1 Understanding large language models

This chapter covers

- High-level explanations of the fundamental concepts behind large language models (LLMs)
- Insights into the transformer architecture from which LLMs are derived
- A plan for building an LLM from scratch

Large language models (LLMs), such as those offered in OpenAI's ChatGPT, are deep neural network models that have been developed over the past few years. They ushered in a new era for natural language processing (NLP). Before the advent of LLMs, traditional methods excelled at categorization tasks such as email spam classification and straightforward pattern recognition that could be captured with handcrafted rules or simpler models. However, they typically underperformed in language tasks that demanded complex understanding and generation abilities, such as parsing detailed instructions, conducting contextual analysis, and creating coherent and contextually appropriate original text. For example, previous generations of language models could not write an email from a list of keywords—a task that is trivial for contemporary LLMs.

1

理解大型语言模型

本章涵盖

- 高级语言模型背后的基本概念解释（LLMs）
- 洞察到从中提取LLMs的 Transformer 架构
- 一个从头开始构建LLM的计划

大型语言模型（LLMs），如 OpenAI 的 ChatGPT 所提供的，是过去几年中开发的深度神经网络模型。它们为自然语言处理（NLP）带来了新时代。在LLMs出现之前，传统方法在诸如电子邮件垃圾邮件分类和可以用手工规则或简单模型捕获的直观模式识别等分类任务上表现出色。然而，它们通常在需要复杂理解和生成能力的语言任务上表现不佳，例如解析详细指令、进行上下文分析和创建连贯且符合上下文的原创文本。例如，上一代语言模型无法根据关键词列表撰写电子邮件——对于当代LLMs来说，这是一个微不足道的小任务。

LLMs have remarkable capabilities to understand, generate, and interpret human language. However, it's important to clarify that when we say language models "understand," we mean that they can process and generate text in ways that appear coherent and contextually relevant, not that they possess human-like consciousness or comprehension.

Enabled by advancements in deep learning, which is a subset of machine learning and artificial intelligence (AI) focused on neural networks, LLMs are trained on vast quantities of text data. This large-scale training allows LLMs to capture deeper contextual information and subtleties of human language compared to previous approaches. As a result, LLMs have significantly improved performance in a wide range of NLP tasks, including text translation, sentiment analysis, question answering, and many more.

Another important distinction between contemporary LLMs and earlier NLP models is that earlier NLP models were typically designed for specific tasks, such as text categorization, language translation, etc. While those earlier NLP models excelled in their narrow applications, LLMs demonstrate a broader proficiency across a wide range of NLP tasks.

The success behind LLMs can be attributed to the transformer architecture that underpins many LLMs and the vast amounts of data on which LLMs are trained, allowing them to capture a wide variety of linguistic nuances, contexts, and patterns that would be challenging to encode manually.

This shift toward implementing models based on the transformer architecture and using large training datasets to train LLMs has fundamentally transformed NLP, providing more capable tools for understanding and interacting with human language.

The following discussion sets a foundation to accomplish the primary objective of this book: understanding LLMs by implementing a ChatGPT-like LLM based on the transformer architecture step by step in code.

1.1 What is an LLM?

An LLM is a neural network designed to understand, generate, and respond to human-like text. These models are deep neural networks trained on massive amounts of text data, sometimes encompassing large portions of the entire publicly available text on the internet.

The "large" in "large language model" refers to both the model's size in terms of parameters and the immense dataset on which it's trained. Models like this often have tens or even hundreds of billions of parameters, which are the adjustable weights in the network that are optimized during training to predict the next word in a sequence. Next-word prediction is sensible because it harnesses the inherent sequential nature of language to train models on understanding context, structure, and relationships within text. Yet, it is a very simple task, and so it is surprising to many researchers that it can produce such capable models. In later chapters, we will discuss and implement the next-word training procedure step by step.

LLMs 具有理解、生成和解释人类语言的显著能力。然而，重要的是要明确，当我们说语言模型“理解”时，我们是指它们可以以看似连贯和上下文相关的方式处理和生成文本，而不是它们拥有类似人类的意识或理解。

得益于深度学习的发展，深度学习是机器学习和人工智能（AI）的一个子集，专注于神经网络，LLMs 在大量文本数据上进行训练。这种大规模训练使LLMs 相比以往的方法能够捕捉更深层次的语境信息和人类语言的微妙之处。因此，LLMs 在包括文本翻译、情感分析、问答等在内的广泛 NLP 任务中的性能得到了显著提升。

当代LLMs与早期 NLP 模型之间的重要区别在于，早期 NLP 模型通常是为特定任务设计的，例如文本分类、语言翻译等。虽然那些早期 NLP 模型在其狭窄的应用中表现出色，但LLMs在广泛的 NLP 任务中展现出更广泛的熟练度。

LLMs的成功归因于支撑许多LLMs的 Transformer 架构以及大量用于训练LLMs的数据，这使得它们能够捕捉到广泛的语言细微差别、语境和模式，这些内容手动编码将极具挑战性。

这一转向基于 Transformer 架构实施模型和使用大型训练数据集训练LLMs，从根本上改变了自然语言处理，为理解和交互人类语言提供了更强大的工具。

以下讨论为理解本书的主要目标奠定基础：通过逐步在代码中实现基于 Transformer 架构的类似 ChatGPT 的LLM来实施LLMs。

1.1 什么是LLM?

一个LLM是一种神经网络，旨在理解、生成和回应类似人类的文本。这些模型是经过大量文本数据训练的深度神经网络，有时甚至包括整个互联网上可公开获取文本的大部分内容。

“大型”在“大型语言模型”中既指模型的参数规模，也指其训练所使用的海量数据集。这类模型通常拥有数十亿甚至数百亿个参数，这些参数是网络中的可调整权重，在训练过程中被优化以预测序列中的下一个单词。预测下一个单词是有意义的，因为它利用了语言的内在序列性质，以训练模型理解文本中的上下文、结构和关系。然而，这实际上是一个非常简单的任务，因此许多研究人员对此能够产生如此强大的模型感到惊讶。在后面的章节中，我们将逐步讨论和实现下一个单词的训练过程。

LLMs utilize an architecture called the *transformer*, which allows them to pay selective attention to different parts of the input when making predictions, making them especially adept at handling the nuances and complexities of human language.

Since LLMs are capable of *generating* text, LLMs are also often referred to as a form of generative artificial intelligence, often abbreviated as *generative AI* or *GenAI*. As illustrated in figure 1.1, AI encompasses the broader field of creating machines that can perform tasks requiring human-like intelligence, including understanding language, recognizing patterns, and making decisions, and includes subfields like machine learning and deep learning.

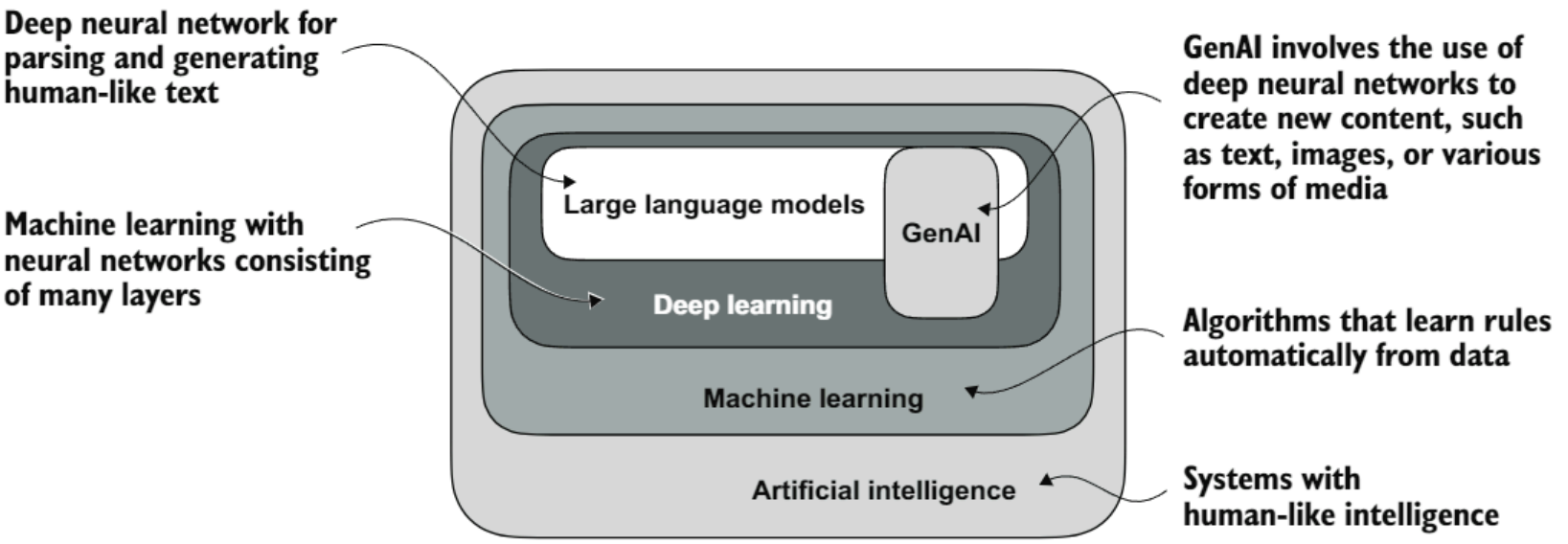

Figure 1.1 As this hierarchical depiction of the relationship between the different fields suggests, LLMs represent a specific application of deep learning techniques, using their ability to process and generate human-like text. Deep learning is a specialized branch of machine learning that focuses on using multilayer neural networks. Machine learning and deep learning are fields aimed at implementing algorithms that enable computers to learn from data and perform tasks that typically require human intelligence.

The algorithms used to implement AI are the focus of the field of machine learning. Specifically, machine learning involves the development of algorithms that can learn from and make predictions or decisions based on data without being explicitly programmed. To illustrate this, imagine a spam filter as a practical application of machine learning. Instead of manually writing rules to identify spam emails, a machine learning algorithm is fed examples of emails labeled as spam and legitimate emails. By minimizing the error in its predictions on a training dataset, the model then learns to recognize patterns and characteristics indicative of spam, enabling it to classify new emails as either spam or not spam.

As illustrated in figure 1.1, deep learning is a subset of machine learning that focuses on utilizing neural networks with three or more layers (also called deep neural networks) to model complex patterns and abstractions in data. In contrast to deep learning, traditional machine learning requires manual feature extraction. This means that human experts need to identify and select the most relevant features for the model.

利用名为 Transformer 的架构，这使得它们在预测时能够对输入的不同部分进行选择性关注，使它们特别擅长处理人类语言的细微差别和复杂性。

由于LLMs能够生成文本，LLMs也常被称为生成式人工智能，通常简称为生成 AI 或 GenAI。如图 1.1 所示，人工智能涵盖了更广泛的领域，即创建能够执行需要类似人类智能的任务的机器，包括理解语言、识别模式和做出决策，并包括机器学习和深度学习等子领域。

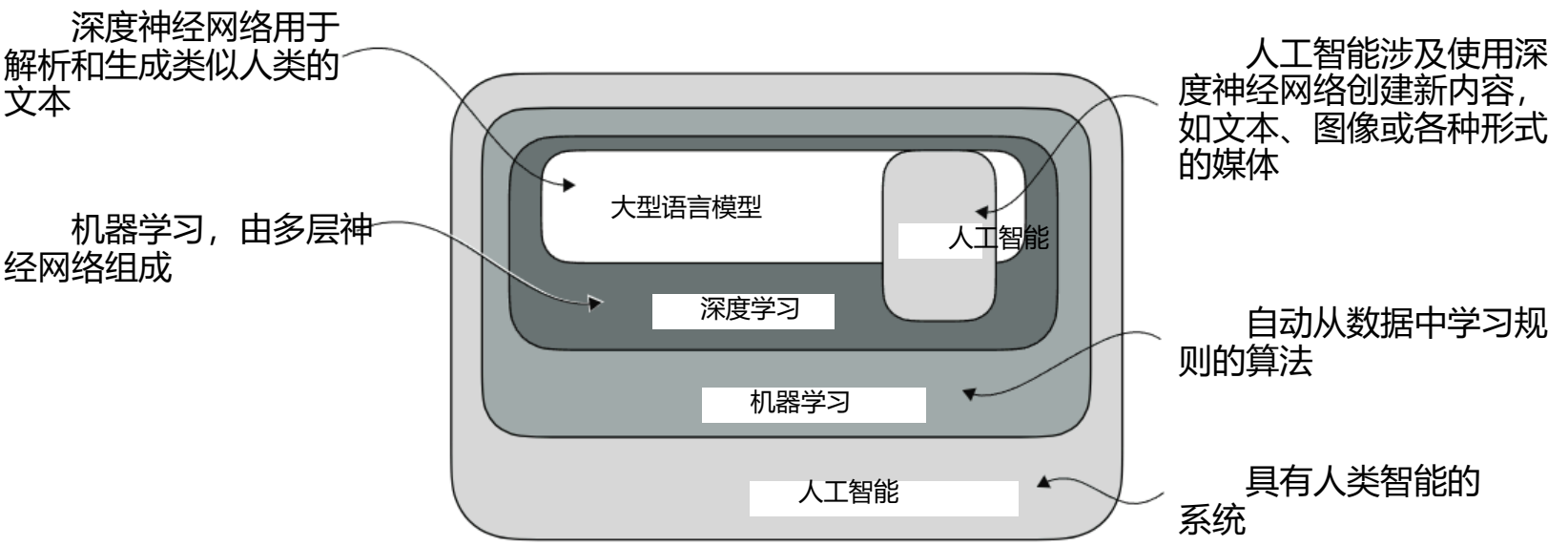

图 1.1 如此分层展示不同领域之间关系所示，LLMs代表深度学习技术的特定应用，利用其处理和生成类似人类文本的能力。深度学习是机器学习的一个专门分支，专注于使用多层神经网络。机器学习和深度学习是旨在实现算法的领域，这些算法使计算机能够从数据中学习并执行通常需要人类智能的任务。

人工智能实现的算法是机器学习领域的焦点。具体来说，机器学习包括开发能够从数据中学习并基于数据做出预测或决策的算法，而不需要明确编程。为了说明这一点，想象一下垃圾邮件过滤器作为机器学习的一个实际应用。不是手动编写规则来识别垃圾邮件，而是将标记为垃圾邮件和合法邮件的电子邮件示例输入机器学习算法。通过最小化其在训练数据集上的预测错误，模型随后学会识别表明垃圾邮件的模式和特征，使其能够将新电子邮件分类为垃圾邮件或非垃圾邮件。

如图 1.1 所示，深度学习是机器学习的一个子集，它侧重于利用具有三层或更多层的神经网络（也称为深度神经网络）来模拟数据中的复杂模式和抽象。与传统深度学习相比，传统机器学习需要手动提取特征。这意味着人类专家需要识别和选择对模型最相关的特征。

While the field of AI is now dominated by machine learning and deep learning, it also includes other approaches—for example, using rule-based systems, genetic algorithms, expert systems, fuzzy logic, or symbolic reasoning.

Returning to the spam classification example, in traditional machine learning, human experts might manually extract features from email text such as the frequency of certain trigger words (for example, "prize," "win," "free"), the number of exclamation marks, use of all uppercase words, or the presence of suspicious links. This dataset, created based on these expert-defined features, would then be used to train the model. In contrast to traditional machine learning, deep learning does not require manual feature extraction. This means that human experts do not need to identify and select the most relevant features for a deep learning model. (However, both traditional machine learning and deep learning for spam classification still require the collection of labels, such as spam or non-spam, which need to be gathered either by an expert or users.)

Let's look at some of the problems LLMs can solve today, the challenges that LLMs address, and the general LLM architecture we will implement later.

1.2 Applications of LLMs

Owing to their advanced capabilities to parse and understand unstructured text data, LLMs have a broad range of applications across various domains. Today, LLMs are employed for machine translation, generation of novel texts (see figure 1.2), sentiment analysis, text summarization, and many other tasks. LLMs have recently been used for content creation, such as writing fiction, articles, and even computer code.

LLMs can also power sophisticated chatbots and virtual assistants, such as OpenAI's ChatGPT or Google's Gemini (formerly called Bard), which can answer user queries and augment traditional search engines such as Google Search or Microsoft Bing.

Moreover, LLMs may be used for effective knowledge retrieval from vast volumes of text in specialized areas such as medicine or law. This includes sifting through documents, summarizing lengthy passages, and answering technical questions.

In short, LLMs are invaluable for automating almost any task that involves parsing and generating text. Their applications are virtually endless, and as we continue to innovate and explore new ways to use these models, it's clear that LLMs have the potential to redefine our relationship with technology, making it more conversational, intuitive, and accessible.

We will focus on understanding how LLMs work from the ground up, coding an LLM that can generate texts. You will also learn about techniques that allow LLMs to carry out queries, ranging from answering questions to summarizing text, translating text into different languages, and more. In other words, you will learn how complex LLM assistants such as ChatGPT work by building one step by step.

人工智能领域目前由机器学习和深度学习主导，但也包括其他方法——例如，使用基于规则的系统、遗传算法、专家系统、模糊逻辑或符号推理。

回到垃圾邮件分类的例子，在传统机器学习中，人类专家可能会手动从电子邮件文本中提取特征，例如某些触发词（例如，“奖品”、“赢”、“免费”）的频率，感叹号的数量，使用全部大写字母的单词，或可疑链接的存在。这个数据集是基于这些专家定义的特征创建的，然后用于训练模型。与传统的机器学习相比，深度学习不需要手动特征提取。这意味着人类专家不需要为深度学习模型识别和选择最相关的特征。（然而，传统机器学习和用于垃圾邮件分类的深度学习仍然需要收集标签，如垃圾邮件或非垃圾邮件，这些标签需要由专家或用户收集。）

让我们看看LLMs今天可以解决的问题，LLMs所面临的挑战，以及我们稍后将要实施的通用LLM架构。

1.2 应用 LLMs

由于它们解析和理解非结构化文本数据的高级能力，LLMs 在各个领域都有广泛的应用。如今，LLMs 被用于机器翻译、生成新颖文本（见图 1.2）、情感分析、文本摘要以及许多其他任务。LLMs 最近被用于内容创作，如撰写小说、文章，甚至计算机代码。

LLMs 也能为复杂的聊天机器人和虚拟助手提供动力，例如 OpenAI 的 ChatGPT 或 Google 的 Gemini（之前称为 Bard），它们可以回答用户查询并增强传统搜索引擎，如 Google 搜索或 Microsoft Bing。

此外，LLMs可用于从大量医学或法律等特定领域的文本中有效检索知识。这包括筛选文档、总结长篇段落和回答技术问题。

简而言之，LLMs 对于自动化几乎任何涉及解析和生成文本的任务都极为宝贵。它们的应用几乎无限，随着我们继续创新和探索这些模型的新用途，很明显，LLMs 有潜力重新定义我们与技术的关系，使其更加对话式、直观和易于访问。

我们将从底层开始关注理解LLMs的工作原理，编写一个可以生成文本的LLM。你还将了解允许LLMs执行查询的技术，这些技术包括回答问题、总结文本、将文本翻译成不同语言等。换句话说，你将通过逐步构建来学习如何复杂LLM助手（如 ChatGPT）工作。

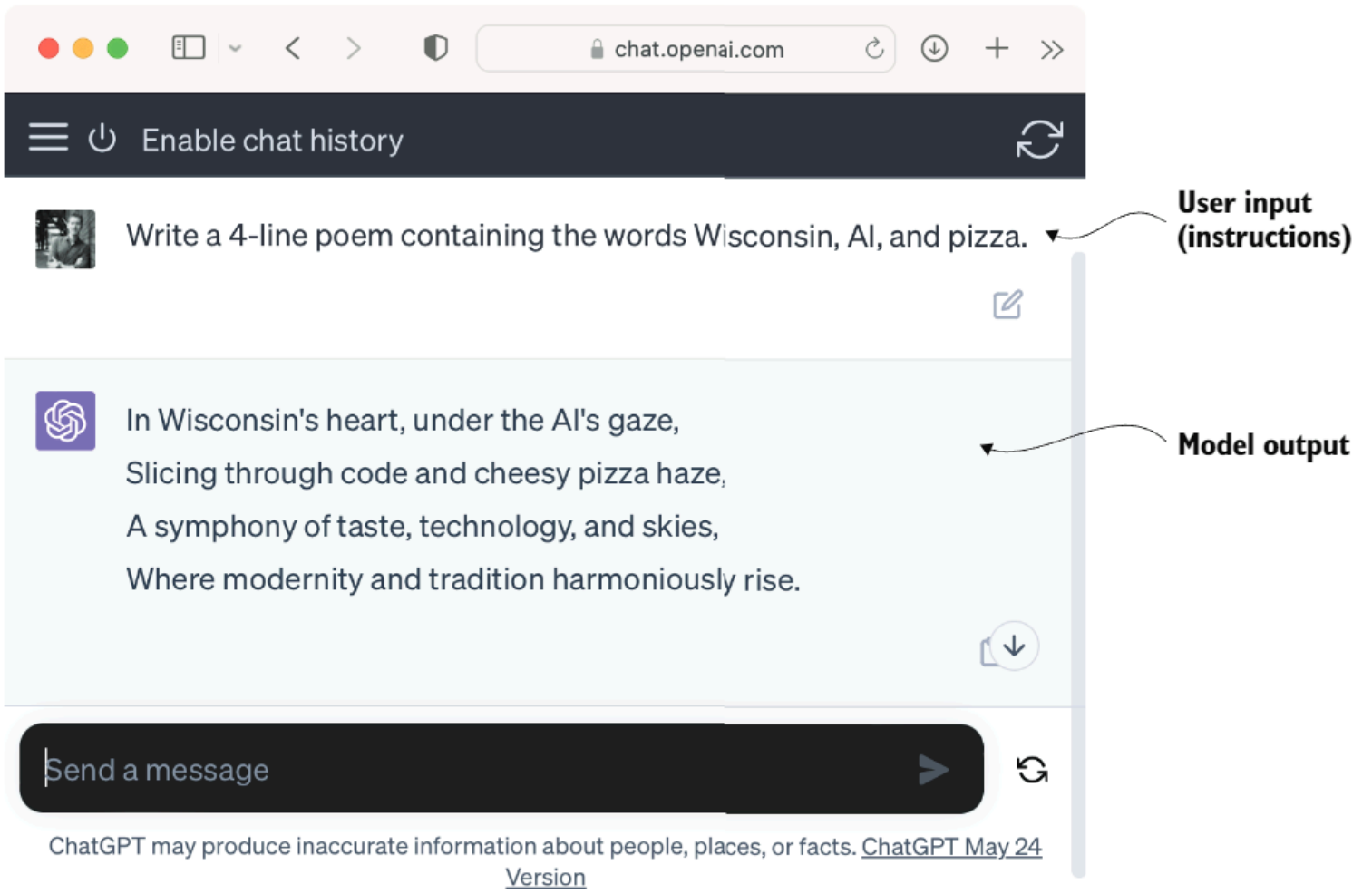

Figure 1.2 LLM interfaces enable natural language communication between users and AI systems. This screenshot shows ChatGPT writing a poem according to a user's specifications.

1.3 Stages of building and using LLMs

Why should we build our own LLMs? Coding an LLM from the ground up is an excellent exercise to understand its mechanics and limitations. Also, it equips us with the required knowledge for pretraining or fine-tuning existing open source LLM architectures to our own domain-specific datasets or tasks.

> **NOTE** Most LLMs today are implemented using the PyTorch deep learning library, which is what we will use. Readers can find a comprehensive introduction to PyTorch in appendix A.

Research has shown that when it comes to modeling performance, custom-built LLMs—those tailored for specific tasks or domains—can outperform general-purpose LLMs, such as those provided by ChatGPT, which are designed for a wide array of applications. Examples of these include BloombergGPT (specialized for finance) and LLMs tailored for medical question answering (see appendix B for more details).

Using custom-built LLMs offers several advantages, particularly regarding data privacy. For instance, companies may prefer not to share sensitive data with third-party LLM providers like OpenAI due to confidentiality concerns. Additionally, developing smaller custom LLMs enables deployment directly on customer devices, such as laptops and smartphones, which is something companies like Apple are currently exploring.

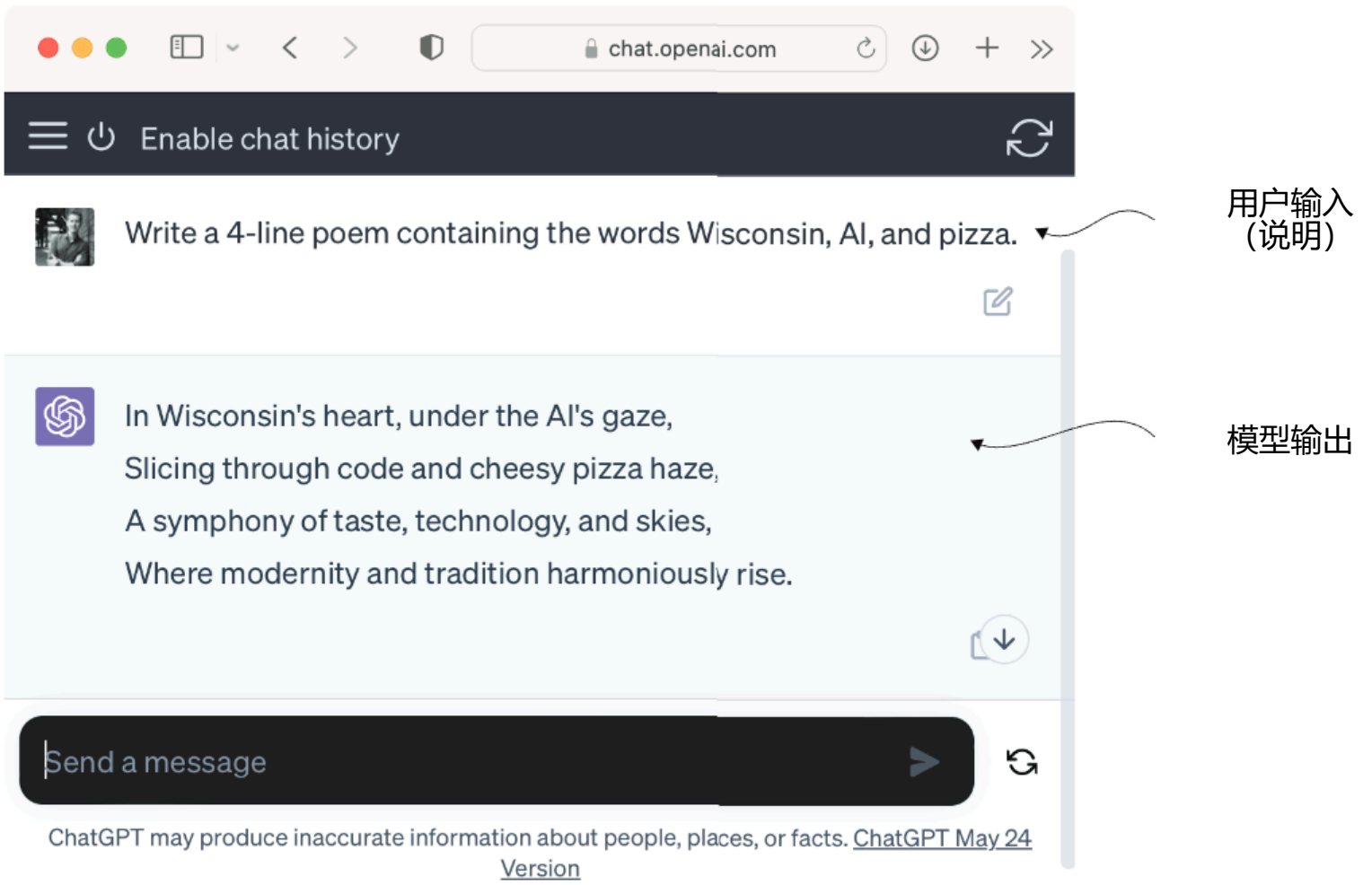

图 1.2 LLM 接口使用户与 AI 系统之间实现自然语言交流。此截图显示 ChatGPT 根据用户要求创作一首诗。

1.3 建筑和使用 LLMs 阶段

为什么我们要构建自己的LLMs？从头开始编写LLM是一个理解其机制和限制的绝佳练习。此外，它使我们具备了对现有开源LLM架构进行预训练或微调到我们自己的特定领域数据集或任务所需的知识。

注意：大多数LLMs今天都是使用 PyTorch 深度学习库实现的，这就是我们将要使用的。读者可以在附录 A 中找到 PyTorch 的全面介绍。

研究表明，在建模性能方面，定制的LLMs——针对特定任务或领域量身定制的——可以优于通用型LLMs，例如 ChatGPT 提供的，这些是为广泛的应用而设计的。这些例子包括专门用于金融的 BloombergGPT 和针对医疗问答定制的LLMs（更多详情见附录 B）。

使用定制的LLMs提供多个优势，尤其是在数据隐私方面。例如，公司可能由于保密问题而更愿意不与第三方LLM提供商如 OpenAI 共享敏感数据。此外，开发更小的定制LLMs可以直接部署在客户设备上，如笔记本电脑和智能手机，这是苹果公司目前正在探索的事情。

This local implementation can significantly decrease latency and reduce server-related costs. Furthermore, custom LLMs grant developers complete autonomy, allowing them to control updates and modifications to the model as needed.

The general process of creating an LLM includes pretraining and fine-tuning. The "pre" in "pretraining" refers to the initial phase where a model like an LLM is trained on a large, diverse dataset to develop a broad understanding of language. This pretrained model then serves as a foundational resource that can be further refined through fine-tuning, a process where the model is specifically trained on a narrower dataset that is more specific to particular tasks or domains. This two-stage training approach consisting of pretraining and fine-tuning is depicted in figure 1.3.

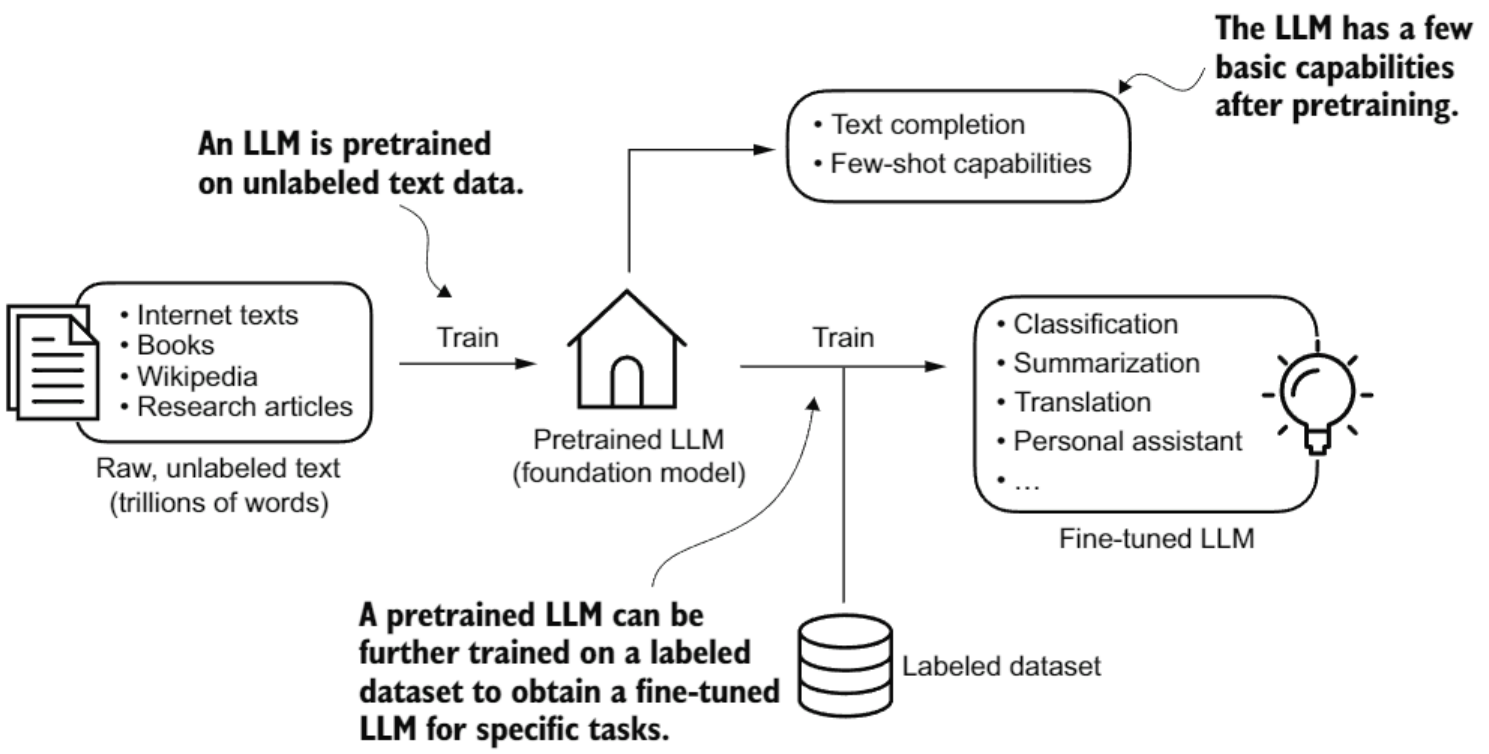

Figure 1.3 Pretraining an LLM involves next-word prediction on large text datasets. A pretrained LLM can then be fine-tuned using a smaller labeled dataset.

The first step in creating an LLM is to train it on a large corpus of text data, sometimes referred to as *raw* text. Here, "raw" refers to the fact that this data is just regular text without any labeling information. (Filtering may be applied, such as removing formatting characters or documents in unknown languages.)

> **NOTE** Readers with a background in machine learning may note that labeling information is typically required for traditional machine learning models and deep neural networks trained via the conventional supervised learning paradigm. However, this is not the case for the pretraining stage of LLMs. In this phase, LLMs use self-supervised learning, where the model generates its own labels from the input data.

本本地实现可以显著降低延迟并减少与服务器相关的成本。此外，自定义LLMs授予开发者完全自主权，允许他们根据需要控制模型的更新和修改。

创建LLM的一般过程包括预训练和微调。"预训练"中的“预”指的是初始阶段，其中类似于LLM的模型在大规模、多样化的数据集上训练，以发展对语言的广泛理解。这个预训练模型随后作为基础资源，可以通过微调进一步优化，微调是一个在更具体于特定任务或领域的较窄数据集上对模型进行特定训练的过程。这种由预训练和微调组成的两阶段训练方法在图 1.3 中展示。

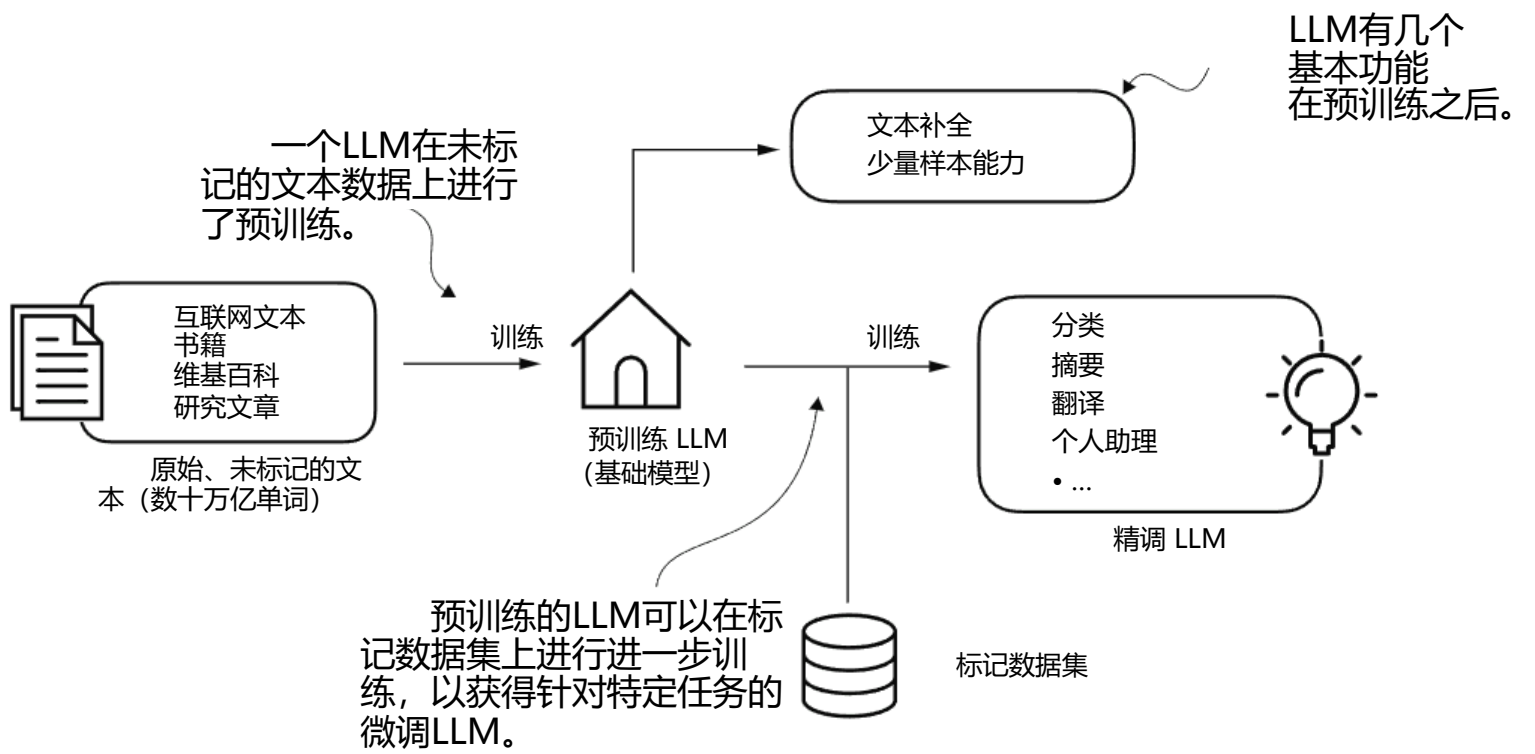

图 1.3 预训练LLM涉及在大规模文本数据集上进行下一词预测。然后可以使用较小的标记数据集对预训练的LLM进行微调。

创建LLM的第一步是在大量文本数据集上对其进行训练，有时也称为原始文本。这里的“原始”指的是这些数据仅仅是常规文本，没有任何标签信息。（可能应用过滤，例如删除格式化字符或未知语言的文档。）

注意：具有机器学习背景的读者可能会注意到，对于传统的机器学习模型和通过传统监督学习范式训练的深度神经网络，通常需要标签信息。然而，在LLMs的预训练阶段并非如此。在这个阶段，LLMs使用自监督学习，模型从输入数据中生成自己的标签。

This first training stage of an LLM is also known as *pretraining*, creating an initial pretrained LLM, often called a *base* or *foundation model*. A typical example of such a model is the GPT-3 model (the precursor of the original model offered in ChatGPT). This model is capable of text completion—that is, finishing a half-written sentence provided by a user. It also has limited few-shot capabilities, which means it can learn to perform new tasks based on only a few examples instead of needing extensive training data.

After obtaining a pretrained LLM from training on large text datasets, where the LLM is trained to predict the next word in the text, we can further train the LLM on labeled data, also known as *fine-tuning*.

The two most popular categories of fine-tuning LLMs are *instruction fine-tuning* and *classification fine-tuning*. In instruction fine-tuning, the labeled dataset consists of instruction and answer pairs, such as a query to translate a text accompanied by the correctly translated text. In classification fine-tuning, the labeled dataset consists of texts and associated class labels—for example, emails associated with "spam" and "not spam" labels.

We will cover code implementations for pretraining and fine-tuning an LLM, and we will delve deeper into the specifics of both instruction and classification fine-tuning after pretraining a base LLM.

1.4 Introducing the transformer architecture

Most modern LLMs rely on the *transformer* architecture, which is a deep neural network architecture introduced in the 2017 paper "Attention Is All You Need" (https://arxiv.org/abs/1706.03762). To understand LLMs, we must understand the original transformer, which was developed for machine translation, translating English texts to German and French. A simplified version of the transformer architecture is depicted in figure 1.4.

The transformer architecture consists of two submodules: an encoder and a decoder. The encoder module processes the input text and encodes it into a series of numerical representations or vectors that capture the contextual information of the input. Then, the decoder module takes these encoded vectors and generates the output text. In a translation task, for example, the encoder would encode the text from the source language into vectors, and the decoder would decode these vectors to generate text in the target language. Both the encoder and decoder consist of many layers connected by a so-called self-attention mechanism. You may have many questions regarding how the inputs are preprocessed and encoded. These will be addressed in a step-by-step implementation in subsequent chapters.

A key component of transformers and LLMs is the self-attention mechanism (not shown), which allows the model to weigh the importance of different words or tokens in a sequence relative to each other. This mechanism enables the model to capture long-range dependencies and contextual relationships within the input data, enhancing its ability to generate coherent and contextually relevant output. However, due to

这个LLM的第一个训练阶段也被称为预训练，创建一个初始预训练的LLM，通常称为基础或基础模型。此类模型的典型例子是 GPT-3 模型（ChatGPT 中提供的原始模型的前身）。该模型能够完成文本——即完成用户提供的半写句子。它还具有有限的少样本能力，这意味着它可以通过仅几个示例来学习执行新任务，而不是需要大量的训练数据。

在从大型文本数据集上训练获得预训练的LLM之后，其中LLM被训练来预测文本中的下一个单词，我们可以在标记数据上进一步训练LLM，这被称为微调。

两个最受欢迎的微调类别是指令微调和分类微调。在指令微调中，标记数据集由指令和答案对组成，例如一个查询翻译文本并附带正确翻译的文本。在分类微调中，标记数据集由文本及其相关类别标签组成——例如，与“垃圾邮件”和“非垃圾邮件”标签相关的电子邮件。

我们将介绍预训练和微调LLM的代码实现，并在基于LLM预训练后，更深入地探讨指令和分类微调的细节。

1.4 介绍 Transformer 架构

大多数现代LLMs依赖于 Transformer 架构，这是一种在 2017 年论文“Attention Is All You Need”（https://arxiv.org/abs/1706.03762）中引入的深度神经网络架构。要理解LLMs，我们必须了解原始的 Transformer，它是为机器翻译开发的，将英语文本翻译成德语和法语。Transformer 架构的简化版本在图 1.4 中展示。

变压器架构由两个子模块组成：一个编码器和一个解码器。编码器模块处理输入文本并将其编码成一系列数值表示或向量，这些向量捕捉输入的上下文信息。然后，解码器模块接收这些编码向量并生成输出文本。例如，在翻译任务中，编码器会将源语言的文本编码成向量，解码器会将这些向量解码以生成目标语言的文本。编码器和解码器都由许多层组成，这些层通过所谓的自注意力机制相互连接。您可能对输入如何进行预处理和编码有许多疑问。这些问题将在后续章节的逐步实现中解决。

Transformer 和LLMs的关键组件是自注意力机制（未显示），该机制允许模型根据彼此的重要性对序列中的不同单词或标记进行加权。这种机制使模型能够捕捉输入数据中的长距离依赖关系和上下文关系，增强其生成连贯且上下文相关的输出的能力。然而，由于

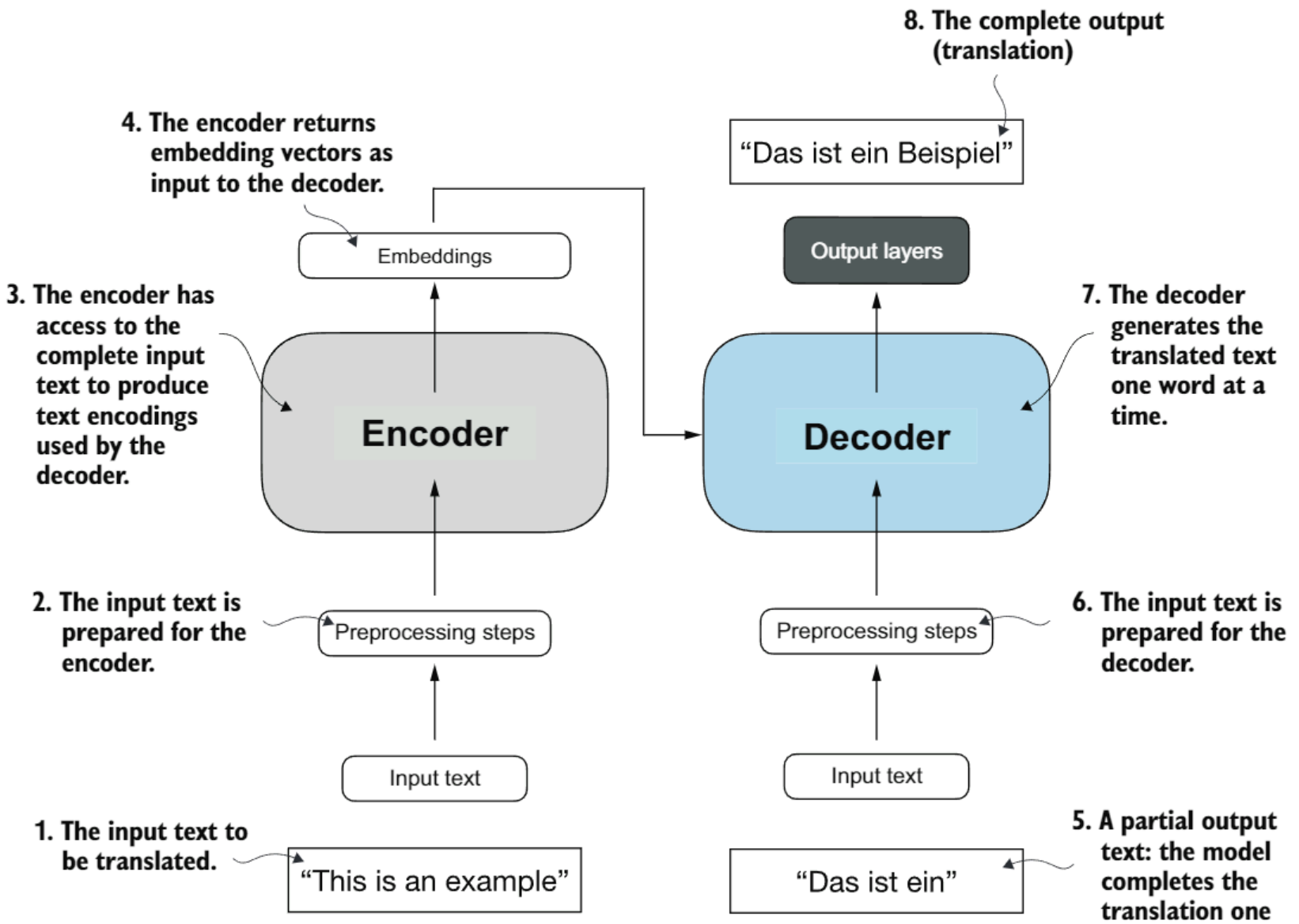

Figure 1.4 A simplified depiction of the original transformer architecture, which is a deep learning model for language translation. The transformer consists of two parts: (a) an encoder that processes the input text and produces an embedding representation (a numerical representation that captures many different factors in different dimensions) of the text that the (b) decoder can use to generate the translated text one word at a time. This figure shows the final stage of the translation process where the decoder has to generate only the final word (“Beispiel”), given the original input text (“This is an example”) and a partially translated sentence (“Das ist ein”), to complete the translation.

its complexity, we will defer further explanation to chapter 3, where we will discuss and implement it step by step.

Later variants of the transformer architecture, such as BERT (short for *bidirectional encoder representations from transformers*) and the various GPT models (short for *generative pretrained transformers*), built on this concept to adapt this architecture for different tasks. If interested, refer to appendix B for further reading suggestions.

BERT, which is built upon the original transformer’s encoder submodule, differs in its training approach from GPT. While GPT is designed for generative tasks, BERT and its variants specialize in masked word prediction, where the model predicts masked

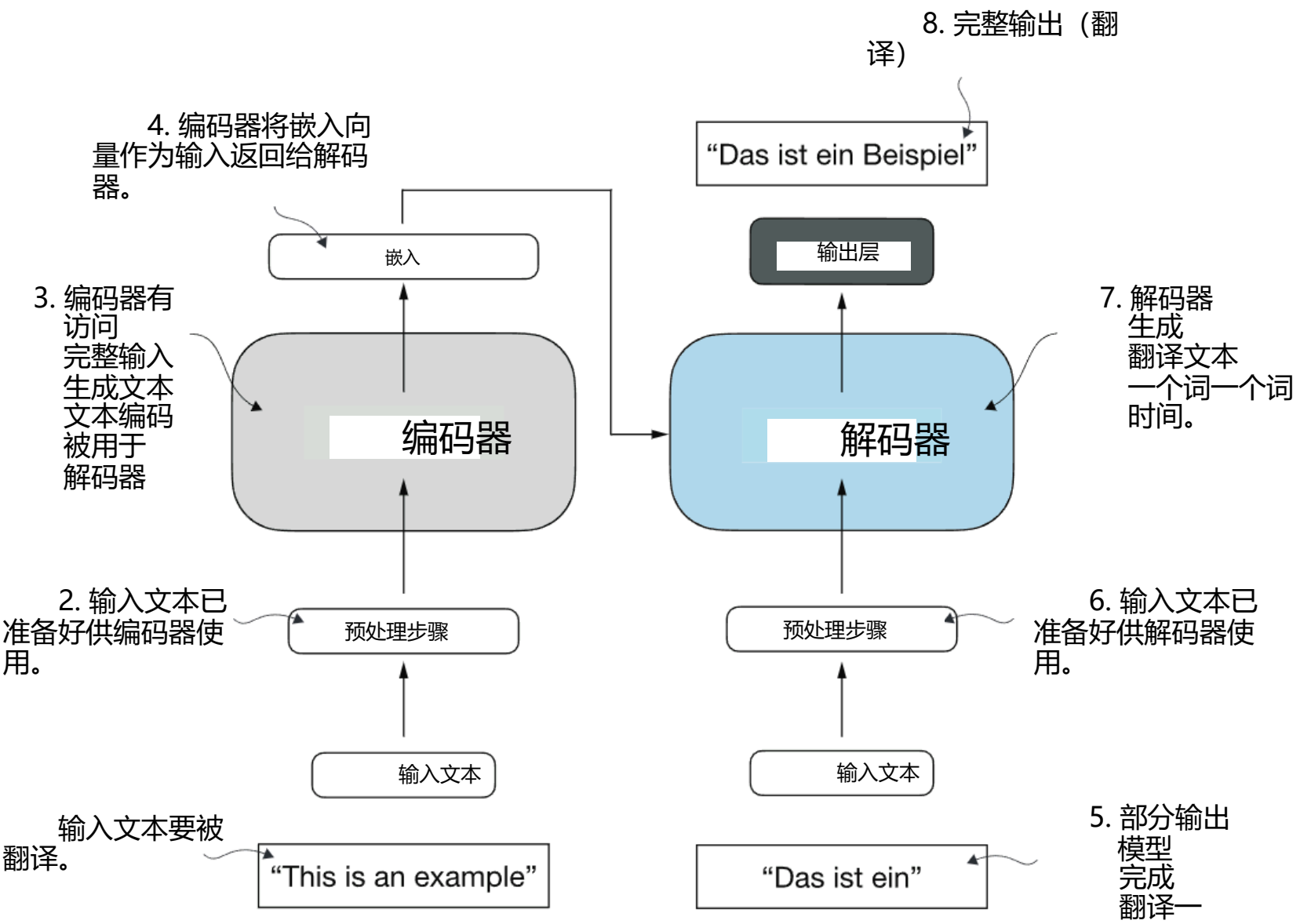

图 1.4 展示了原始变压器架构的简化图，这是一个用于语言翻译的深度学习模型。变压器由两部分组成：(a)一个编码器，它处理输入文本并生成文本的嵌入表示（一种在多个维度上捕捉许多不同因素的数值表示），(b)解码器可以使用该表示逐词生成翻译文本。此图显示了翻译过程的最后阶段，其中解码器必须根据原始输入文本（“这是一个例子”）和部分翻译句子（“这是”）生成最后一个单词（“Beispiel”），以完成翻译。

其复杂性，我们将进一步解释推迟到第 3 章，在那里我们将逐步讨论和实现它。

后续的 Transformer 架构变体，如 BERT（双向编码器表示的 Transformer 的缩写）和各种 GPT 模型（生成预训练 Transformer 的缩写），基于此概念来适应不同任务。如感兴趣，请参阅附录 B 以获取进一步阅读建议。

BERT，基于原始 transformer 的编码子模块构建，其在训练方法上与 GPT 不同。虽然 GPT 是为生成任务设计的，BERT 及其变体则专注于掩码词预测，其中模型预测掩码

or hidden words in a given sentence, as shown in figure 1.5. This unique training strategy equips BERT with strengths in text classification tasks, including sentiment prediction and document categorization. As an application of its capabilities, as of this writing, X (formerly Twitter) uses BERT to detect toxic content.

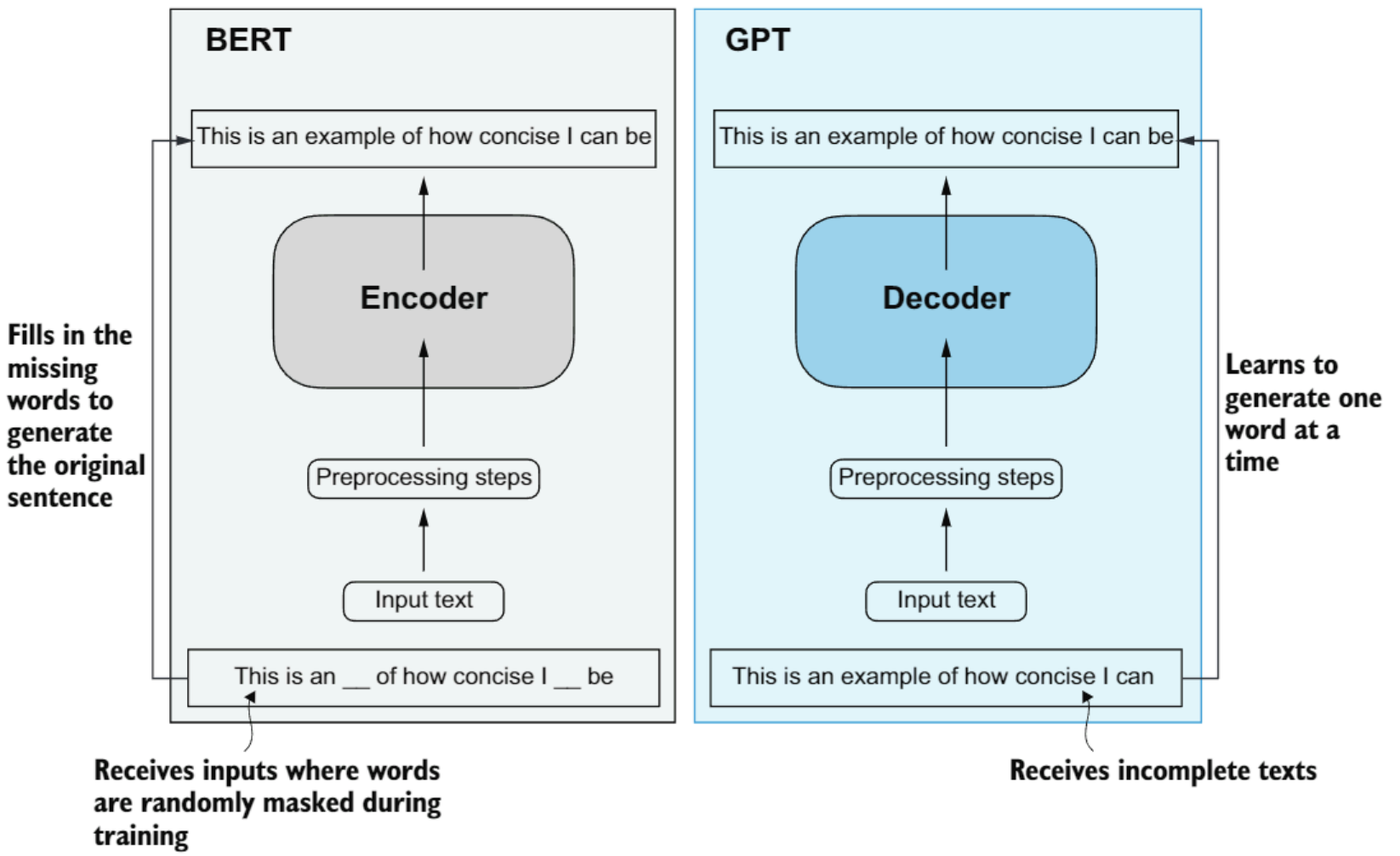

Figure 1.5 A visual representation of the transformer's encoder and decoder submodules. On the left, the encoder segment exemplifies BERT-like LLMs, which focus on masked word prediction and are primarily used for tasks like text classification. On the right, the decoder segment showcases GPT-like LLMs, designed for generative tasks and producing coherent text sequences.

GPT, on the other hand, focuses on the decoder portion of the original transformer architecture and is designed for tasks that require generating texts. This includes machine translation, text summarization, fiction writing, writing computer code, and more.

GPT models, primarily designed and trained to perform text completion tasks, also show remarkable versatility in their capabilities. These models are adept at executing both zero-shot and few-shot learning tasks. Zero-shot learning refers to the ability to generalize to completely unseen tasks without any prior specific examples. On the other hand, few-shot learning involves learning from a minimal number of examples the user provides as input, as shown in figure 1.6.

或隐藏在给定句子中的单词，如图 1.5 所示。这种独特的训练策略使 BERT 在文本分类任务中具有优势，包括情感预测和文档分类。作为其能力的一种应用，截至本文撰写时，X（前身为 Twitter）使用 BERT 来检测有害内容。

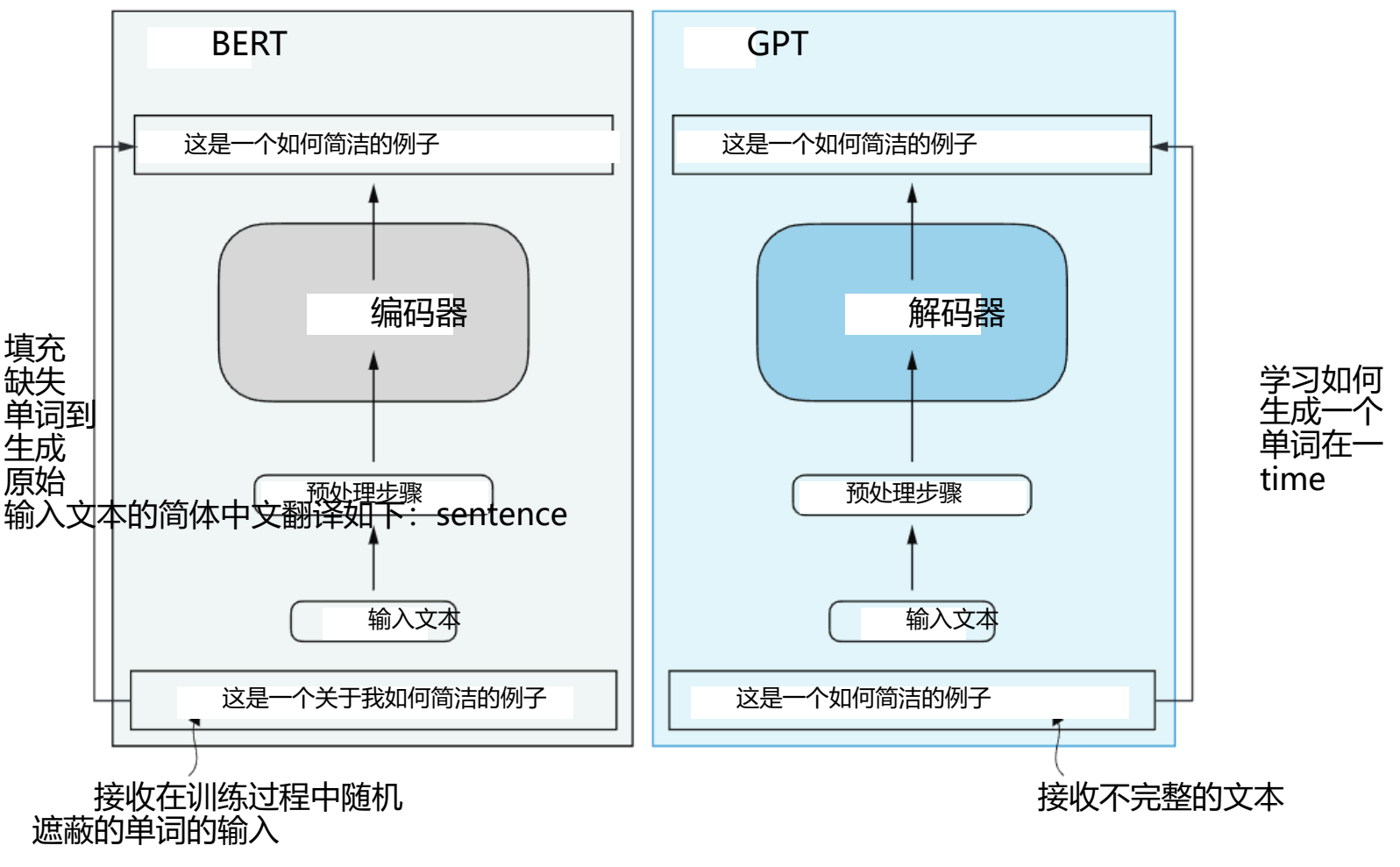

图 1.5 变压器的编码器和解码器子模块的视觉表示。左侧的编码器部分展示了类似 BERT 的LLMs，专注于掩码词预测，主要用于文本分类等任务。右侧的解码器部分展示了类似 GPT 的LLMs，用于生成任务，产生连贯的文本序列。

GPT 另一方面，专注于原始 Transformer 架构的解码器部分，旨在处理需要生成文本的任务。这包括机器翻译、文本摘要、小说创作、编写计算机代码等。

GPT 模型，主要设计和训练用于执行文本补全任务，在能力上也表现出显著的灵活性。这些模型擅长执行零样本和少样本学习任务。零样本学习指的是在没有任何先前特定示例的情况下，泛化到完全未见过的任务的能力。另一方面，少样本学习涉及从用户提供的最少数量示例中进行学习，如图 1.6 所示。

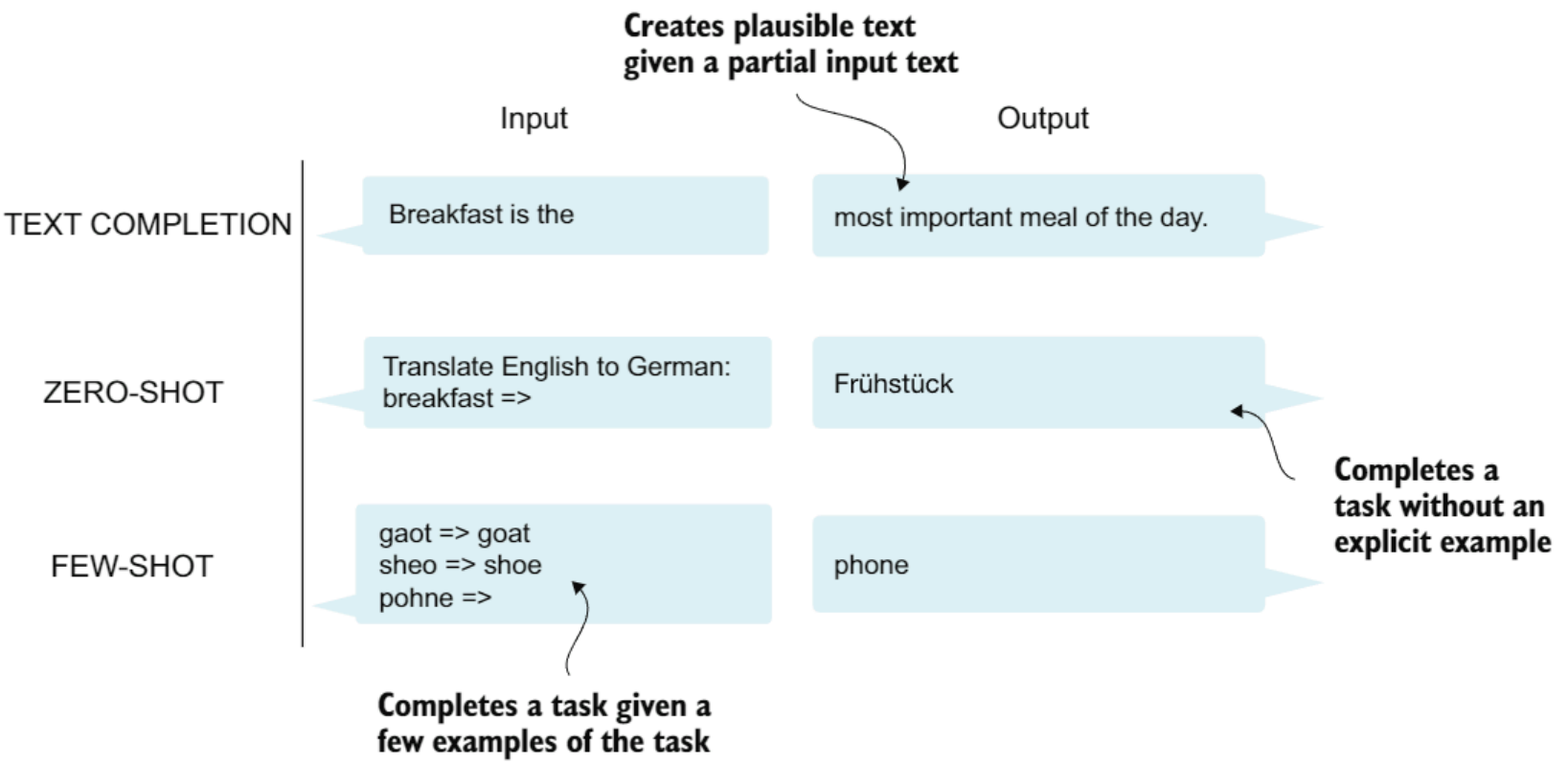

Figure 1.6 In addition to text completion, GPT-like LLMs can solve various tasks based on their inputs without needing retraining, fine-tuning, or task-specific model architecture changes. Sometimes it is helpful to provide examples of the target within the input, which is known as a few-shot setting. However, GPT-like LLMs are also capable of carrying out tasks without a specific example, which is called zero-shot setting.

Transformers vs. LLMs

Today's LLMs are based on the transformer architecture. Hence, transformers and LLMs are terms that are often used synonymously in the literature. However, note that not all transformers are LLMs since transformers can also be used for computer vision. Also, not all LLMs are transformers, as there are LLMs based on recurrent and convolutional architectures. The main motivation behind these alternative approaches is to improve the computational efficiency of LLMs. Whether these alternative LLM architectures can compete with the capabilities of transformer-based LLMs and whether they are going to be adopted in practice remains to be seen. For simplicity, I use the term "LLM" to refer to transformer-based LLMs similar to GPT. (Interested readers can find literature references describing these architectures in appendix B.)

1.5 Utilizing large datasets

The large training datasets for popular GPT- and BERT-like models represent diverse and comprehensive text corpora encompassing billions of words, which include a vast array of topics and natural and computer languages. To provide a concrete example, table 1.1 summarizes the dataset used for pretraining GPT-3, which served as the base model for the first version of ChatGPT.

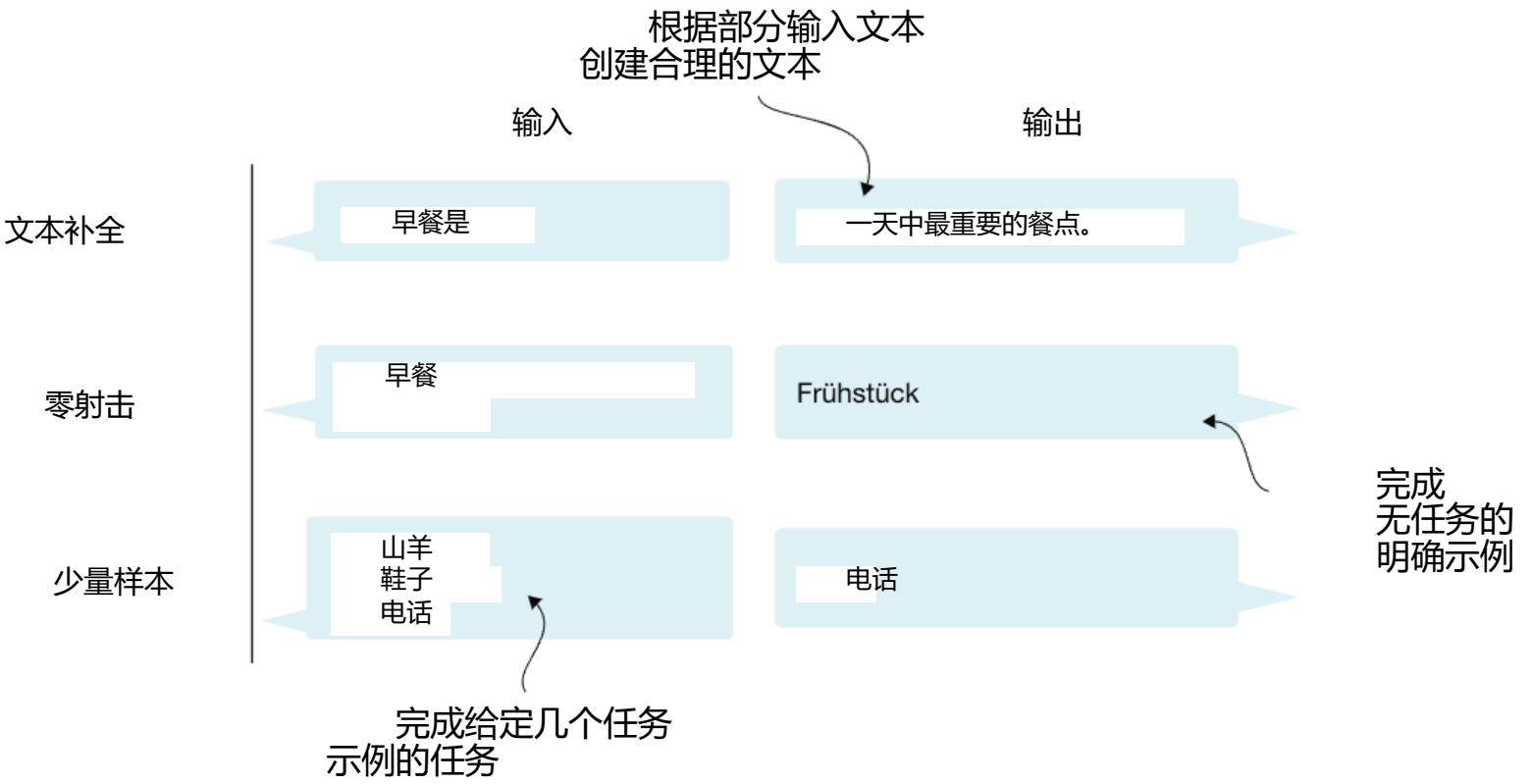

图 1.6 除了文本补全，GPT-like LLMs 还可以根据其输入解决各种任务，无需重新训练、微调或特定任务模型架构更改。有时在输入中提供目标示例是有帮助的，这被称为少量样本设置。然而，GPT-like LLMs 也能在没有特定示例的情况下执行任务，这被称为零样本设置。

Transformer 与LLMs

今天的LLMs基于 Transformer 架构。因此，在文献中，transformers 和LLMs是经常被同义使用的术语。然而，请注意，并非所有 transformers 都是LLMs，因为 transformers 也可以用于计算机视觉。同样，并非所有LLMs都是 transformers，因为还有基于循环和卷积架构的LLMs。这些替代方法背后的主要动机是提高LLMs的计算效率。这些替代的LLM架构能否与基于 transformer 的LLMs的能力竞争，以及它们是否将在实践中被采用，还有待观察。为了简单起见，我使用“LLM”一词来指代类似于 GPT 的基于 transformer 的LLMs。（感兴趣的读者可以在附录 B 中找到描述这些架构的文献参考。）

1.5 利用大型数据集

大型训练数据集为流行的 GPT 和 BERT 类似模型提供了涵盖数十亿词汇的多样化和全面文本语料库，包括广泛的主题和自然语言与计算机语言。为了提供一个具体的例子，表 1.1 总结了用于预训练 GPT-3 的数据集，该数据集是 ChatGPT 第一版的基础模型。

Table 1.1 The pretraining dataset of the popular GPT-3 LLM

Dataset name	Dataset description	Number of tokens	Proportion in training data
CommonCrawl (filtered)	Web crawl data	410 billion	60%
WebText2	Web crawl data	19 billion	22%
Books1	Internet-based book corpus	12 billion	8%
Books2	Internet-based book corpus	55 billion	8%
Wikipedia	High-quality text	3 billion	3%

Table 1.1 reports the number of tokens, where a token is a unit of text that a model reads and the number of tokens in a dataset is roughly equivalent to the number of words and punctuation characters in the text. Chapter 2 addresses tokenization, the process of converting text into tokens.

The main takeaway is that the scale and diversity of this training dataset allow these models to perform well on diverse tasks, including language syntax, semantics, and context—even some requiring general knowledge.

GPT-3 dataset details

Table 1.1 displays the dataset used for GPT-3. The proportions column in the table sums up to 100% of the sampled data, adjusted for rounding errors. Although the subsets in the Number of Tokens column total 499 billion, the model was trained on only 300 billion tokens. The authors of the GPT-3 paper did not specify why the model was not trained on all 499 billion tokens.

For context, consider the size of the CommonCrawl dataset, which alone consists of 410 billion tokens and requires about 570 GB of storage. In comparison, later iterations of models like GPT-3, such as Meta's LLaMA, have expanded their training scope to include additional data sources like Arxiv research papers (92 GB) and StackExchange's code-related Q&As (78 GB).

The authors of the GPT-3 paper did not share the training dataset, but a comparable dataset that is publicly available is *Dolma: An Open Corpus of Three Trillion Tokens for LLM Pretraining Research* by Soldaini et al. 2024 (https://arxiv.org/abs/2402.00159). However, the collection may contain copyrighted works, and the exact usage terms may depend on the intended use case and country.

The pretrained nature of these models makes them incredibly versatile for further fine-tuning on downstream tasks, which is why they are also known as base or foundation models. Pretraining LLMs requires access to significant resources and is very expensive. For example, the GPT-3 pretraining cost is estimated to be $4.6 million in terms of cloud computing credits (https://mng.bz/VxEW).

表 1.1 GPT-3 LLM的预训练数据集

数据集名称	数据集描述	token 数量	比例 训练数据
公共爬虫（过滤后）	网络爬虫数据	4100 亿	60%
网络文本 2	网络爬虫数据	190 亿	22%
书籍 1	基于互联网的书语料库	120 亿	8%
书籍 2	基于互联网的书语料库	550 亿	8%
维基百科	高质量文本	30 亿	3%

表 1.1 报告了标记的数量，其中标记是模型读取的文本单位，一个数据集中的标记数量大致相当于文本中的单词和标点符号的数量。第二章讨论了分词，即将文本转换为标记的过程。

主要收获是，这个训练数据集的规模和多样性使得这些模型能够在包括语言语法、语义和上下文在内的各种任务上表现良好——甚至包括一些需要一般知识的任务。

GPT-3 数据集详情

表 1.1 显示了用于 GPT-3 的数据集。表中比例列的总和为调整舍入误差后的样本数据的 100%。尽管令牌数量列的子集总数为 4990 亿，但模型仅训练了 3000 亿个令牌。GPT-3 论文的作者没有说明为什么没有在所有 4990 亿个令牌上训练模型。

考虑到 CommonCrawl 数据集的大小，它本身包含 4100 亿个标记，需要大约 570GB 的存储空间。相比之下，GPT-3 等模型的后续版本，如 Meta 的 LLaMA，已经扩大了它们的训练范围，包括额外的数据源，如 Arxiv 研究论文（92GB）和 StackExchange 的与代码相关的问答（78GB）。

GPT-3 论文的作者没有分享训练数据集，但一个可比较的

公开可用的数据集是 Dolma：Soldaini 等人 2024 年发表的用于LLM预训练研究的三万亿标记开放语料库（https://arxiv.org/abs/2402.00159）。然而，该集合可能包含受版权保护的作品，具体使用条款可能取决于预期用途和所在国家。

这些模型的预训练特性使它们在下游任务上的微调方面具有极高的灵活性，这也是它们被称为基础或底层模型的原因。在 1001#上进行预训练需要大量资源，并且成本非常高。例如，GPT-3 的预训练成本据估计为 460 万美元的云计算信用额度（https://mng.bz/VxEW）。

The good news is that many pretrained LLMs, available as open source models, can be used as general-purpose tools to write, extract, and edit texts that were not part of the training data. Also, LLMs can be fine-tuned on specific tasks with relatively smaller datasets, reducing the computational resources needed and improving performance.

We will implement the code for pretraining and use it to pretrain an LLM for educational purposes. All computations are executable on consumer hardware. After implementing the pretraining code, we will learn how to reuse openly available model weights and load them into the architecture we will implement, allowing us to skip the expensive pretraining stage when we fine-tune our LLM.

1.6 A closer look at the GPT architecture

GPT was originally introduced in the paper "Improving Language Understanding by Generative Pre-Training" (https://mng.bz/x2qg) by Radford et al. from OpenAI. GPT-3 is a scaled-up version of this model that has more parameters and was trained on a larger dataset. In addition, the original model offered in ChatGPT was created by fine-tuning GPT-3 on a large instruction dataset using a method from OpenAI's InstructGPT paper (https://arxiv.org/abs/2203.02155). As figure 1.6 shows, these models are competent text completion models and can carry out other tasks such as spelling correction, classification, or language translation. This is actually very remarkable given that GPT models are pretrained on a relatively simple next-word prediction task, as depicted in figure 1.7.

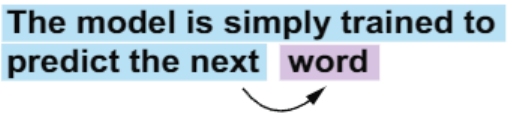

Figure 1.7 In the next-word prediction pretraining task for GPT models, the system learns to predict the upcoming word in a sentence by looking at the words that have come before it. This approach helps the model understand how words and phrases typically fit together in language, forming a foundation that can be applied to various other tasks.

The next-word prediction task is a form of self-supervised learning, which is a form of self-labeling. This means that we don't need to collect labels for the training data explicitly but can use the structure of the data itself: we can use the next word in a sentence or document as the label that the model is supposed to predict. Since this next-word prediction task allows us to create labels "on the fly," it is possible to use massive unlabeled text datasets to train LLMs.

Compared to the original transformer architecture we covered in section 1.4, the general GPT architecture is relatively simple. Essentially, it's just the decoder part without the encoder (figure 1.8). Since decoder-style models like GPT generate text by predicting text one word at a time, they are considered a type of *autoregressive* model. Autoregressive models incorporate their previous outputs as inputs for future

好消息是，许多可公开获取的预训练模型LLMs可以用作通用工具，用于编写、提取和编辑训练数据之外的文本。此外，LLMs可以在相对较小的数据集上进行微调，从而减少所需的计算资源并提高性能。

我们将实现预训练代码，并用于教育目的进行预训练一个LLM。所有计算都可以在消费级硬件上执行。在实现预训练代码后，我们将学习如何重用公开可用的模型权重，并将它们加载到我们将要实现的架构中，使我们能够在微调我们的LLM时跳过昂贵的预训练阶段。

1.6 更深入地了解 GPT 架构

GPT 最初由 OpenAI 的 Radford 等人发表的论文《通过生成预训练改进语言理解》（https://mng.bz/x2qg）中引入。GPT-3 是这个模型的扩展版本，具有更多参数，并在更大的数据集上进行了训练。此外，ChatGPT 中提供的原始模型是通过在大型指令数据集上使用 OpenAI 的 InstructGPT 论文（https://arxiv.org/abs/2203.02155）中的方法微调 GPT-3 创建的。如图 1.6 所示，这些模型是胜任文本补全模型，可以执行其他任务，如拼写校正、分类或语言翻译。考虑到 GPT 模型是在如图 1.7 所示的相对简单的下一个单词预测任务上进行预训练的，这实际上是非常了不起的。

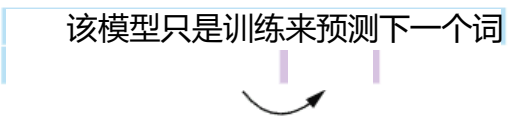

图 1.7 在 GPT 模型的下一个词预测预训练任务中，系统通过观察句子中前面的词来学习预测句子中的下一个词。这种方法有助于模型理解单词和短语在语言中通常是如何搭配在一起的，形成一个可以应用于各种其他任务的基础。

下一词预测任务是一种自监督学习形式，它是一种自标注形式。这意味着我们不需要显式收集训练数据的标签，而是可以使用数据本身的结构：我们可以使用句子或文档中的下一个词作为模型应该预测的标签。由于这个下一词预测任务允许我们“即时”创建标签，因此可以使用大量的未标记文本数据集进行训练。

与我们在第 1.4 节中介绍的原始 Transformer 架构相比，通用 GPT 架构相对简单。本质上，它只是没有编码器的解码器部分（图 1.8）。由于像 GPT 这样的解码器风格模型通过逐词预测文本来生成文本，因此它们被视为一种自回归模型。自回归模型将它们的先前输出作为未来输入。

predictions. Consequently, in GPT, each new word is chosen based on the sequence that precedes it, which improves the coherence of the resulting text.

Architectures such as GPT-3 are also significantly larger than the original transformer model. For instance, the original transformer repeated the encoder and decoder blocks six times. GPT-3 has 96 transformer layers and 175 billion parameters in total.

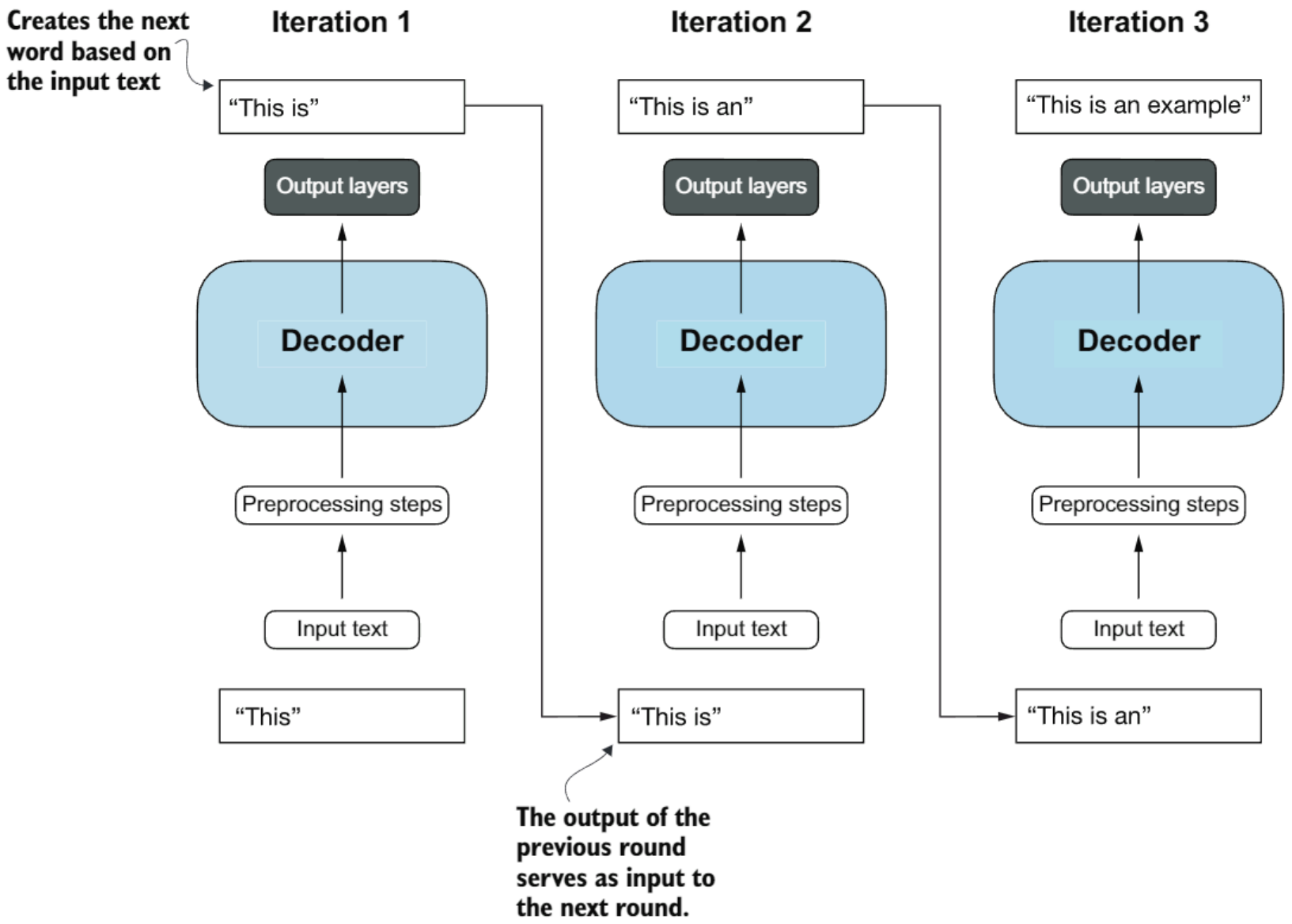

Figure 1.8 The GPT architecture employs only the decoder portion of the original transformer. It is designed for unidirectional, left-to-right processing, making it well suited for text generation and next-word prediction tasks to generate text in an iterative fashion, one word at a time.

GPT-3 was introduced in 2020, which, by the standards of deep learning and large language model development, is considered a long time ago. However, more recent architectures, such as Meta's Llama models, are still based on the same underlying concepts, introducing only minor modifications. Hence, understanding GPT remains as relevant as ever, so I focus on implementing the prominent architecture behind GPT while providing pointers to specific tweaks employed by alternative LLMs.

Although the original transformer model, consisting of encoder and decoder blocks, was explicitly designed for language translation, GPT models—despite their larger yet

预测。因此，在 GPT 中，每个新词的选择都是基于其前面的序列，这提高了生成文本的连贯性。

架构如 GPT-3 也远大于原始的 Transformer 模型。例如，原始的 Transformer 重复了编码器和解码器块六次。GPT-3 总共有 96 个 Transformer 层和 1750 亿个参数。

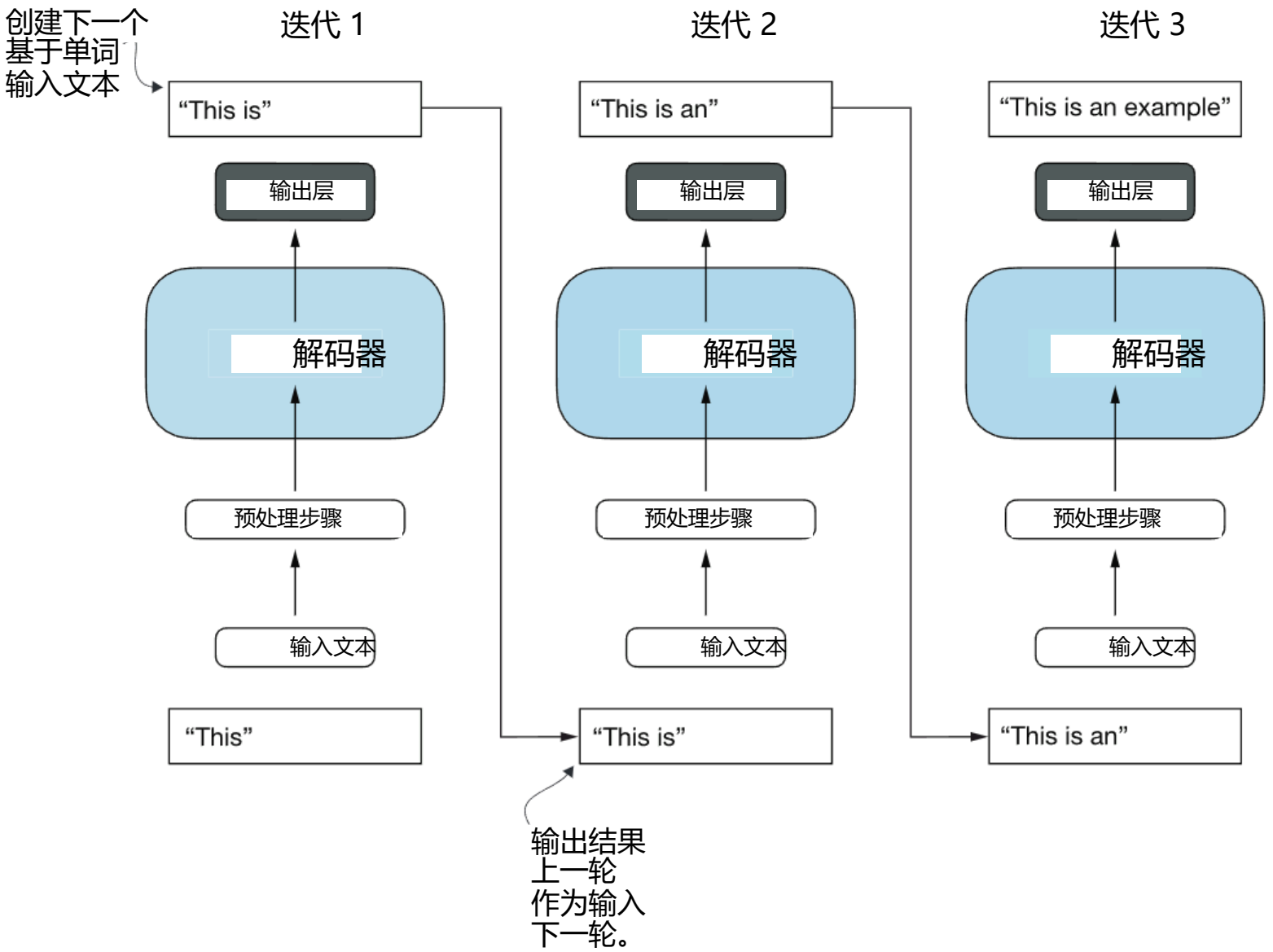

图 1.8 GPT 架构仅使用原始 Transformer 的解码器部分。它设计用于单向、从左到右的处理，非常适合以迭代方式逐词生成文本和预测下一个单词的任务。

GPT-3 于 2020 年推出，按照深度学习和大型语言模型发展的标准，这已经是很久以前了。然而，更近期的架构，如 Meta 的 Llama 模型，仍然基于相同的基本概念，只是进行了细微的修改。因此，理解 GPT 仍然像以前一样重要，所以我专注于实现 GPT 背后的突出架构，同时提供对替代方案中使用的特定调整的提示。

尽管原始的 Transformer 模型，由编码器和解码器块组成，是专门为语言翻译设计的，但 GPT 模型——尽管它们更大，却

simpler decoder-only architecture aimed at next-word prediction—are also capable of performing translation tasks. This capability was initially unexpected to researchers, as it emerged from a model primarily trained on a next-word prediction task, which is a task that did not specifically target translation.

The ability to perform tasks that the model wasn't explicitly trained to perform is called an *emergent behavior*. This capability isn't explicitly taught during training but emerges as a natural consequence of the model's exposure to vast quantities of multilingual data in diverse contexts. The fact that GPT models can "learn" the translation patterns between languages and perform translation tasks even though they weren't specifically trained for it demonstrates the benefits and capabilities of these large-scale, generative language models. We can perform diverse tasks without using diverse models for each.

1.7 Building a large language model

Now that we've laid the groundwork for understanding LLMs, let's code one from scratch. We will take the fundamental idea behind GPT as a blueprint and tackle this in three stages, as outlined in figure 1.9.

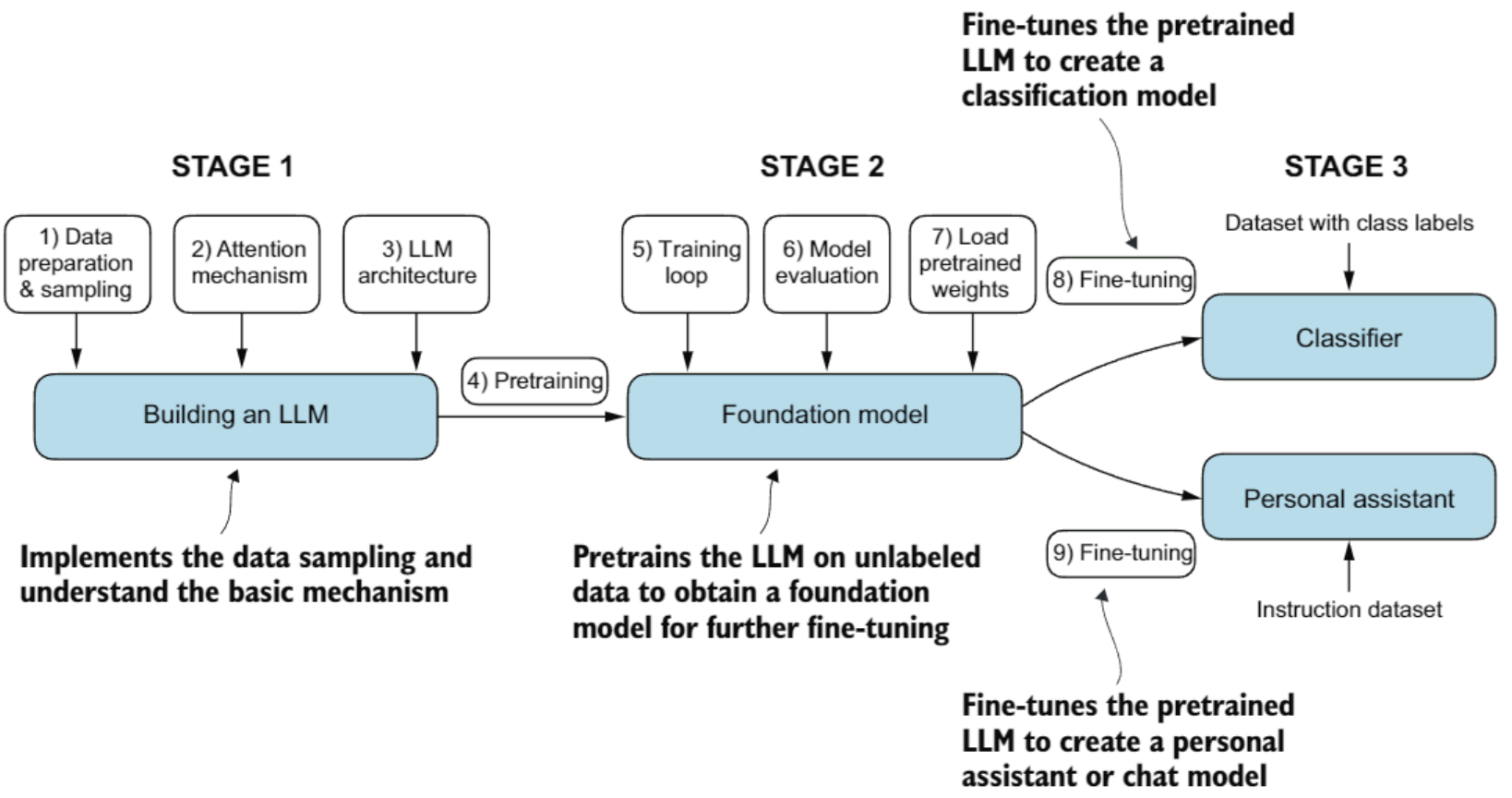

Figure 1.9 The three main stages of coding an LLM are implementing the LLM architecture and data preparation process (stage 1), pretraining an LLM to create a foundation model (stage 2), and fine-tuning the foundation model to become a personal assistant or text classifier (stage 3).

更简单的仅解码器架构，旨在进行下一词预测——也能执行翻译任务。这一能力最初让研究人员感到意外，因为它主要是在一个下一词预测任务上训练的模型中出现的，而这一任务并没有专门针对翻译。

模型能够执行其未明确训练过的任务，这被称为涌现行为。这种能力在训练期间并未明确教授，而是作为模型在多种语境下接触大量多语言数据的自然结果而出现。GPT 模型能够“学习”不同语言之间的翻译模式，即使它们并未为此专门训练，也能执行翻译任务，这展示了这些大规模生成语言模型的好处和能力。我们可以执行各种任务，而无需为每个任务使用不同的模型。

1.7 构建大型语言模型

现在我们已经为理解LLMs打下了基础，让我们从头开始编写代码。我们将以 GPT 背后的基本理念为蓝图，并按照图 1.9 中的三个阶段来解决这个问题。

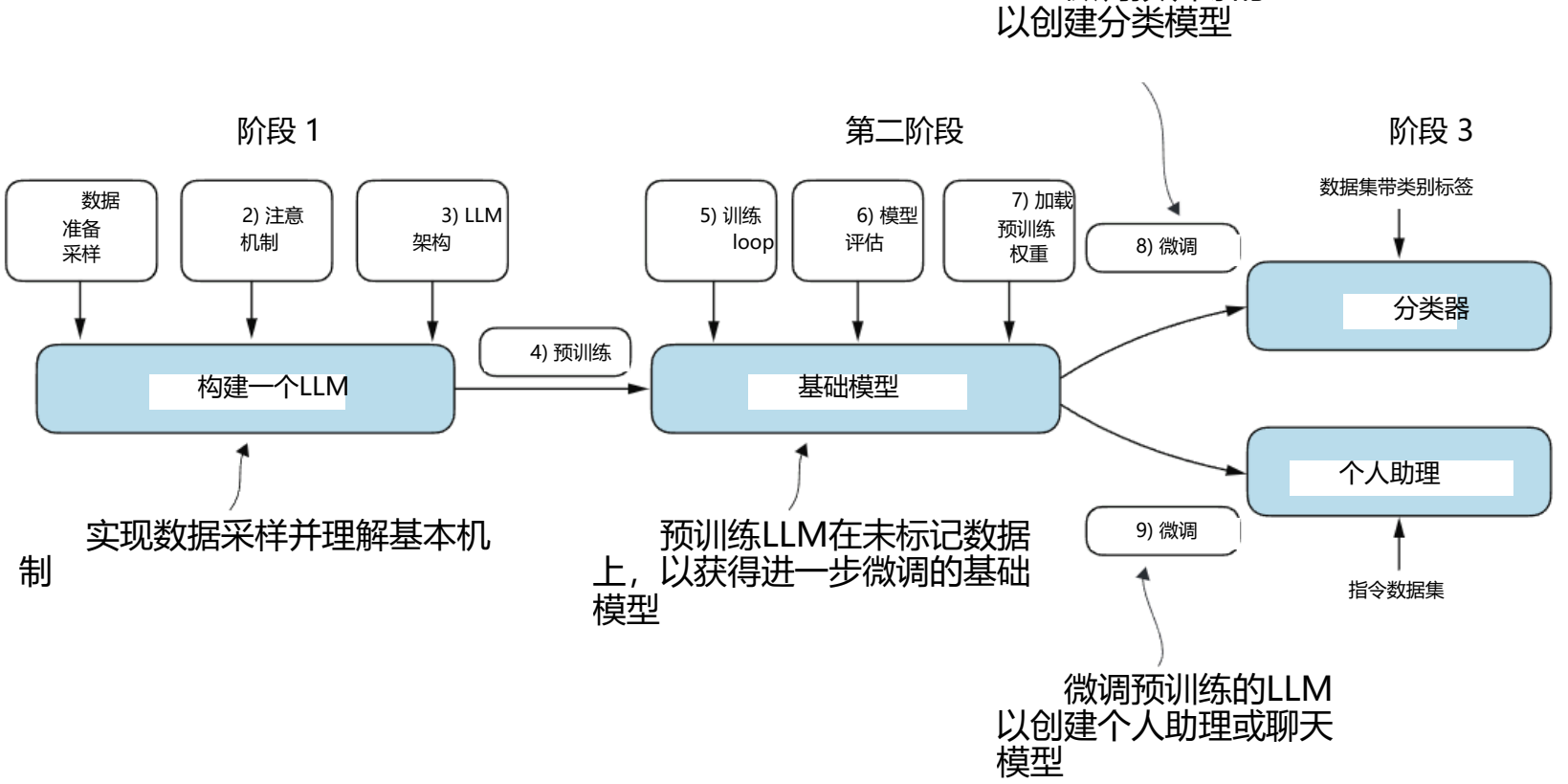

图 1.9 LLM 编码的三个主要阶段是实施 LLM 架构和数据准备过程（阶段 1），预训练 LLM 以创建基础模型（阶段 2），以及微调基础模型以成为个人助理或文本分类器（阶段 3）。

In stage 1, we will learn about the fundamental data preprocessing steps and code the attention mechanism at the heart of every LLM. Next, in stage 2, we will learn how to code and pretrain a GPT-like LLM capable of generating new texts. We will also go over the fundamentals of evaluating LLMs, which is essential for developing capable NLP systems.

Pretraining an LLM from scratch is a significant endeavor, demanding thousands to millions of dollars in computing costs for GPT-like models. Therefore, the focus of stage 2 is on implementing training for educational purposes using a small dataset. In addition, I also provide code examples for loading openly available model weights.

Finally, in stage 3, we will take a pretrained LLM and fine-tune it to follow instructions such as answering queries or classifying texts—the most common tasks in many real-world applications and research.

I hope you are looking forward to embarking on this exciting journey!

Summary

- LLMs have transformed the field of natural language processing, which previously mostly relied on explicit rule-based systems and simpler statistical methods. The advent of LLMs introduced new deep learning-driven approaches that led to advancements in understanding, generating, and translating human language.
- Modern LLMs are trained in two main steps:
  - First, they are pretrained on a large corpus of unlabeled text by using the prediction of the next word in a sentence as a label.
  - Then, they are fine-tuned on a smaller, labeled target dataset to follow instructions or perform classification tasks.
- LLMs are based on the transformer architecture. The key idea of the transformer architecture is an attention mechanism that gives the LLM selective access to the whole input sequence when generating the output one word at a time.
- The original transformer architecture consists of an encoder for parsing text and a decoder for generating text.
- LLMs for generating text and following instructions, such as GPT-3 and ChatGPT, only implement decoder modules, simplifying the architecture.
- Large datasets consisting of billions of words are essential for pretraining LLMs.
- While the general pretraining task for GPT-like models is to predict the next word in a sentence, these LLMs exhibit emergent properties, such as capabilities to classify, translate, or summarize texts.

在第一阶段，我们将学习基本的数据预处理步骤，并编写每个LLM核心的注意力机制代码。接下来，在第二阶段，我们将学习如何编码和预训练一个类似于 GPT 的 LLM，能够生成新的文本。我们还将了解评估LLMs的基本原理，这对于开发能够处理自然语言处理系统的能力至关重要。

从头开始预训练一个LLM是一个重大的努力，对于类似 GPT 的模型，需要数千到数百万美元的计算机成本。因此，第二阶段的重点是使用小数据集进行教育目的的训练。此外，我还提供了加载公开可用的模型权重的代码示例。

最终，在第三阶段，我们将使用预训练的LLM进行微调，以执行如回答查询或对文本进行分类等指令——这些是在许多实际应用和研究中最常见的任务。

我希望您期待着开始这段激动人心的旅程！

摘要

- LLMs 已将自然语言处理领域进行了变革，该领域之前主要依赖基于显式规则的系统以及更简单的统计方法。LLMs 的出现引入了新的深度学习驱动方法，这些方法推动了人类语言的理解、生成和翻译的进步。

- 现代LLMs的训练分为两个主要步骤：首先，它们在大量未标记文本语料库上通过使用句子中下一个单词的预测作为标签进行预训练。然后，它们在较小、标记的目标数据集上进行微调，以遵循指令或执行分类任务。

- 基于 Transformer 架构。Transformer 架构的关键思想是一种注意力机制，在逐词生成输出时，为整个输入序列提供选择性访问。

- 原始的 Transformer 架构包括用于解析文本的编码器以及用于生成文本的解码器。
- LLMs 用于生成文本和遵循指令，如 GPT-3 和 ChatGPT，仅实现解码器模块，简化架构。
- 大规模包含数十亿词汇的数据集对于预训练至关重要。

- 尽管 GPT 类模型的通用预训练任务是预测句子中的下一个单词，但这些LLMs展现出涌现特性，例如分类、翻译或总结文本的能力。

- Once an LLM is pretrained, the resulting foundation model can be fine-tuned more efficiently for various downstream tasks.
- LLMs fine-tuned on custom datasets can outperform general LLMs on specific tasks.

- 一旦预训练了LLM，结果的基础模型可以更高效地针对各种下游任务进行微调。
- 在自定义数据集上微调可以优于通用的在特定任务上。