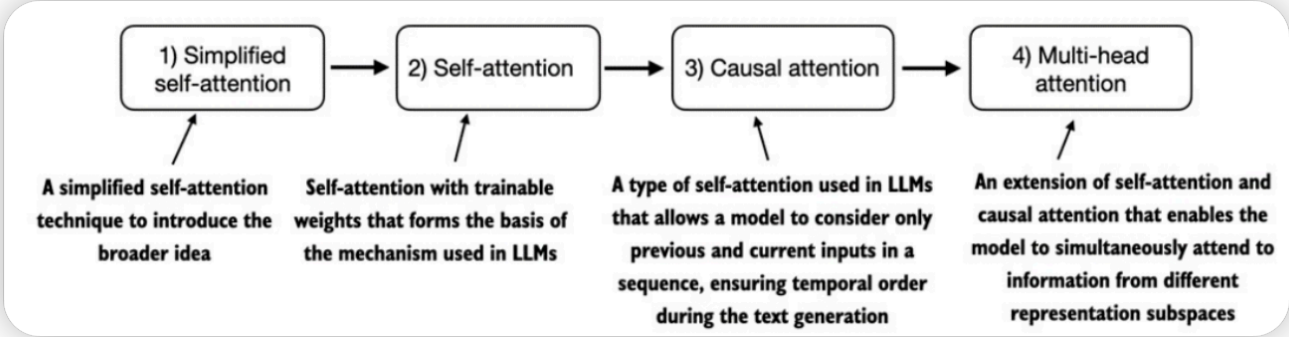
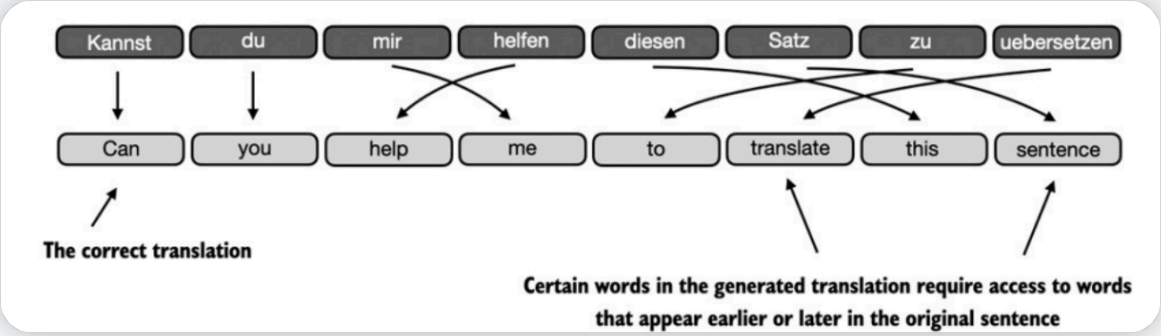
- Attention 注意力机制：将输入向量转换为增强的上下文向量。
- 添加 Dropout层 遮蔽来减少大语言模型的过拟合问题。

下图是将在本章编写的四种注意力机制。



长序列建模问题

> 引入：翻译由于源语言和目标语言的语法结构，我们不能简单地逐字翻译文本
>
> 
>
>
> 使用两个DNN模块：编码器和解码器。

- 在Transformer架构出现之前，RNN是最流行的编码器和解码器架构
  - 编码器将整个输入处理成隐藏层状态，解码器仅依据此生成输出，无法从编码器访问早期的隐藏状态。可能会导致跨长句子依赖关系的丢失。

使用注意力机制捕获数据依赖关系

解码器可以有选择的访问所有输入令牌。

自注意力机制

定义

允许输入序列的每个位置在计算序列的表示时关注同一序列的所有位置。

目标

为每个输入元素，计算一个上下文向量。

作用

通过整合序列中所有其他元素的信息，为输入序列中的每一个元素创造出更丰富的表征。

简单自注意力

3-1简单自注意力

- 通过 `attn_scores = inputs @ inputs.T` 计算注意力得分

带有可训练权重的自注意力

3-2带有可训练权重的自注意力

- 引入单组查询 W_q ， 键 W_k，值 W_v 权重矩阵，
- 通过 `attn_scores = queries @ keys.T` 计算注意力得分

使用因果注意力机制隐藏后续词

3-3因果注意力

- 对注意力权重进行遮蔽（masked attention）

多头注意力

3-4多头注意力

- 多组查询 W_q ，键 W_k，值 W_v 权重矩阵。