编码器与解码器LLM全解析：掌握NLP核心技术的关键！

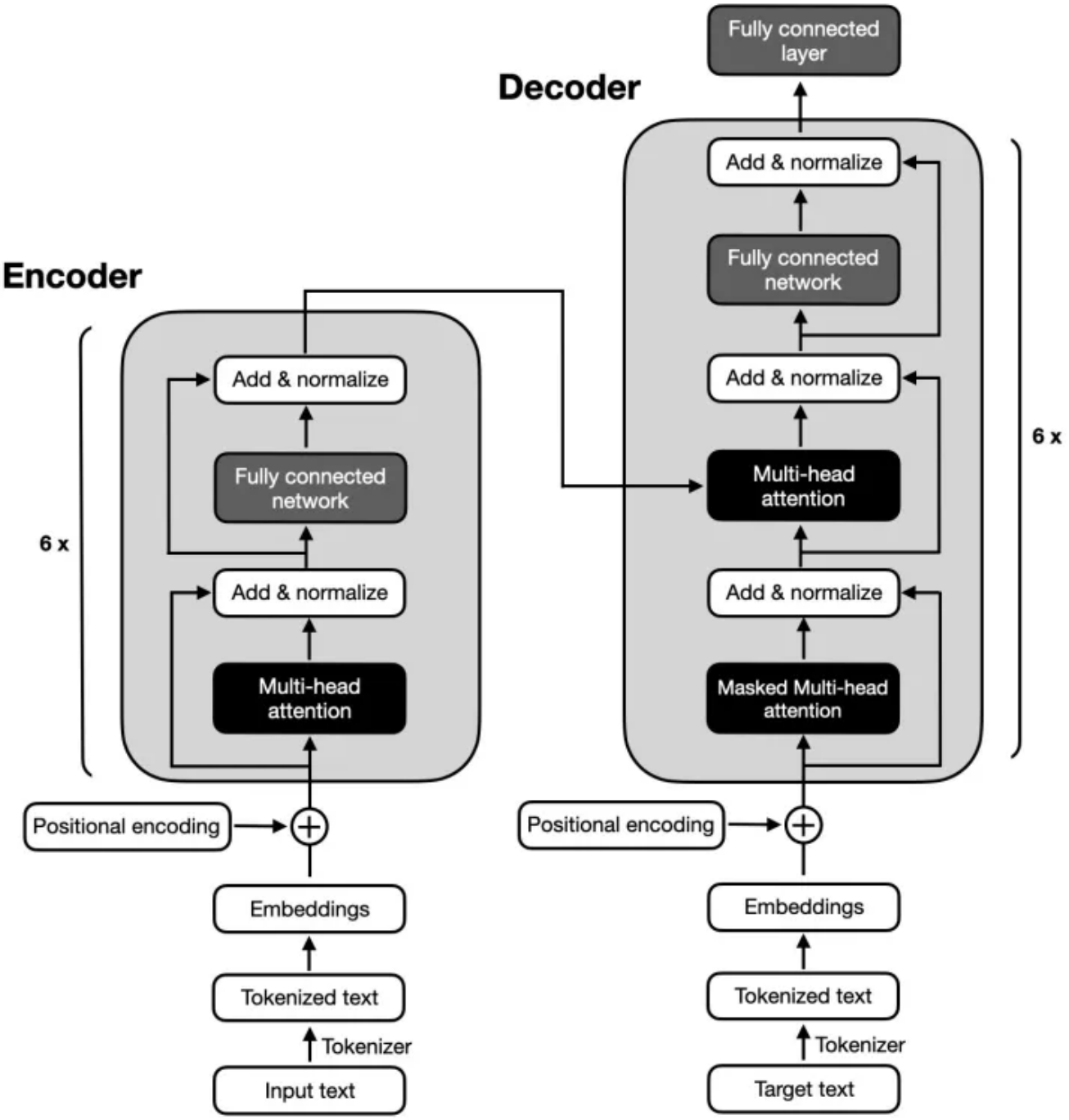
让我们深入了解：基于编码器和基于解码器的模型有什么区别？

编码器与解码器风格的Transformer

从根本上说，编码器和解码器风格的架构都使用相同的自注意力层来编码词汇标记。然而，主要区别在于编码器旨在学习可以用于各种预测建模任务（如分类）的嵌入表示。相比之下，解码器则设计用于生成新文本，例如回答用户查询。

原始的Transformer

2017年开发的原始Transformer架构，旨在进行英译法和英译德的语言翻译，它同时利用了编码器和解码器，如下图所示。



在上图中，输入文本（即要翻译的文本中的句子）首先被分词成单独的词汇标记，然后通过嵌入层进行编码，再进入编码器部分。之后，在每个嵌入词汇添加位置编码向量后，这些嵌入通过多头自注意力层。多头注意力层之后是“添加 & 归一化”步骤，进行层归一化并通过跳跃连接（也称为残差或快捷连接）添加原始嵌入。最后，在进入“全连接层”后，该层是由两个全连接层组成的小型多层感知机，中间有非线性激活函数，输出再次被添加和归一化，然后传递给解码器部分的多头自注意力层。

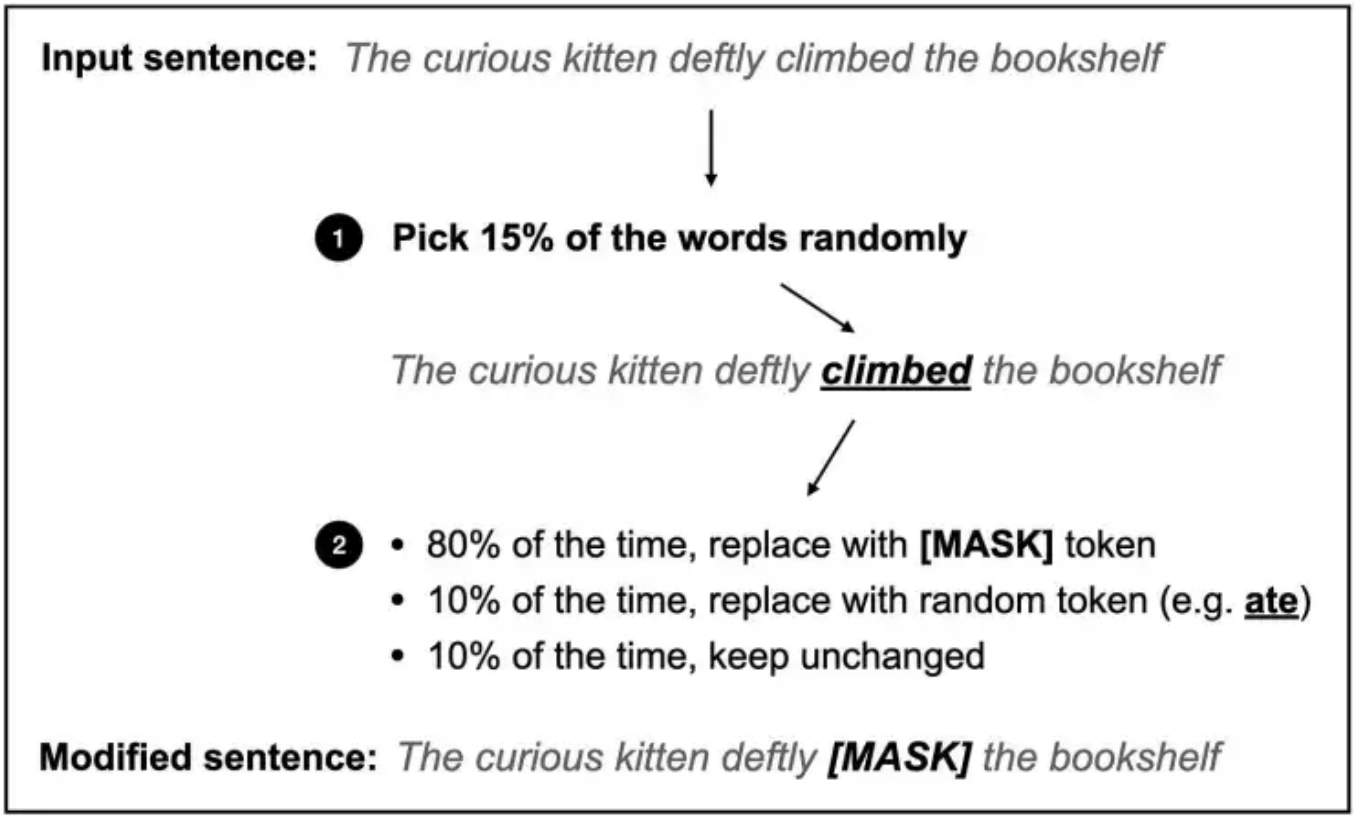
上图中的解码器部分与编码器部分有类似的整体结构。关键区别在于输入和输出不同。编码器接收要翻译的输入文本，而解码器生成翻译文本。

编码器

如前图所示，原始Transformer中的编码器部分负责理解和提取输入文本中的相关信息。然后，它输出输入文本的连续表示（嵌入），传递给解码器。最终，解码器基于从编码器接收到的连续表示生成翻译文本（目标语言）。

多年来，基于上述原始Transformer模型的编码器模块，开发出了多种仅包含编码器的架构。著名的例子包括BERT和RoBERTa。

BERT（双向编码器表示Transformer）是一种仅基于Transformer编码器模块的架构。BERT模型使用掩码语言建模（如下图所示）和下一句话预测任务在大型文本语料库上进行预训练。



掩码语言建模背后的主要思想是在输入序列中掩盖（或替换）随机的词汇标记，然后训练模型根据周围的上下文预测原始掩盖的标记。

除了上图中所示的掩码语言建模预训练任务外，下一句话预测任务要求模型预测两个随机打乱顺序的句子是否保持了原始文档的句子顺序。例如，两个以随机顺序排列的句子，由[SEP]标记分隔：

- [CLS] 吐司是一种简单但美味的食物 [SEP] 它通常与黄油、果酱或蜂蜜一起食用。
- [CLS] 它通常与黄油、果酱或蜂蜜一起食用。 [SEP] 吐司是一种简单但美味的食物。

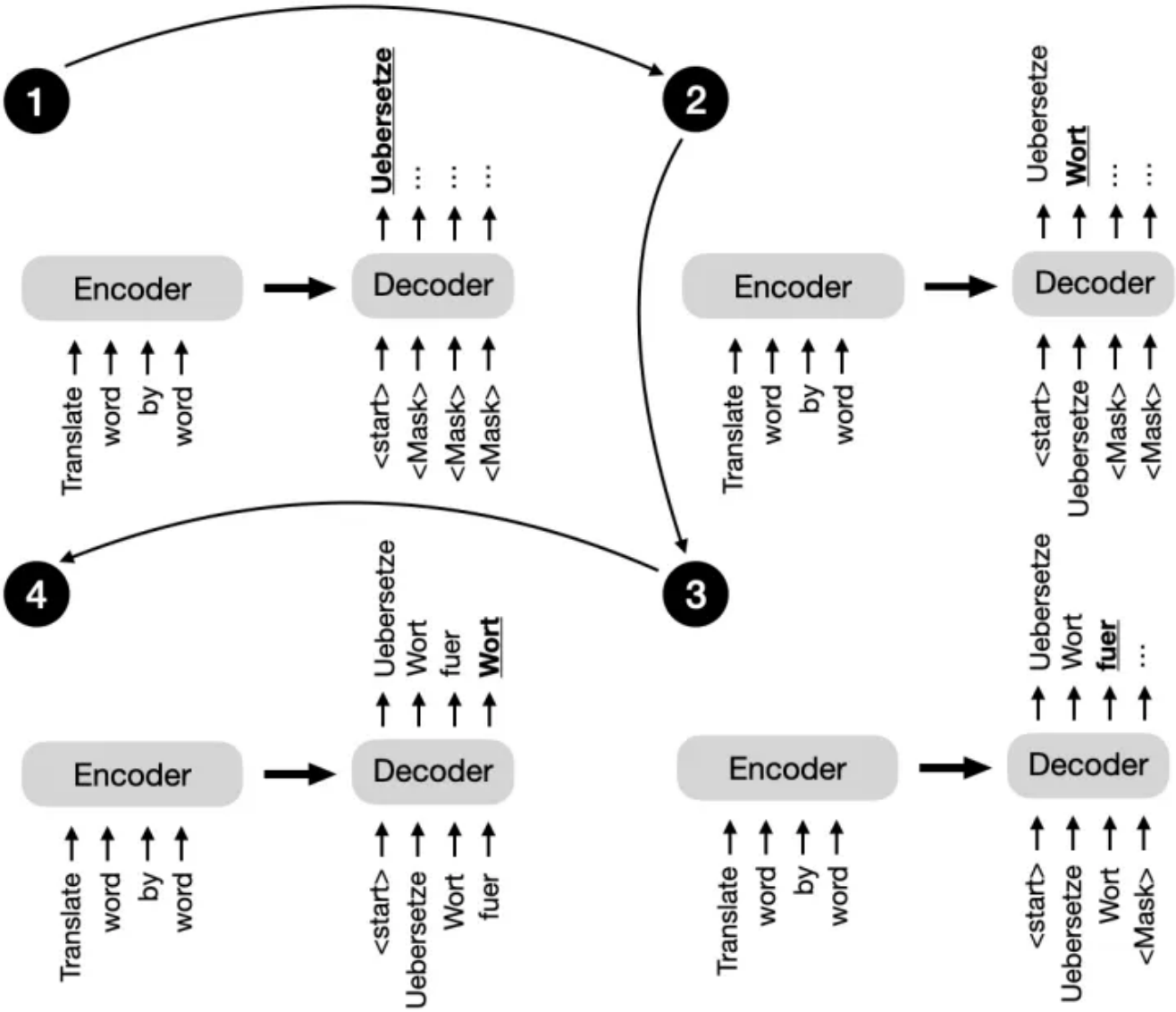
[CLS]标记是模型的占位符标记，提示模型返回一个True或False标签，表示句子是否按正确顺序排列。

掩码语言和下一句话预训练目标（这是自监督学习的一种形式，如第2章所讨论）允许BERT学习输入文本的丰富上下文表示，然后可以针对各种下游任务（如情感分析、问答和命名实体识别）进行微调。

RoBERTa（鲁棒优化的BERT方法）是BERT的优化版本。它保持了与BERT相同的整体架构，但采用了几项训练和优化改进，例如更大的批量大小、更多的训练数据，以及消除了下一句话预测任务。这些改变使RoBERTa在各种自然语言理解任务上的性能超越了BERT。

解码器

回到本节开头所述的原始Transformer架构，解码器中的多头自注意力机制与编码器中的类似，但它被掩盖以防止模型关注未来位置，确保位置i的预测只能依赖于i位置之前的已知输出。如下图所示，解码器逐字生成输出。



这种掩码（如上图所示，尽管它在解码器的多头自注意力机制中内部发生）对于维持变换器模型在训练和推理期间的自回归性质至关重要。自回归性质确保了模型一次生成一个输出标记，并使用先前生成的标记作为生成下一个词标记的上下文。

多年来，研究人员在原始的编码器–解码器变换器架构的基础上进行了改进，开发了多个仅包含解码器的模型，在各种自然语言处理任务中被证明非常有效。其中最著名的模型包括GPT系列。

GPT（生成式预训练变换器）系列是仅包含解码器的模型，它们在大规模无监督文本数据上进行预训练，然后针对特定任务（如文本分类、情感分析、问答和概括）进行微调。GPT模型，包括GPT–2、GPT–3（《GPT–3语言模型是少样本学习者》，2020）以及最近的GPT–4，在各种基准测试中表现出色，目前是自然语言处理领域最受欢迎的架构之一。

GPT模型最显著的特点之一是它们的出现性质。出现性质指的是模型由于其下一个词预测预训练而发展出的能力和技能。尽管这些模型只被教导预测下一个词，但预训练的模型能够进行文本概括、翻译、问答、分类等。此外，这些模型可以通过上下文学习执行新任务，而无需更新模型参数，这在第18章中有更详细的讨论。

编码器–解码器混合体

除了传统的编码器和解码器架构外，还有新的编码器–解码器模型的开发，利用了这两个组件的优势。这些模型通常结合了新技术、预训练目标或架构修改，以提高它们在各种自然语言处理任务中的性能。这些新编码器–解码器模型的一些著名例子包括：

- BART
- T5

编码器–解码器模型通常用于涉及理解输入序列和生成输出序列的自然语言处理任务，这些任务的长度和结构往往不同。它们特别适用于输入和输出序列之间存在复杂映射的任务，以及捕捉两个序列中元素之间关系至关重要的任务。编码器–解码器模型的一些常见用例包括文本翻译和概括。

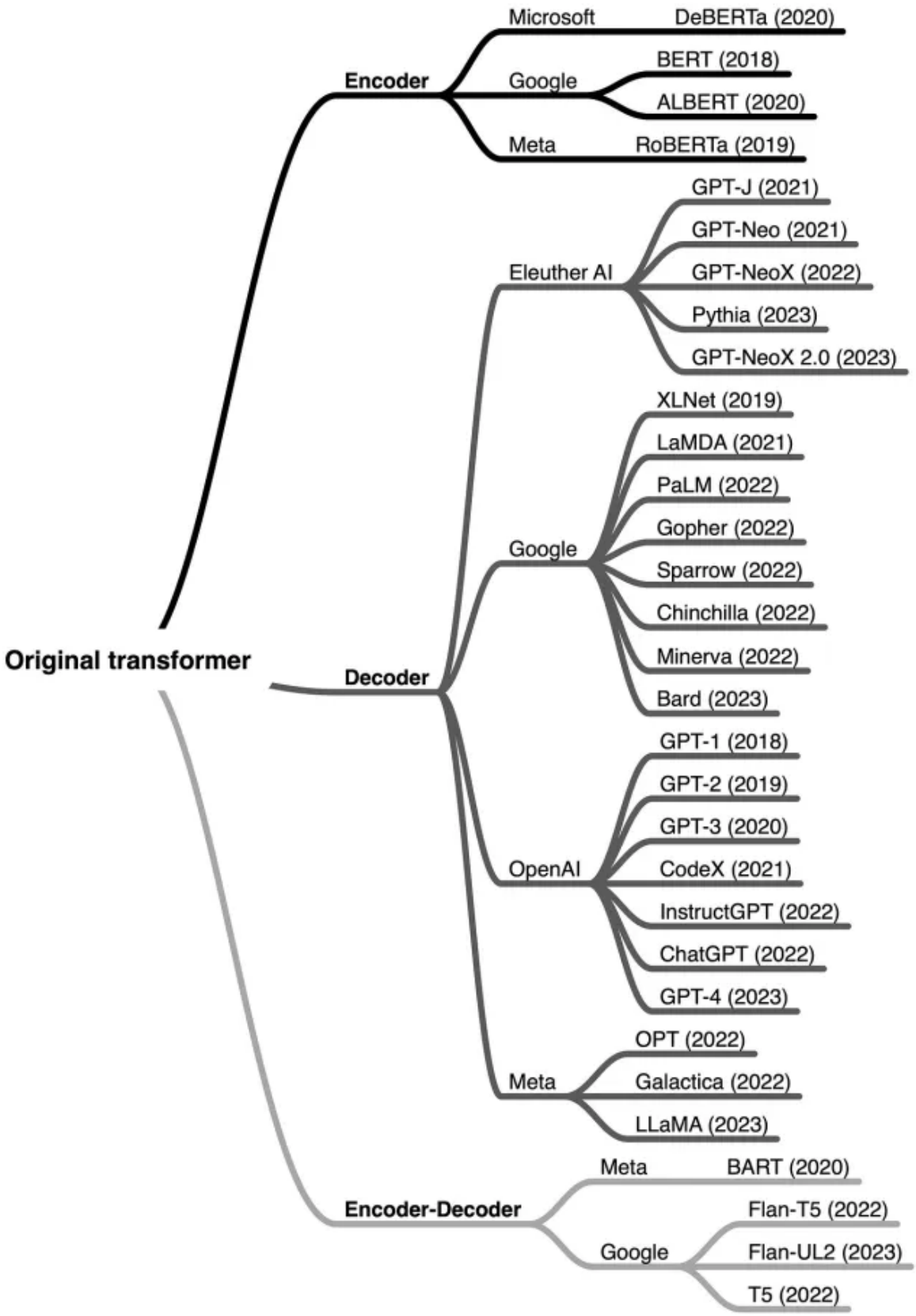
术语和行话

所有这些方法，无论是仅编码器、仅解码器还是编码器–解码器模型，都是序列到序列模型（通常缩写为seq2seq）。注意，虽然我们将BERT风格的方法称为仅编码器，但描述为仅编码器可能会产生误导，因为这些方法在预训练期间也将嵌入解码为输出标记或文本。

换句话说，无论是仅编码器还是仅解码器架构都在“解码”。然而，与仅解码器和编码器–解码器架构相比，仅编码器架构并不是以自回归方式解码。自回归解码指的是一次生成一个输出序列标记，每个标记都基于之前生成的标记。仅编码器模型并不以这种方式生成连贯的输出序列。相反，它们专注于理解输入文本并生成特定任务的输出，例如标签或标记预测。

结论

简而言之，编码器风格的模型在学习用于分类任务的嵌入方面很受欢迎，编码器–解码器风格的模型用于依赖输入的生成任务（例如，翻译和概括），而仅解码器模型用于包括问答在内的其他类型的生成任务。自从第一个变换器架构出现以来，已经开发了数以百计的仅编码器、仅解码器和编码器–解码器混合体，如下图所概述。


Original transformer
Encoder
Microsoft
DeBERTa (2020)
Google
BERT (2018)
ALBERT (2020)
Meta
RoBERTa (2019)
Decoder
Eleuther AI
GPT-J (2021)
GPT-Neo (2021)
GPT-NeoX (2022)
Pythia (2023)
GPT-NeoX 2.0 (2023)
Google
XLNet (2019)
LaMDA (2021)
PaLM (2022)
Gopher (2022)
Sparrow (2022)
Chinchilla (2022)
Minerva (2022)
Bard (2023)
OpenAI
GPT-1 (2018)
GPT-2 (2019)
GPT-3 (2020)
CodeX (2021)
InstructGPT (2022)
ChatGPT (2022)
GPT-4 (2023)
Meta
OPT (2022)
Galactica (2022)
LLaMA (2023)
Encoder-Decoder
Meta
BART (2020)
Google
Flan-T5 (2022)
Flan-UL2 (2023)
T5 (2022)

尽管仅编码器模型的受欢迎程度逐渐下降，但像GPT这样的仅解码器模型由于GPT-3、ChatGPT和GPT-4在文本生成方面的突破而迅速走红。然而，仅编码器模型在基于文本嵌入的预测模型训练方面仍然非常有用，相对于生成文本而言。

https://magazine.sebastianraschka.com/p/understanding-encoder-and-decoder