从零构建大模型 (Sebastian Raschka, 塞巴斯蒂安·拉施卡, 覃立波, 冯晓骋, .pdf

- 视频课
- 课本章节
  - Ch 1: 理解语言大模型
    - 什么是大语言模型
      - 大语言模型是一种用于理解、生成和响应类似人类语言文本的神经网络。这类模型通常拥有数百亿甚至数千亿个参数。由于大语言模型能够生成文本，因此它们通常也被归类为生成式人工智能。

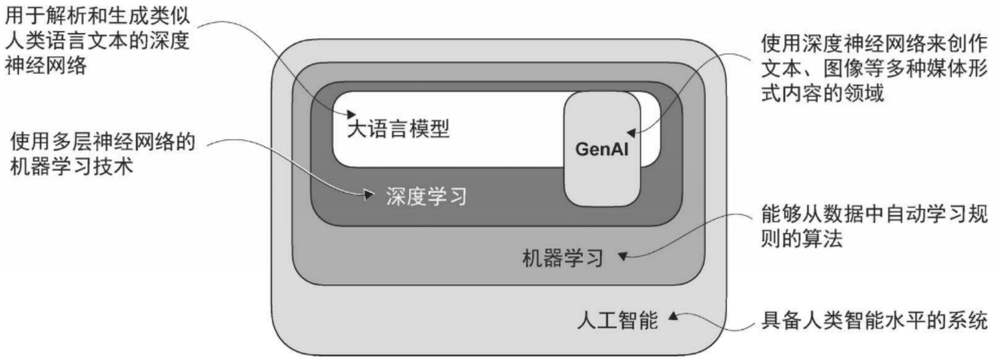


图 1-1　这一层级关系图展示了不同领域之间的关系。大语言模型是深度学习技术的具体应用，能够处理和生成类似人类语言的文本；深度学习是机器学习的一个分支，主要使用多层神经网络；机器学习和深度学习致力于开发算法，使计算机能够从数据中学习，并执行需要人类智能水平的任务

    - 大语言模型的应用
      - 大语言模型已被应用于机器翻译、文本生成、情感分析、文本摘要等多种任务，此外还可以为复杂的聊天机器人和虚拟助手提供支持，还被用于从大量文本中有效地提取知识。
    - 构建和使用大语言模型的各个阶段
      - 大语言模型的构建通常包括预训练（pre-training）和微调（fine-tuning）两个阶段。
        - 预训练阶段：预训练是大语言模型的第一个训练阶段，预训练后的大语言模型通常称为基础模型。
        - 微调阶段：以预训练模型为基础，在规模较小的特定任务或领域数据集上对模型进行针对性训练，以进一步提升其特定能力。
          - 微调大语言模型最流行的两种方法是指令微调和分类任务微调。
            - 指令微调：标注数据集由“指令-答案”对（比如翻译任务中的“原文-正确翻译文本”）组成。
            - 分类任务微调：标注数据集由文本及其类别标签组成。
    - Transformer架构

- 由两个子模块构成：编码器和解码器。

  原始编码器风格的 Transformer，即 BERT专注于掩码预测，在情感预测、文档分类等文本分类任务中具有优势（BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding）

  解码器风格的 GPT-3，主要用于处理生成文本的任务（Language Models are Few-Shot Learners）

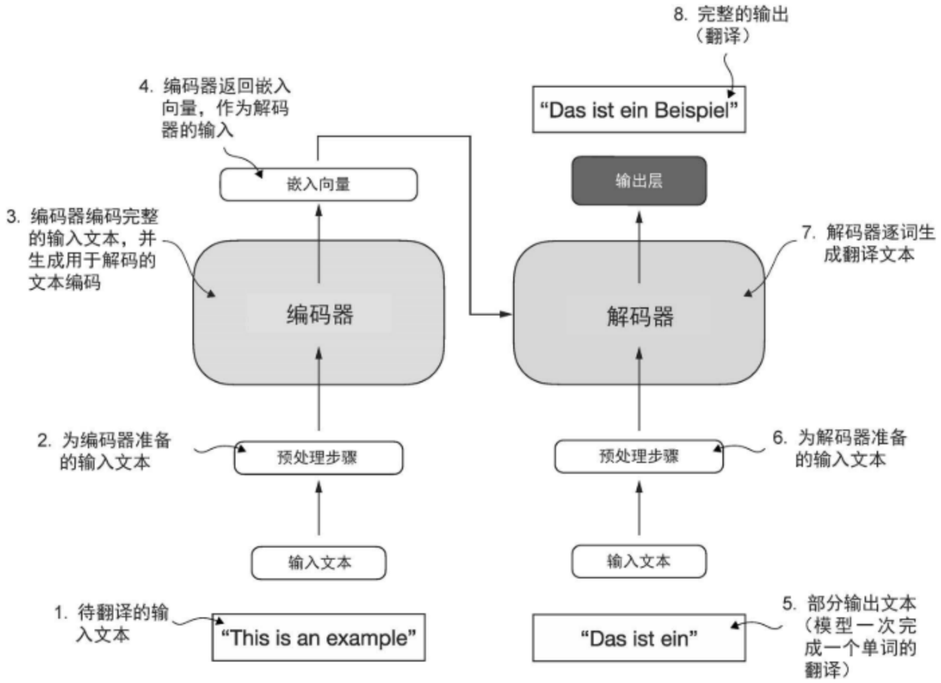


图 1-4　原始 Transformer 架构的简化描述，这是一种用于机器翻译的深度学习模型。Transformer 由两部分组成：一个是编码器，用于处理输入文本并生成文本嵌入（一种能够在不同维度中捕获许多不同因素的数值表示）；另一个是解码器，用于使用这些文本嵌入逐词生成翻译后的文本。请注意，图中展示的是翻译过程的最后阶段，此时解码器根据原始输入文本（"This is an example"）和部分翻译的句子（"Das ist ein"），生成最后一个单词（"Beispiel"）以完成翻译

- **大型数据集**
  - 训练数据集的庞大规模和丰富多样性使得这些模型在包括语言语法、语义、上下文，甚至一些需要通用知识的任务上都拥有了良好表现。
- **GPT架构**
  - 由于像 GPT 这样的解码器模型是通过逐词预测生成文本，因此它们被认为是一种自回归模型

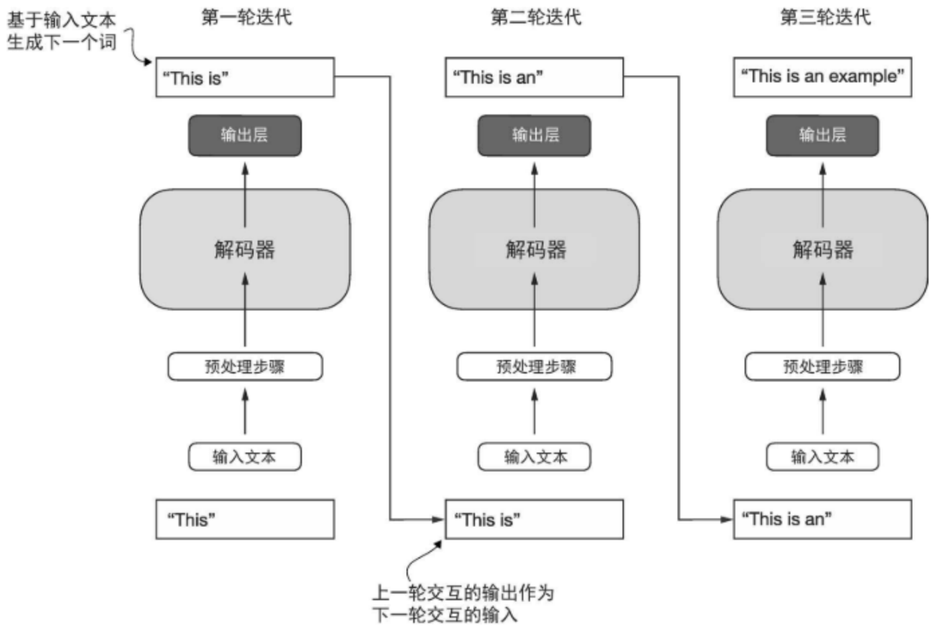


图 1-8　GPT 架构仅使用原始的 Transformer 解码器部分。它被设计为单向的从左到右处理，这使得它非常适合文本生成和下一单词预测任务，可以逐个词地迭代生成文本

- **构建大语言模型**
  - 三个阶段

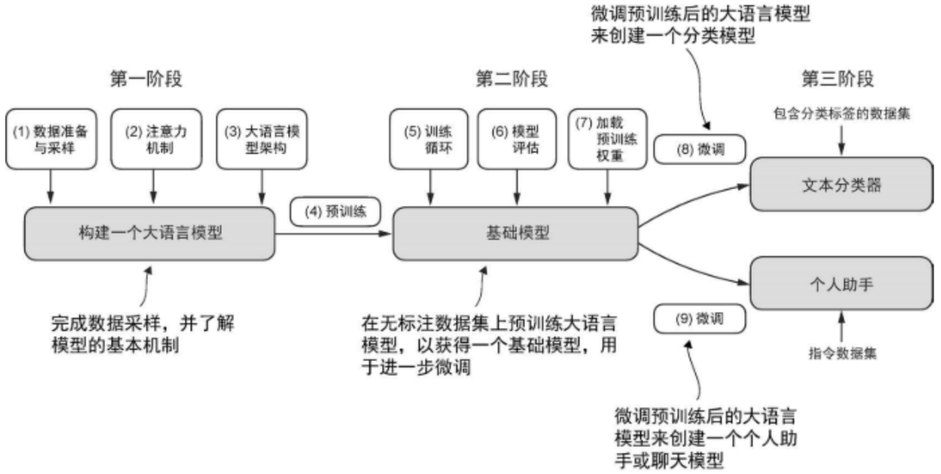


图 1-9　构建大语言模型的 3 个主要阶段：实现模型架构和准备数据（第一阶段）、预训练大语言模型以获得基础模型（第二阶段），以及微调基础模型以得到个人助手或文本分类器（第三阶段）

- **Ch 2: 处理文本数据**
  - **理解词嵌入**
    - 嵌入的本质是将离散对象（如单词、图像甚至整个文档）映射到连续向量空间中的点，其主要目的是将非数值的数据转换为神经网络可以处理的格式。
    - 词嵌入的维度（dimension）可以从一维到数千维不等。更高的维度有助于捕捉到更细微的关系，但这通常以牺牲计算效率为代价。
  - **文本分词**
    - 词元既可以是单个单词，也可以是包括标点符号在内的特殊字符

      样例见jupyter 第二章 2.2 文本分词样例

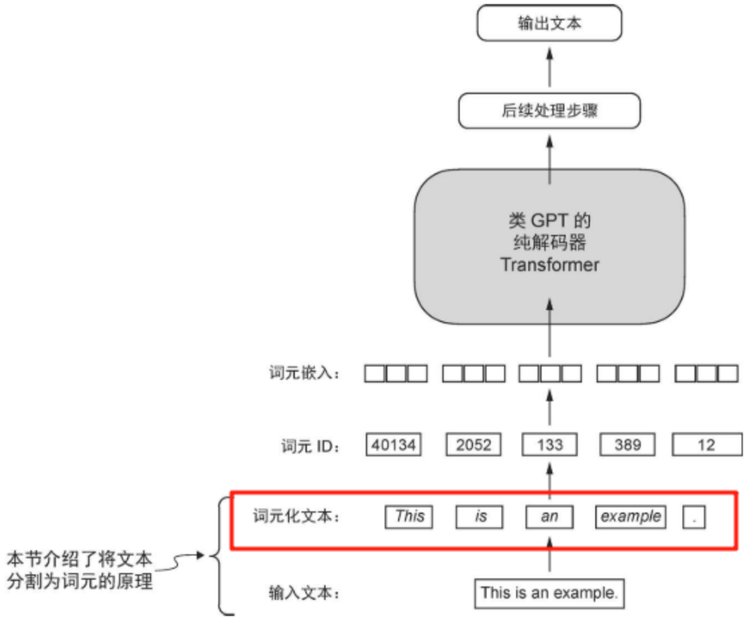
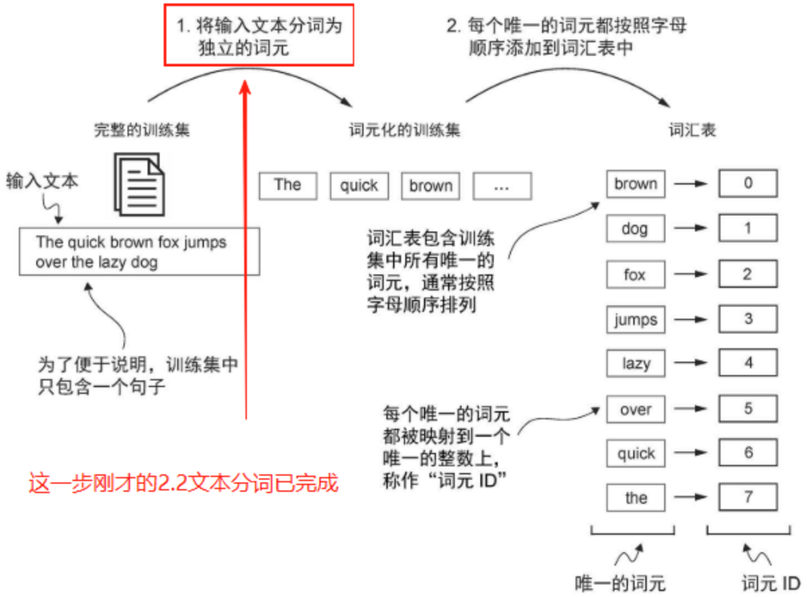


图 2-4　大语言模型中文本处理步骤概览。在这一步骤图中，我们将输入文本分割成了单独的词元，这些词元既可以是单词，也可以是诸如标点符号之类的特殊字符

- **将词元转换为词元ID**
  - 把这些词元从 Python字符串转换为整数表示，以生成词元 ID

    样例见jupyter 第二章 2.3 将词元转换为词元ID



- **引入特殊上下文词元**
  - 2.3节的内容中，未收录的未知单词会在处理时出现错误，为了解决该问题需要对分词器进行必要的修改，通过引入特殊上下文词元可以增强模型对上下文和其他相关信息的理解。主要是两个新词元<|unk|>和<|endoftext|>，此外还有其他特殊词元如表。
    样例见jupyter 第二章 2.4引入特殊上下文词元

在不同的大语言模型中，研究人员可能会考虑引入如下这些特殊词元。

- [BOS]（**序列开始**）：标记文本的起点，告知大语言模型一段内容的开始。
- [EOS]（**序列结束**）：位于文本的末尾，类似<|endoftext|>，特别适用于连接多个不相关的文本。例如，在合并两篇不同的维基百科文章（或两本不同的图书）时，[EOS]词元指示一篇文章的结束和下一篇文章的开始。
- [PAD]（**填充**）：当使用**批次大小**（batch size）大于 1 的批量数据训练大语言模型时，数据中的文本长度可能不同。为了使所有文本具有相同的长度，较短的文本会通过添加[PAD]词元进行扩展或“填充”，以匹配批量数据中的最长文本的长度。

  - <|unk|>（Unknown，未知词元）：解决未收录词汇问题，即当分词器遇到未出现在词汇表（vocab）中的未知单词时，用<|unk|> 替代，避免因找不到对应 ID 而报错。
    - 任何词汇表都是基于有限语料构建的，实际应用中必然会遇到生词（如新词、拼写错误、专有名词等）。如果没有<|unk|>，分词器会抛出 KeyError（找不到对应 ID），导致程序中断。

- <|endoftext|>（End of Text，文本结束词元）：标记文本边界，即明确标记一段文本的结束位置，帮助模型区分不同文本的边界。<|endoftext|>词元与[EOS]词元作用相似。此外，<|endoftext|>也被用于文本的填充。
  - 在处理多篇文本（如批量训练、多轮对话）时，模型需要知道“当前文本何时结束”，避免将不同文本的内容混淆。

- **BPE**
  - BPE算法的原理是将不在预定义词汇表中的单词分解为更小的子词单元甚至单个字符，从而能够处理词汇表之外的单词。

    样例见jupyter 第二章 2.5 一种基于 BPE 概念的更复杂的分词方案

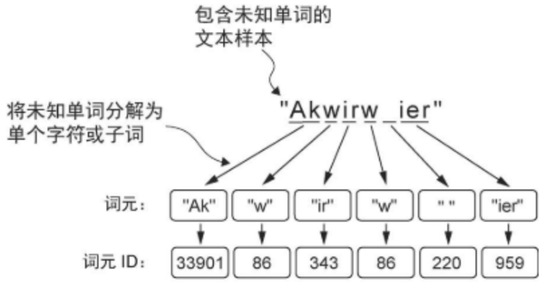


图 2-11　BPE 分词器会将未知单词分解为子词和单个字符。如此一来，BPE 分词器便可以解析任何单词，而无须使用特殊词元（如<|unk|>）来替换未知单词

- **使用滑动窗口进行数据采样**
  - 使用滑动窗口（Sliding Window）进行数据采样是一种常用技术，用于将长文本分割成固定长度的样本序列。这种方法能有效处理超出模型最大序列长度的文本，并通过控制窗口移动步长（stride）来平衡样本数量和数据利用率。

    样例见jupyter 第二章 2.6 使用滑动窗口进行数据采样

- **创建词元嵌入**
  - 为大语言模型训练准备输入文本的最后一步是将词元 ID转换为嵌入向量

    样例见jupyter 第二章 2.7 词元 ID转换为嵌入向量的工作原理

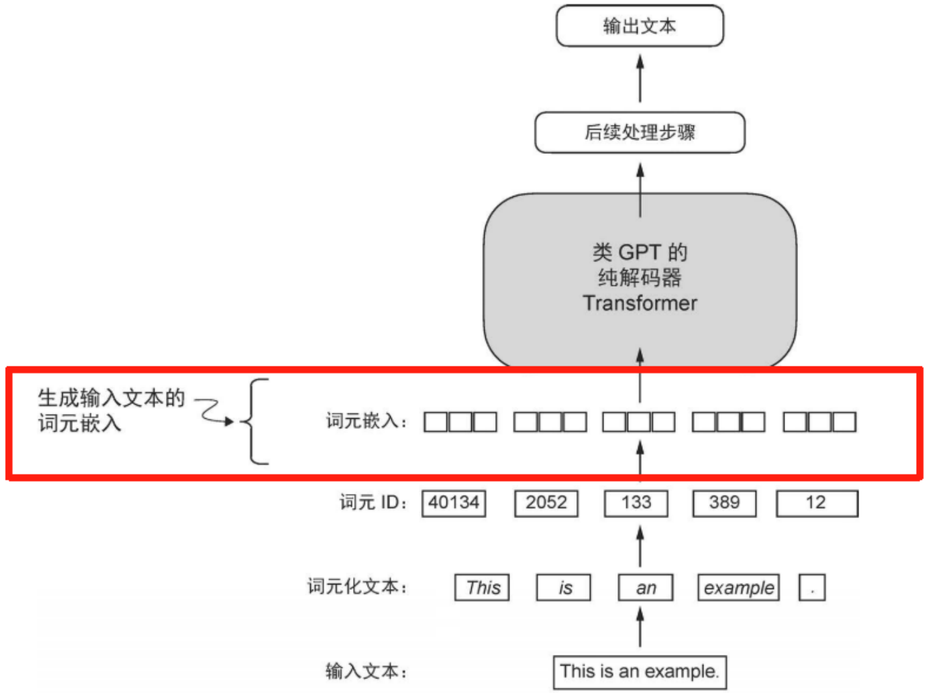


图 2-15　大语言模型的输入文本的准备工作包括文本分词、将词元转换为词元 ID，以及将词元 ID 转换为嵌入向量。本节将利用此前生成的词元 ID 来创建词元嵌入向量

- **编码单词位置信息**
  - 嵌入层的工作机制是，无论词元 ID在输入序列中的位置如何，相同的词元 ID始终被映射到相同的向量表示。由于大语言模型的自注意力机制本质上与位置无关，因此向模型中注入额外的位置信息是有帮助的。

    两种位置信息嵌入策略：绝对位置嵌入和相对位置嵌入

    样例见jupyter 第二章 2.8 创建初始的位置嵌入

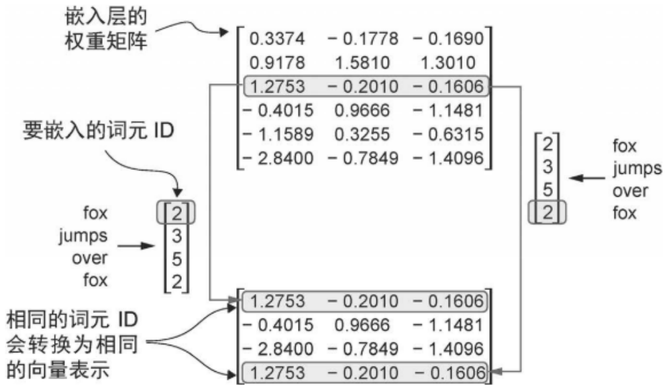


图 2-17　嵌入层始终将相同的词元 ID 转换为相同的向量表示，不受其在输入序列中的位置的影响。例如，无论词元 ID 为 5 的词元出现在输入向量的第一个位置还是第四个位置，它都将被映射为相同的嵌入向量

- 绝对位置嵌入：直接与序列中的特定位置相关联。对于输入序列的每个位置，该方法都会向对应词元的嵌入向量中添加一个独特的位置嵌入，以明确指示其在序列中的确切位置。
- 相对位置嵌入：关注的是词元之间的相对位置或距离，而非它们的绝对位置。这意味着模型学习的是词元之间的“距离”关系，而不是它们在序列中的“具体位置”

- Ch 3: 编码注意力机制
- Ch 4: 从头实现GPT模型进行文本生成
- Ch 5: 在无标签数据上进行预训练
- Ch 6: 针对分类的微调
- Ch 7: 通过指令遵循人类指令