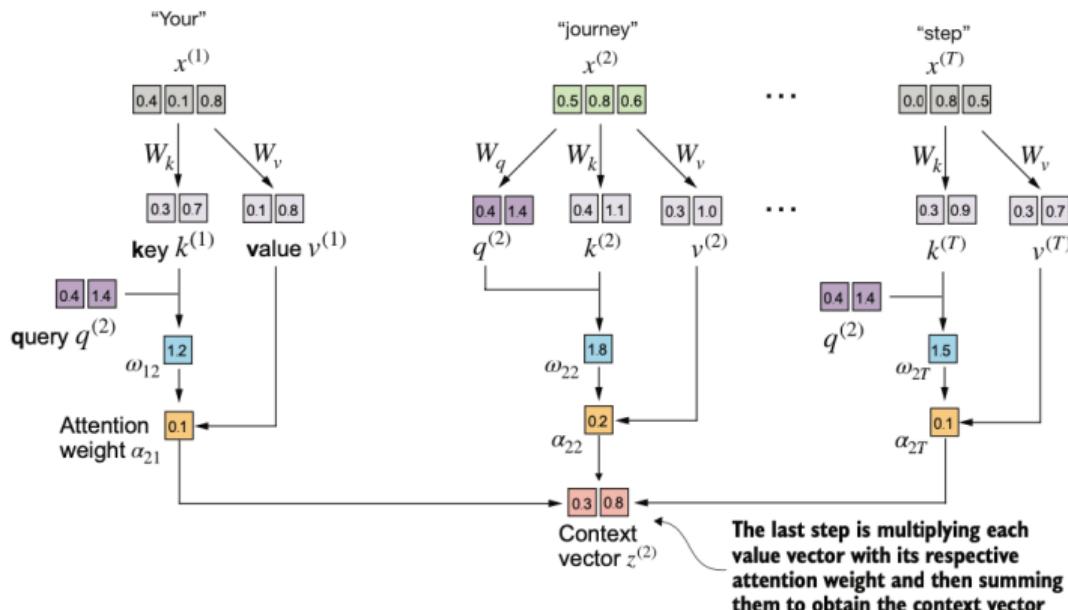
# Compute Self Attention



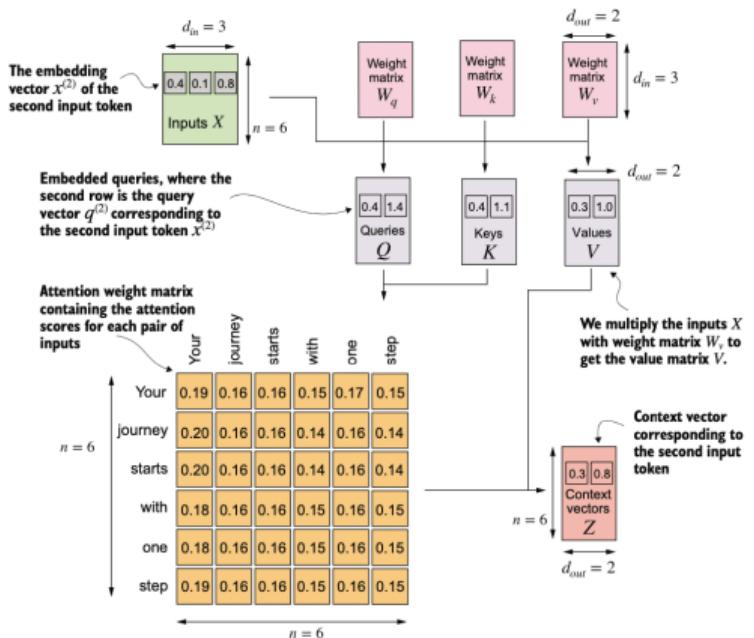
# Compute Self Attention



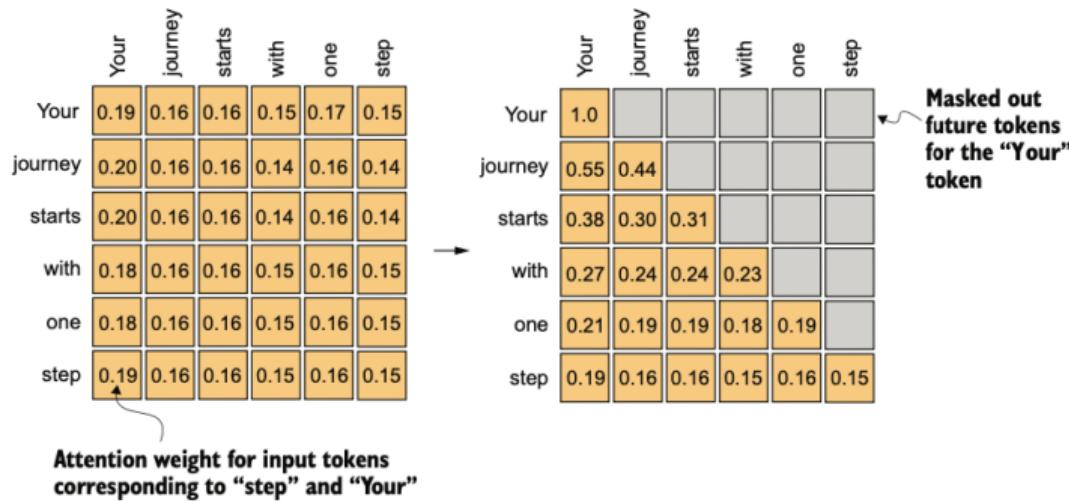
# Compute Self Attention



# Compute Self Attention



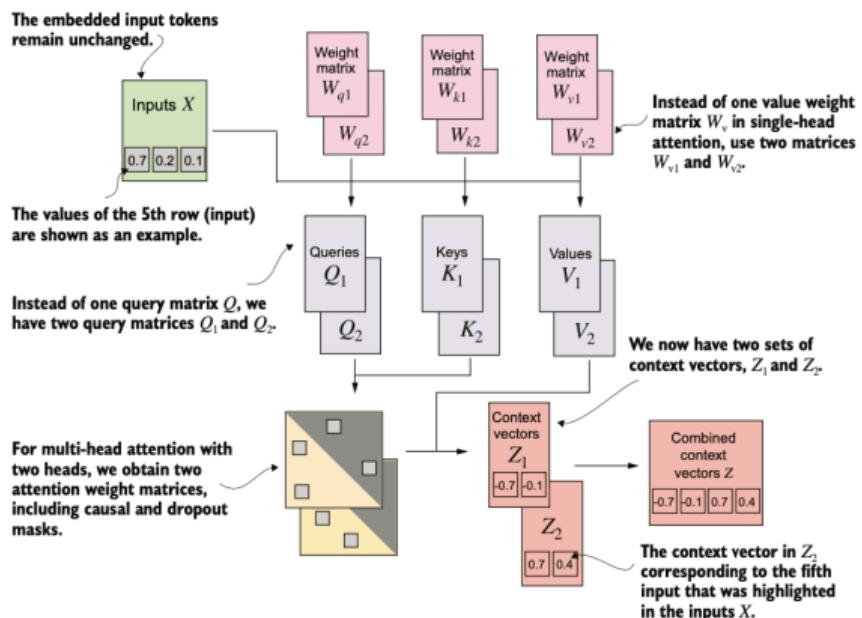
# Causal Attention



# Masked Attention



# Multihead Attention

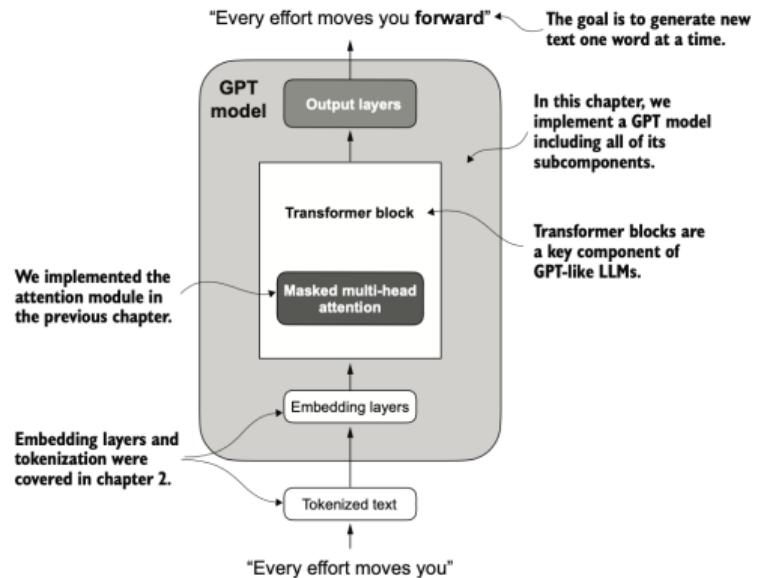


第 5 节

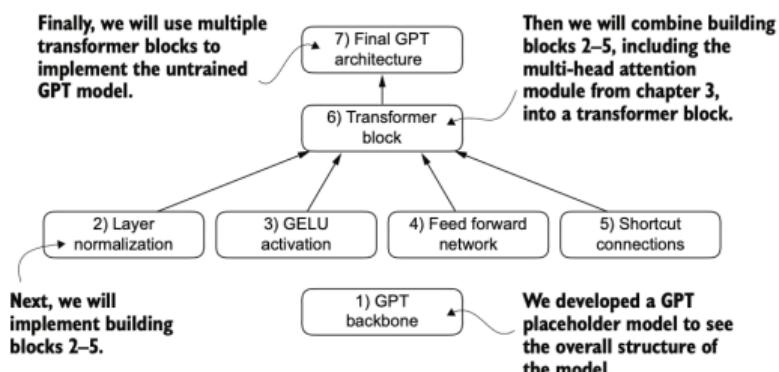
## GPT 实现



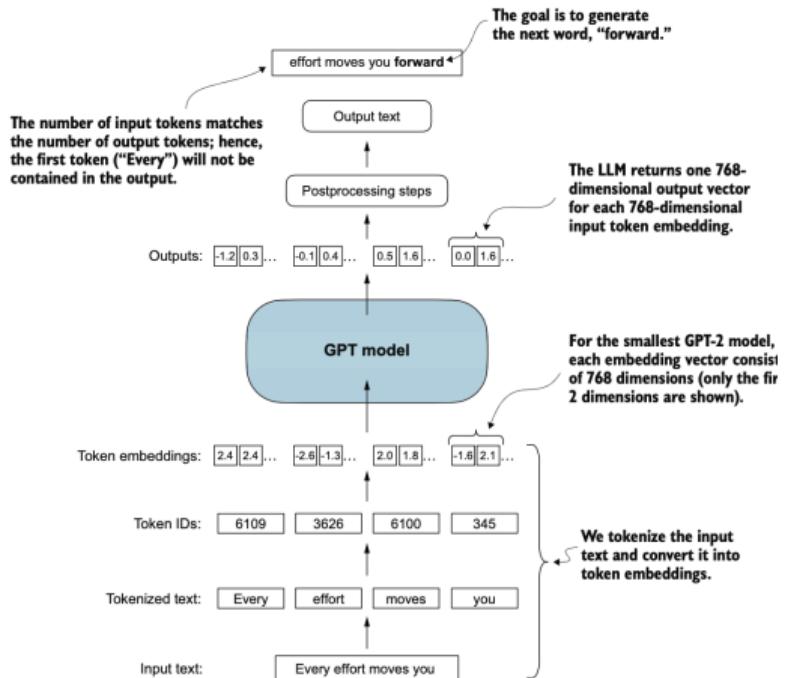
# Quick Review



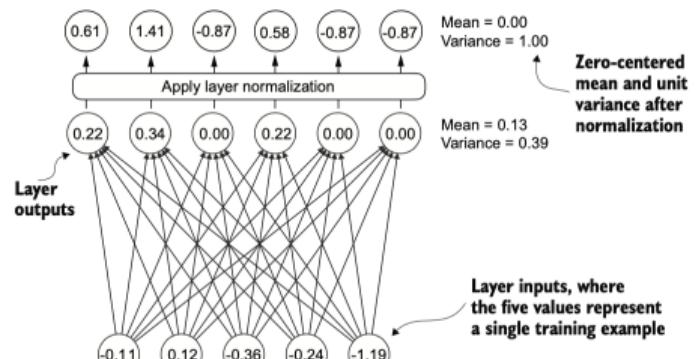
# 核心组件



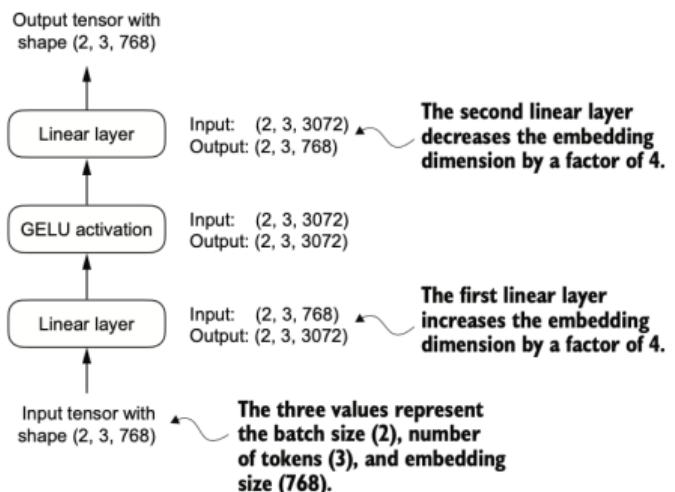
# Workflow Review



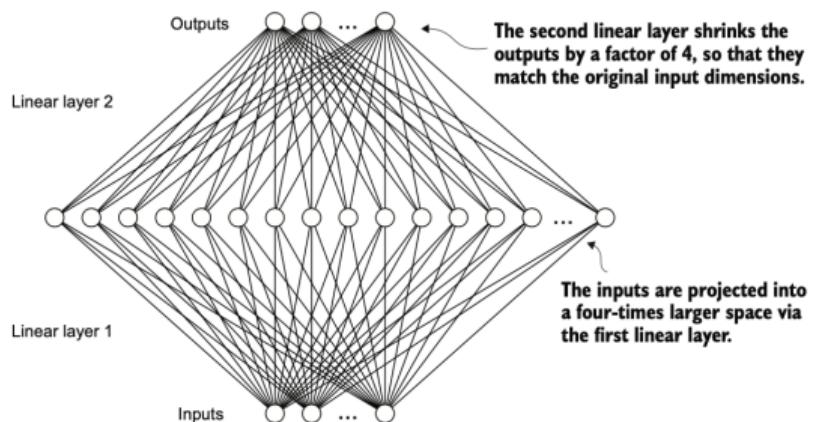
# Normalization



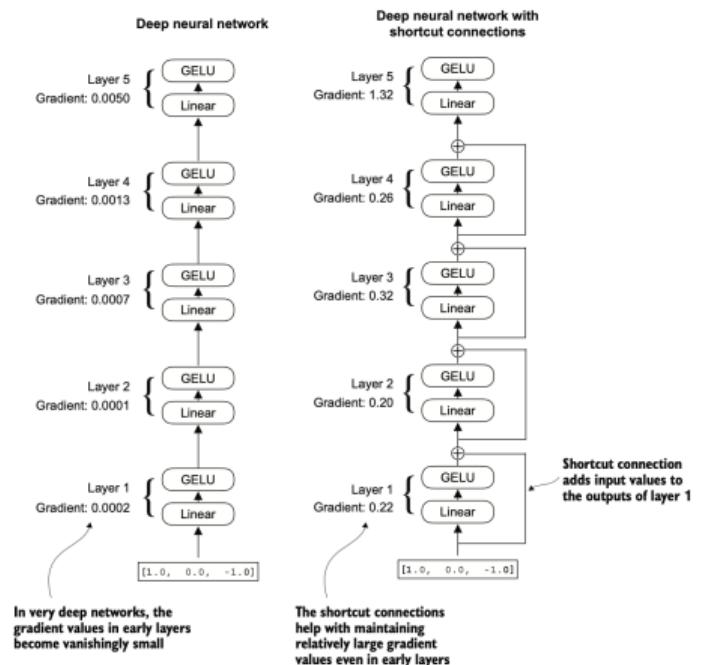
# Layer Connection



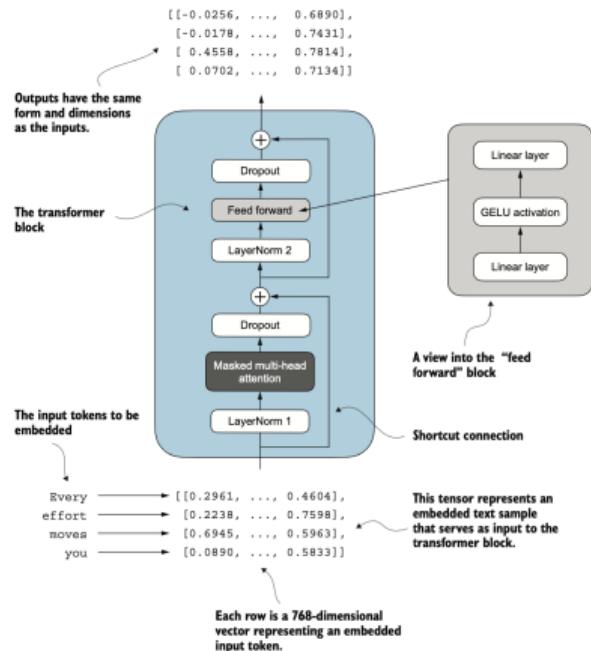
# FNN



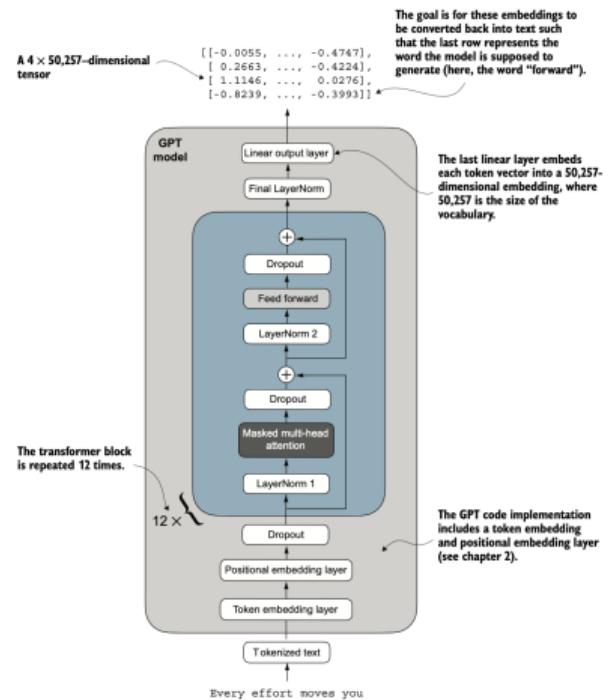
# Short Cut



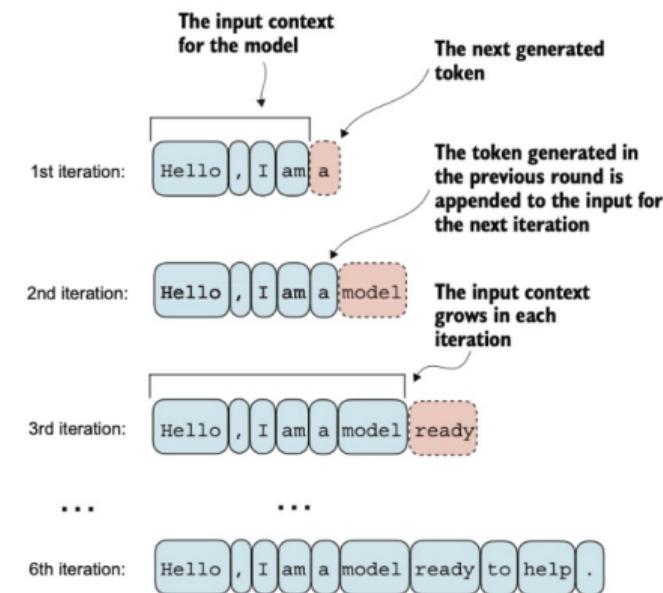
# Transformer Block



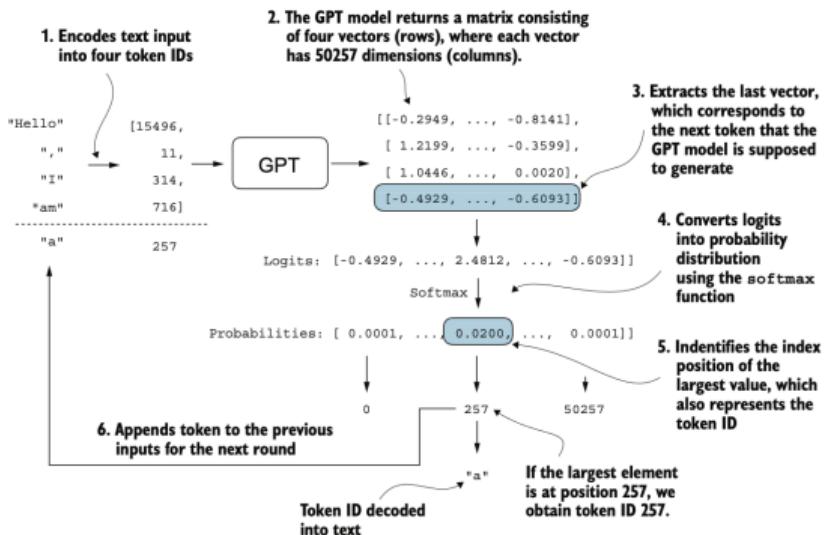
# GPT 架构



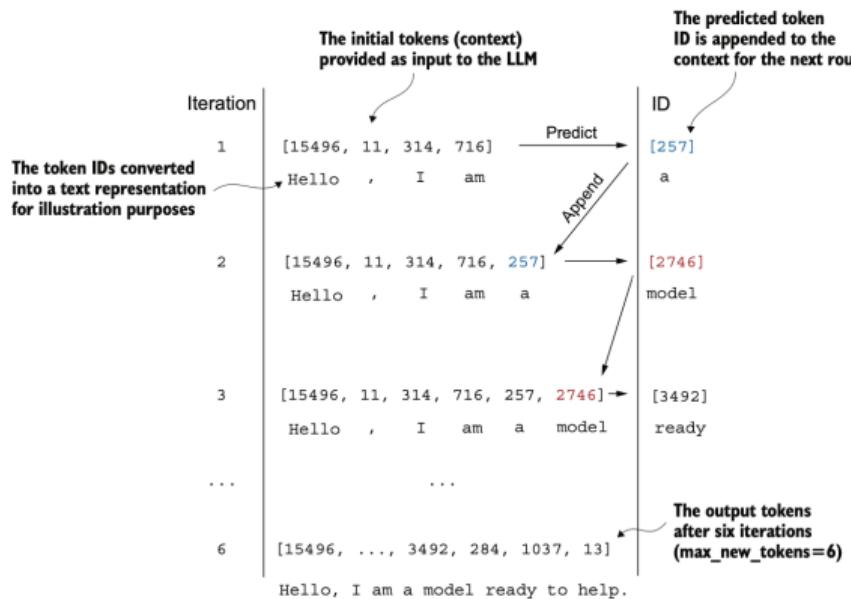
# 生成模式



# 生成模式



# Prediction Cycle



第 6 节

## Conclusion



# 检索增强生成 (Retrieval-Augmented Generation)



- **核心痛点 (Pain Points):**

- 幻觉 (Hallucination): 模型可能会一本正经地胡说八道。
- 知识时效性: 模型的知识截止于训练结束那一刻, 无法回答实时问题。

- **工作流程 (Workflow):**

- ① **Retrieve (检索):** 将用户的问题向量化, 在外部向量数据库中匹配最相关的文档片段。
- ② **Augment (增强):** 将检索到的事实作为上下文 (Context) 拼接到 Prompt 中。
- ③ **Generate (生成):** LLM 基于增强后的 Prompt 生成准确且可溯源的答案。

# 多智能体系统 (Multi-Agent Systems)



- **从单体到群体 (From Solo to Swarm):** 面对复杂的长流程任务 (如开发一个游戏)，单个 LLM 容易遗忘上下文或逻辑混乱。
- **协作机制 (Collaboration Mechanism):**
  - **角色扮演 (Role Playing):** 不同的 Agent 扮演不同角色 (如产品经理、架构师、工程师、测试员)。
  - **流程标准化 (SOP):** 将复杂任务拆解为流水线，Agent 之间通过对话进行规划、执行和互相审查。
- **典型框架:**
  - **AutoGen (Microsoft):** 允许创建可定制、可对话的 Agent 网络。
  - **MetaGPT:** 只要给一句需求，就能输出整个软件工程项目

# 推理加速关键技术：KV Cache



- **背景 (Background):** LLM 推理是自回归 (Auto-regressive) 的，每生成一个 Token，都要重新计算一次 Attention。
- **空间换时间 (Space-Time Trade-off):**
  - Attention 公式:  $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$
  - 生成第  $t$  个 token 时，前  $t - 1$  个 token 的  $K$  (Key) 和  $V$  (Value) 矩阵计算结果是 \*\* 不变的 \*\*。
  - **KV Cache:** 将历史的  $K, V$  向量缓存在显存中，每次只计算当前新 token 的  $Q, K, V$ ，避免重复计算。
- **挑战 (Challenge):**
  - **显存杀手:** KV Cache 随序列长度线性增长，是长文本推理显存占用过高 (Memory Bound) 的主要原因。
  - 解决方案示例：*PagedAttention (vLLM)*.

第 7 节

## References



# References I

- [1] A. Vaswani et al., "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
- [2] J. Kaplan et al., *Scaling laws for neural language models*, 2020. arXiv: 2001.08361 [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2001.08361>.
- [3] L. Ouyang et al., *Training language models to follow instructions with human feedback*, 2022. arXiv: 2203.02155 [cs.CL]. [Online]. Available: <https://arxiv.org/abs/2203.02155>.
- [4] J. Wei et al., *Emergent abilities of large language models*, 2022. arXiv: 2206.07682 [cs.CL]. [Online]. Available: <https://arxiv.org/abs/2206.07682>.

## References II



- [5] T. B. Brown et al., *Language models are few-shot learners*, 2020. arXiv: 2005.14165 [cs.CL]. [Online]. Available: <https://arxiv.org/abs/2005.14165>.
- [6] J. Wei et al., *Chain-of-thought prompting elicits reasoning in large language models*, 2023. arXiv: 2201.11903 [cs.CL]. [Online]. Available: <https://arxiv.org/abs/2201.11903>.

谢 谢



上海交通大学  
SHANGHAI JIAO TONG UNIVERSITY

