第 1 章

大模型时代的开端

大模型时代是指当前人工智能技术阶段，此阶段采用了基于深度学习等技术的大型神经网络模型。这些模型具有数量级庞大的参数和极高的计算复杂度，需要海量数据、大规模计算力和强大的算法优化能力等条件来支撑它们的训练和应用。大模型时代催生了一系列新兴的 AI 技术和应用场景，如自然语言处理（NLP）、计算机视觉、语音识别、推荐系统等，使得 AI 技术逐渐成为人类社会发展的重要推动力量。

在大模型时代，AI 技术已经不仅仅是若干个单独的算法或者模型，而是通过集成、协作、创新等方式形成的更加完整和广泛的技术体系，拥有更多的可能性和应用空间。同时，AI 技术的发展也带来了许多挑战和问题，如数据隐私、算法公正性、人机关系等。

1.1 大模型的历史与发展

大模型凭借强大的建模能力和高效的训练速度，迅速成为自然语言处理领域的明星。随着研究的深入，人们逐渐发现模型参数的数量直接决定了模型的表达能力，于是研究者们开始不断增大模型的参数规模，从数百万增大到数亿，甚至更多。

随着大模型的普及和应用，其优点和潜力逐渐得到人们的认可。大模型具有强大的泛化能力，可以在大规模数据上进行训练，从而获得更高的准确率和更广泛的应用领域。同时，大模型也具有强大的表达能力和灵活性，可以适应各种不同的任务和场景。

1.1.1 大模型的“涌现”

在数字时代的浩瀚星空中，大模型如同新星般以其独特的光芒和力量，照亮了人工智能的未来之路。它们的出现，不仅是技术进步的象征，更是对人类智慧的一次深刻模拟和扩展。

从传承来看，大模型的研究与深度学习的研究是紧密相连的，它们之间的关系仿佛血脉相连，这种关系的起源可以追溯至 20 世纪 80 年代。在那个时代，反向传播算法的提出与应用激活了多层感知机（Multi-Layer Perceptron，MLP）的训练可能性，这就好像一场瑞雪，预示着深度学习春天的到来。然而，由于受到当时计算机算力和数据规模的限制，深度学习仍然像一朵含苞待放的花蕾，尚未能取得突破性的进展。

进入 21 世纪，技术的车轮滚滚向前，为深度学习的发展揭开了新的篇章。2006 年，Hinton 等人正式提出了深度学习的概念，他们巧妙地运用无监督预训练的方法，解决了深层网络训练中的梯度消失难题。这一创新如同阳光雨露，滋润了深度学习这朵待放的花蕾，使其渐渐繁荣起来。尤其值得一提的是，2012 年 Hinton 领导的团队凭借深度学习模型 AlexNet 在 ImageNet 图像识别挑战赛中一举夺冠，这无疑在全球范围内造成了极大的震动，让人们看到了深度学习的无穷潜力。

深度学习模型的规模在此基础上持续攀升，催生了大模型的问世。大模型的出现得益于两方面的推动力：一方面是 GPU、TPU 等专用硬件的出现提升了算力，这就好比将汽车的发动机升级为火箭发动机，为大规模模型训练提供了可能；另一方面是互联网大数据的爆炸式增长为模型训练提供了海量的数据支持，这就如同将小溪的水流汇集成为大海的波涛。在这两大推动力的共同作用下，大模型如雨后春笋般涌现，其中最具里程碑意义的是 Transformer 结构的提出（2017 年由 Vaswani 等人在论文 *Attention is All You Need* 中提出，并在自然语言处理领域中得到广泛应用），它使得深度学习模型的参数突破了 1 亿大关，这无疑标志着我们已经迈入了大模型时代。

大模型之所以被冠以“大”之名，是因为它们的规模和能力相比于普通模型来说是巨大的。它们不再局限于完成简单和特定的任务，而是能够完成更加复杂和高级的任务，例如自然语言理解、语音识别、图像识别等，这些任务都需要大量的数据和计算资源才能完成。大模型使我们在面对复杂和具有挑战性的问题时，有了更强大的工具和技术支持。

大模型的架构与普通模型相比，具有更加复杂和庞大的网络结构，更多的参数和更深的层数，这就好比一座摩天大楼与一间平房的区别。这种复杂性使得大模型能够处理和学习更复杂、更高级的模式和规律，从而在各种任务中产生出乎意料的优秀表现。而这正是大模型的涌现能力的体现，也是大模型最具魅力的地方。大模型在不同任务产生“涌现”现象的参数量比较如图 1-1 所示。

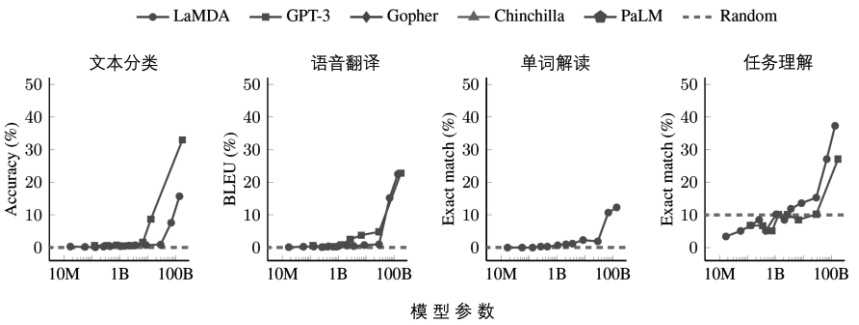

图 1-1　大模型在不同任务产生“涌现”现象的参数量比较

随着模型参数的递增，准确率仿佛经历了一场蜕变，模型在某一刹那“突然”就实现了跨越式的提升。这种变化可以简单地理解为量变引发质变——当模型的规模突破某个阈值时，精度的增速由负转正，呈现出一种异于常规的增速曲线，如同抛物线突破顶点，扶摇直上。因此，在模型规模与准确率的二维空间中，我们可以观察到一条非线性增长的轨迹，这是大模型所独有的魅力。

这种精度增速现象的涌现，不仅体现在数字的提升上，更在于模型所展现出的更高层次的抽象能力和泛化能力。换句话说，大模型在处理复杂任务时，能够捕捉到更深层次的数据模式和规律，从而给出更准确、更全面的预测和判断。这种涌现能力的出现并非偶然，而是有其深刻的内在逻辑。

首先，更复杂的神经网络结构是大模型涌现能力的重要基石。随着模型规模的扩张，神经元之间的连接逐渐丰富和深化，形成了一个错综复杂但有序的网络结构。这样的结构使得模型能够更好地挖掘输入数据中的高层次特征，将原始数据转换为具有丰富语义信息的特征向量，从而提高模型的表现能力。

其次，更多的参数意味着模型具备了更强的表达能力。大型模型通常拥有数以亿计的参数，这些参数为模型提供了巨大的自由度，使其能够对输入数据进行各种复杂的非线性变换。在自然语言处理领域，大语言模型（Large Language Model，LLM）正是凭借这种强大的表达能力，通过对海量文本数据的深度训练，学习到了语言背后的抽象特征和规律，从而能够生成流畅、自然的文本内容。

最后，更强的数据驱动能力是大模型涌现能力的关键所在。大型模型的训练过程往往需要海量的数据支持，这使得它们能够充分吸收和利用数据中的信息，学习到更为普遍和更加鲁棒的特征和规律。这种数据驱动的学习方式不仅提高了模型在训练任务上的表现，更重要的是赋予了模型在面对新任务时的强大适应能力和泛化能力。

本书将以大模型的涌现能力为切入点，带领读者深入探索深度学习大模型的内在机理和应用技巧。我们将通过源码精讲的方式，逐一剖析大模型的核心组件和工作原理，让读者对大型神经网络有一个全面而深入的了解；同时，还将介绍一系列高效的大模型应用开发和微调方法，帮助读者更好地利用这些巨型智能工具来解决实际问题。在这个过程中，我们将带领读者领略深度学习大模型的魅力和潜力，以及它们为人工智能领域带来的巨大变革和影响。

1.1.2　深度学习与大模型的起源

随着技术的日新月异，深度学习与大模型逐渐成为自然语言处理等领域的主流方法，它们不仅引领了人工智能技术的新潮流，更为我们的未来描绘出了一幅充满无限可能的画卷。

Google 的 BERT 和 OpenAI 的 GPT-3 是这一时代的杰出代表。BERT 全名为 Bidirectional Encoder Representations from Transformers，是基于 Transformer 的一个预训练语言模型。自 2018 年发布以来，凭借其在自然语言理解和自然语言生成任务中的卓越性能，BERT 已经成为 NLP 领域的新里程碑。与此同时，GPT-3 作为 OpenAI 在 2020 年的杰出作品，拥有惊人的 1750 亿模型参数，展现了在自然语言生成任务中出色的生成能力和泛化能力，成为当时最强大的语言模型之一。

深度学习与大模型的成功并非偶然。在众多机构和企业的推动下，各种大模型如雨后春笋般涌现出来。Facebook 的 RoBERTa、微软的 MT-DNN 等大模型都在自然语言处理、计算机视觉、语音识别等领域取得了显著进展，为人工智能技术的发展注入了新的活力。尤其值得一提的是，2021 年 Google 的 Switch Transformer 首次突破了万亿规模，同年 12 月推出的 1.2 万亿参数 GLaM 通用大语言模型再次刷新了纪录，展现了人工智能技术的巨大潜力。

大模型的影响力已经跨越了自然语言处理领域，对计算机视觉、语音识别等领域也产生了深远影响。这些领域的突破，不仅提升了人工智能技术的整体水平，更为我们的日常生活带来了前所未有的便利。例如，通过大模型的帮助，自然语言翻译变得越来越准确和流畅，智能客服能够更好地理解我们的需求并提供满意的解答，个人助理可以更加智能地管理我们的日程和生活。

尽管大模型的训练仍需大量的数据和计算资源，但随着技术的进步，其训练和应用正变得越来越可行和普遍。云计算、边缘计算等新技术的发展，为大模型的训练和应用提供了强大的基础设施支持。同时，新的算法和优化技术也在不断降低大模型的训练成本和提高其效率。

展望未来，我们有理由相信，在技术的持续推动和创新下，深度学习与大模型将继续为人工智能领域书写新的辉煌。随着模型规模的进一步扩大和算法的不断优化，我们可以期待大模型在自然语言处理、计算机视觉、语音识别等领域取得更加卓越的性能。同时，随着人工智能技术的不断发展和社会应用的不断深化，我们可以期待更多新的应用场景和商业模式涌现出来。

总的来说，深度学习与大模型的成功是人工智能技术发展的一个重要里程碑。它们不仅为我们提供了强大的工具和技术支持，更为我们的未来描绘出了一幅充满无限可能的画卷。在这个新时代里，我们有理由相信，深度学习与大模型将继续引领人工智能技术的发展潮流，不断地为我们带来更多的科技奇迹。

1.1.3 大模型的概念与特点

在人工智能领域，大模型犹如一颗璀璨的明珠，指引着技术发展的方向。它们以巨大的参数规模和复杂的计算结构，展现出了前所未有的智能潜力。本节将从大模型的基本概念出发，逐步深入解析其发展历程、特点、分类以及泛化与微调等内容，带领读者一同探寻大模型的奥秘。

1. 大模型的定义

大模型，顾名思义，是指具有大规模参数和复杂计算结构的机器学习模型。这些模型通常由深度神经网络构建而成，参数数量动辄数十亿，甚至数千亿。大模型的设计初衷是提高模型的表达能力和预测性能，使其能够处理更加复杂的任务和数据。在自然语言处理、计算机视觉、语音识别和推荐系统等领域，大模型都展现出了卓越的性能和广泛的应用前景。

2. 大模型的发展历程

大模型的发展经历了萌芽期、探索沉淀期和迅猛发展期三个阶段。在萌芽期，以卷积神经网络（CNN）为代表的传统神经网络模型，为大模型的发展奠定了基础。在探索沉淀期，Transformer 架构的提出奠定了大模型预训练算法架构的基础，使大模型技术的性能得到了显著提升。到了迅猛发展期，大数据、大算力和大算法的完美结合，大幅提升了大模型的预训练和生成能力以及多模态多场景应用能力，以 GPT 为代表的大模型更是在全球范围内引起了广泛关注。

3. 大模型的特点

相对于普通的深度学习模型，大模型的特点更为突出，一般包括以下几点：

- 巨大的规模：大模型包含数十亿个参数，模型大小可以达到数百吉字节甚至更大，这使得大模型具有强大的表达能力和学习能力。
- 涌现能力：当模型的训练数据突破一定规模时，大模型会突然涌现出之前小模型所没有的、意料之外的复杂能力和特性，展现出类似人类的思维和智能。
- 更好的性能和泛化能力：大模型在各种任务上表现出色，包括自然语言处理、图像识别、语音识别等，具有强大的泛化能力。

- 多任务学习：大模型可以同时学习多种不同的任务，如机器翻译、文本摘要、问答系统等，这使得模型具有更广泛的语言理解能力。
- 依赖大数据和计算资源：大模型需要海量的数据进行训练，同时需要强大的计算资源来支持模型的训练和推理过程。

4. 大模型的分类

根据输入数据类型和应用领域的不同，大模型主要分为语言大模型、视觉大模型和多模态大模型三类。

- 语言大模型主要用于处理文本数据和理解自然语言。
- 视觉大模型主要用于图像处理和分析。
- 多模态大模型能够处理多种不同类型的数据，如文本、图像、音频等。

此外，按照应用领域的不同，大模型还可以分为通用大模型、行业大模型和垂直大模型三个层级。

- 通用大模型：可以在多个领域和任务上通用。
- 行业大模型：针对特定行业或领域进行预训练或微调
- 垂直大模型：针对特定任务或场景进行预训练或微调。

5. 大模型的泛化与微调

大模型的泛化能力是指模型在面对新的、未见过的数据时，能够正确理解和预测这些数据的能力。为了提高模型的泛化能力，通常需要对模型进行微调（Fine-tuning）。

微调是一种利用少量带标签的数据，对预训练模型进行再次训练的方法，以适应特定任务。在微调过程中，模型的参数会根据新的数据分布进行调整，从而提高模型在新任务上的性能和效果。

可以预见，大模型是未来人工智能发展的重要方向和核心技术。随着 AI 技术的不断进步和应用场景的不断拓展，大模型将在更多领域展现出惊人的能力，推动人类社会迈向更加美好的未来。

1.1.4　大模型开启了深度学习的新时代

近十年来，“深度学习+大算力”已成为实现人工智能的主流技术途径，通过这一方式训练得出的模型，在全球掀起了“大练模型”的热潮，并催生出众多的人工智能公司。然而，深度学习技术出现的这十年间，模型大多针对特定场景进行训练，即小模型依然沿用传统的定制化、作坊式的开发方式。这种方式需要完成从研发到应用的全方位流程，包括需求定义、数据收集、模型算法设计、训练调优、应用部署和运营维护等一系列阶段。因此，除了需要产品经理准确定义需求外，还需要人工智能研发人员具备扎实的专业知识和协同合作能力，以应对大量复杂的工作。

相较于传统模型，大模型的优势在于其具备通用能力。通过从海量、多类型的场景数据中学习，大模型能够总结出不同场景、不同业务的通用特征和规律，进而成为具有泛化能力的模型库。在应对新的业务场景或基于大模型开发应用时，可以对大模型进行适配，例如，利用小规模标注数据进行二次训练，或者无须自定义任务即可完成多个应用场景。因此，大模型的通用能力能够有效应对多样化、碎片化的人工智能应用需求，为大规模人工智能落地应用提供了可能。同时，作为一种新

型的算法和工具，大模型正在成为人工智能技术新的制高点和基础设施。

值得一提的是，大模型的变革性技术特性，显著提升了人工智能模型在应用中的性能表现。它能够将人工智能的算法开发过程，由传统的烟囱式开发模式转向集中式建模。通过这种转变，大模型解决了人工智能应用落地过程中的一些关键痛点，包括场景碎片化、模型结构零散化和模型训练需求零散化等问题。这为我们在新时代探索和应用人工智能技术指明了方向，并奠定了坚实的基础。

随着大模型的出现和应用，深度学习技术的发展进入了一个全新的阶段。传统的模型开发方式针对特定场景进行训练，在面对多样化、碎片化的人工智能应用需求时显得力不从心。而大模型的出现则打破了这个局限，通过从海量数据中学习并总结出通用特征和规律，具备了应对各种场景和业务的通用能力。

大模型的优势不仅在于其通用能力，更在于其带来的开发模式的变革。传统的烟囱式开发模式，每个项目都需要从头开始，导致大量的人力、物力和时间成本的浪费。而集中式建模的方式，通过复用和共享大模型的能力，可以极大地提升开发效率，降低成本，同时也提高了模型的性能表现。

此外，大模型的出现也为人工智能技术的发展开辟了新的可能性。它不仅可以应对现有的业务场景，更可以预见和适应未来的需求。大模型的通用能力和高性能表现，使其可以作为一种基础设施，支撑起整个人工智能技术的发展和应用。

总之，大模型的出现是深度学习技术发展的重要里程碑。它不仅提升了模型的性能表现，更改变了我们的开发模式和应用方式。在新时代的人工智能技术探索和应用中，我们将更加依赖于大模型的力量，去揭示和理解这个世界的复杂性和多样性。因此，我们有理由相信，随着大模型的进一步发展和应用，人工智能技术的未来将更加光明和广阔。

1.2 为什么要使用大模型

随着 OpenAI 引领的超大模型风潮，大模型的发展日新月异。在现今的科技舞台上，每周，甚至每一天，我们都能见证到一个全新模型的开源，这些模型的创新性和实用性不断超越前作，彰显出深度学习的无穷潜力。

更重要的是，随着技术的进步和方法的优化，大模型的微调训练成本也大大降低，使得更多的研究者和实践者有机会亲自体验和使用这些大型模型。就如同原本昂贵的奢侈品逐渐走入寻常百姓家，大模型也从曲高和寡的研究领域逐渐扩展到了更广泛、更接地气的应用场景。笔者总结了目前大模型的一些分类及其说明，如下所示：

- 主流大模型：GLM-130B、PaLM、BLOOM、Gopher、Chinchilla、LaMDA、CodeGeeX、CodeGen。
- 分布式训练：3D 并行（包括张量并行、流水线并行、数据并行）、DeepSpeed、混合精度、Megatron-DeepSpeed。
- 微调：FLAN、LoRA、DeepSpeed。
- 应用：工具（包括 Toolformer、ART）。

这种发展趋势不仅预示着大模型将在更多领域得到应用，更重要的是，它为人工智能技术的生活化铺平了道路，使得更多的人可以享受到深度学习带来的便利和乐趣。未来，我们可以期待大模

型在医疗、教育、娱乐等各个领域发挥出更大的作用，为我们的生活带来更多的便利和惊喜。

可以看到，大模型的开源和微调训练成本的降低，是深度学习领域的一大进步，也是人工智能技术发展的重要里程碑。这不仅为我们提供了更多的工具和可能性，更为我们的未来描绘出了一幅充满希望和机遇的画卷。在这个新时代里，我们有理由期待大模型将继续引领深度学习的发展潮流，为我们的生活和社会带来更多的正面影响。

1.2.1　大模型与普通模型的区别

从上一节我们了解到，大模型是指网络规模巨大的深度学习模型，具体表现为模型的参数量规模较大，其规模通常在百亿级别。随着模型参数的提高，人们逐渐接受模型参数越大其性能越好的特点，但是，大模型与普通深度学习模型之间到底有什么区别呢？

简单地解释，可以把普通模型比喻为一个小盒子，它的容量是有限的，只能存储和处理有限数量的数据和信息。这些模型可以完成一些简单的任务，如分类、预测和生成等，但是它们的能力受到了很大的限制。

表 1-1 列出了目前可以公开使用的大模型版本和参数量。

表 1-1　公开使用的大模型版本和参数量

Model	开　　源	参数量(Billion)	类　　型	是否开源
LLaMa	Meta AI	65	Decoder	open
OPT	Meta AI	175	Decoder	open
T5	Google	11	Encoder-Decoder	open
mT5	Google	13	Encoder-Decoder	open
UL2	Google	20	Encoder-Decoder	open
PaLM	Google	540	Decoder	no
LaMDA	Google	137	Decoder	no
FLAN-T5	Google	同 T5	Encoder-Decoder	open
FLAN-UL2	Google	同 UL2	Encoder-Decoder	open
FLAN-PaLM	Google	同 PaLM	Decoder	no
FLAN	Google	同 LaMDA	Decoder	no
BLOOM	BigScience	176	Decoder	open
GPT-Neo	EleutherAI	2.7	Decoder	open
GPT-NeoX	EleutherAI	20	Decoder	open
GPT3	OpenAI	175	Decoder	no
InstructGPT	OpenAI	1.3	Decoder	no

相比之下，大模型就像一个超级大的仓库，它能够存储和处理大量的数据和信息。它不仅可以完成普通模型能完成的任务，还能够处理更加复杂和庞大的数据集。这些大模型通常由数十亿，甚至上百亿个参数组成，需要大量的计算资源和存储空间才能运行。这类似于人类大脑（约有 1 000 亿个神经元细胞），在庞大的运算单元支撑下，完成更加复杂和高级的思考和决策。

1.2.2 为什么选择 ChatGLM

ChatGLM 系列是国产大语言模型中性能最好、回答准确率最高的大模型。

智谱 AI 第一代 ChatGLM-6B 在 2023 年 3 月推出，开源模型推出后不久就获得了很多的关注和使用。到 2023 年 6 月，ChatGLM2 发布，再次引起了业界广泛的关注。ChatGLM Logo 如图 1-2 所示。

图 1-2 ChatGLM Logo

2023 年的 10 月 27 日，智谱 AI 再次发布第三代基础大语言模型 ChatGLM3 系列。本次发布的第三代模型共包含 3 个：基础大语言模型 ChatGLM3-6B-Base、对话调优大语言模型 ChatGLM3-6B 和长文本对话大语言模型 ChatGLM3-6B-32K。

ChatGLM 的独特之处在于，它不仅是一个语言模型，更是一个具备深度思考能力的语言专家。它能够理解并解析复杂的语言结构，对语义的理解更加精准，从而在回答问题、解决问题时更具针对性。同时，ChatGLM 还具备了出色的记忆能力，可以记住与它交流的每一个细节，实现个性化的交流体验。在每一次交流中，它都能根据用户的喜好和需求，提供更加贴心、高效的服务。ChatGLM3 系列模型除了基本对话能力的提升外，还有诸多支持：

- 更强的代码执行能力：即 Code Interpreter。ChatGLM3 的代码增强模块 Code Interpreter 根据用户需求生成代码并执行，自动完成数据分析、文件处理等复杂任务。
- 网络搜索增强 WebGLM：接入搜索增强，能自动根据问题在互联网上查找相关资料，并在回答时提供相关参考文献或文章链接。
- 全新的 Agent 智能体能力：ChatGLM3 集成了自研的 AgentTuning 技术，AI Agent 水平比第二代提升 1000%。关于 AgentTuning，可以参考网络文章“如何提高大语言模型作为 Agent 的能力？清华大学与智谱 AI 推出 AgentTuning 方案”。Agent 能力非常依赖规划和推理，从公布的结果看，ChatGLM3 在 GSM8K 等数学逻辑推理方面的评测结果已经超过 GPT-3.5，因此对于 Agent 的支持理论上应该非常棒。
- 多模态能力：官方宣称具有多模态理解能力的 CogVLM，可以看图识语义，在 10 余个国际标准图文评测数据集上取得了 SOTA（state-of-the-art，最先进的结果）。
- 端侧推理：ChatGLM3 推出可手机部署的端测模型 ChatGLM3-1.5B 和 ChatGLM3-3B，支持在手机端调用，速度可以达到 20 tokens/s。一般成年人阅读的速度是每秒 2~5 个单词，完全足够，而且官方宣称自己的 ChatGLM3-1.5B 和 ChatGLM3-3B 与 ChatGLM2-6B（即第二代）水平差不多。

ChatGLM 系列是非常具有影响力的国产大语言模型系列，从 2023 年 3 月份开源第一代，到 2023 年 10 月迭代到第三代，发展十分迅猛，而且它在 AI Agent、代码执行、多模态等方面都有非常好的布局和提升，十分值得大家关注。

可以预见，ChatGLM 不仅可以作为一个自然语言处理大模型，还可以广泛应用于其他的应用场景，例如教育辅导、智能客服、智能助手、智能写作等多个领域，为人们的生活带来极大的便利。

在教育领域，ChatGLM 发挥了重要的作用。它能够根据学生的提问和需求，提供精准、及时的解答。同时，ChatGLM 还可以根据学生的学习情况和兴趣爱好，提供个性化的学习建议和资源推荐。这使得教育更加智能化、个性化，从而提高学生的学习效果和兴趣。

在智能客服领域，ChatGLM 以其高效、精准的回答能力，解决了传统客服面临的种种问题。它能够快速、准确地理解用户的问题和需求，提供有针对性的解决方案。这大大提高了客服效率和服务质量，提升了用户的满意度和忠诚度。

在智能助手领域，ChatGLM 可以帮助人们完成各种任务，如订餐、购物、日程管理等。通过自然语言交互，用户可以轻松地与助手进行交流，实现快速、便捷的生活体验。

在智能写作领域，ChatGLM 可以帮助人们快速生成文章、报告等文本内容。通过输入关键词或主题，用户可以轻松地获得高质量的文本内容，从而提高写作效率和准确性。

ChatGLM 模型以其卓越的性能和广泛的应用，展现了人工智能领域的强大潜力和无限可能性。作为一款大语言模型，它不仅具备了深度思考能力、精准语义理解能力和个性化交流体验能力等多种优势，还广泛应用于智能客服、智能助手、教育辅导等多个领域。这使得 ChatGLM 成为人工智能领域中的一颗璀璨明珠，为人类社会带来了诸多便利和改变。

1.2.3　大模型应用场合与发展趋势

在人工智能的广袤星空中，大模型犹如一颗璀璨的星辰，引领着深度学习领域的前行。从自然语言处理的源头出发，它们以注意力机制为核心基石，逐渐延伸至 ChatGLM 等巍峨之作，其参数之巨已至千亿、万亿之域。与此同时，训练数据的海洋也在不断扩张，为模型的成长提供了丰沃的土壤，推动着人工智能从对外界的简单感知向深度认知跃进。

大模型之美，在于它能从繁杂多变的场景中汲取智慧，从海量数据中提炼出通用的特征和规律，进而构建一个具有高度泛化能力的模型宇宙。当面对新的业务挑战时，这个大模型宇宙可以轻松地进行自我适配，或是借助少量的标注数据进行微调，或是无须任何定制即可应对多个应用场景，展现出通用的智能魅力。这种通用性，为应对多样且零碎的人工智能需求提供了一把钥匙，为人工智能的大规模落地应用开辟了一条康庄大道。

在制造业领域中，大模型正施展其魔法，将研发、销售及售后的每一个环节都点石成金。在研发环节，它借助 AI 生成图像或 3D 模型技术，为产品设计、工艺设计、工厂设计等流程注入新的活力。在销售和售后环节，它则创造出更加懂客户、更加个性化的智能客服和数字人带货主播，让销售和服务的效率和质量都迈上了一个新的台阶。

在医疗领域中，大模型也在默默奉献。它助力提升医疗服务的效率，从呼叫中心的自动分诊到常见病的问诊辅助，再到医疗影像的解读助手，它都在默默发光发热。此外，它还通过合成数据为医学研究提供强大的支持，为解决部分辅助医疗设备的匮乏问题贡献自己的力量。

金融行业同样在大模型的支持下蓬勃发展。银行业通过智慧网点、智能服务、智能风控等场景应用大模型技术，实现了业务的智能化升级；保险业则借助智能保险销售助手、智能培训助手等工具提高了工作效率；证券期货业也利用大模型在智能投研、智能营销等方面取得了显著成果。

在传媒与互联网领域中，大模型更是掀起了一场革命。它大幅提升了文娱内容的生产效率，降

低了成本，让更多的人能够享受到高质量的文娱产品。从更深远的角度来看，大模型有望颠覆传统的互联网业态和场景入口，取代传统搜索引擎的地位，为我们提供更加高效、便捷的信息获取方式和交互体验。

可以相信，在不久的将来，大模型将在更多领域展现出惊人的能力，推动人类社会迈向更加美好的未来。而我们也将继续努力，传播先进技术理念和实践经验，为科技进步贡献自己的力量。

1.3 本章小结

本章作为本书的开篇，犹如一幅壮丽的画卷，将大模型、人工智能以及深度学习的基本内容呈现在读者眼前。大模型在人工智能领域中具有广泛应用前景，尤其在图像生成、自然语言处理和音频生成等领域中，它如同明亮的灯塔，照亮了我们探索的方向。

随着人工智能领域的持续进步和深度学习技术的日臻成熟，大模型如同智慧的巨人，在各个领域中展现出无穷的魅力。在自然语言处理领域，大模型已化身为自动文本摘要的巧匠、对话系统的智者以及机器翻译的桥梁；在图像处理领域，大模型又变身为图像生成的艺术家、风格转换的魔术师以及图像修复的工匠；在音频处理领域，大模型则成为了语音合成的歌者、音乐生成的创作者。

深度学习技术的不断演进，预示着人工智能与大模型的未来将更加灿烂辉煌。我们热切期待大模型在更多领域中绽放智慧之光，同时，也期待更多创新的生成式模型技术破土而出，为人工智能领域的发展添砖加瓦。本书如同一把钥匙，旨在为读者打开从零开始学习大模型基本原理和实现方法的大门，引领读者深入探究大模型的应用，并感受它在人工智能领域中的辉煌前景。

第 2 章

PyTorch 2.0 深度学习环境搭建

工欲善其事，必先利其器。对于任何一位想要构建深度学习应用程序或是将训练好的模型应用到具体项目的读者，都需要使用编程语言来实现设计意图。在本书中，将使用 Python 语言作为主要的开发语言。

Python 之所以在深度学习领域中被广泛采用，这得益于许多第三方提供的集成了大量科学计算类库的 Python 标准安装包，其中最常用的便是 Miniconda。Python 是一种脚本语言，如果不使用 Miniconda，那么第三方库的安装可能会变得相当复杂，同时各个库之间的依赖性也很难得到妥善的处理。因此，为了简化安装过程并确保库之间的良好配合，推荐安装 Miniconda 来替代原生的 Python 语言安装。

PyTorch 是一种开源的深度学习框架，由 Facebook 的人工智能研究团队开发。它提供了两个高级功能：

- 强大的 GPU 加速的张量计算（类似于 NumPy）。
- 基于深度神经网络的自动求导系统。

PyTorch 的主要特点是动态计算图，这意味着计算图可以在每个运行时刻动态改变，这大大提高了模型的灵活性和效率。

除此之外，PyTorch 还提供了丰富的 API，支持多种深度学习的模型和算法，并能够轻松与其他 Python 库（例如，NumPy 和 SciPy）进行交互。

目前，PyTorch 已广泛应用于学术研究和商业开发，包括自然语言处理、计算机视觉、生成对抗网络（GANs）等领域，是全球最受欢迎的深度学习框架之一。

在本章中，首先将引导读者完成 Miniconda 的完整安装。然后，将通过一个实践项目来帮助读者熟悉 PyTorch 2.0。这个项目将生成可控的手写数字，作为一个入门级的程序，它将帮助读者了解一个完整的 PyTorch 项目的工作流程。通过这个项目，读者将能够初步体验到 PyTorch 2.0 的强大功能和灵活性。

2.1 安装 Python 开发环境

2.1.1 Miniconda 的下载与安装

第一步：下载和安装

（1）在 Miniconda 官网打开下载页面，如图 2-1 所示。

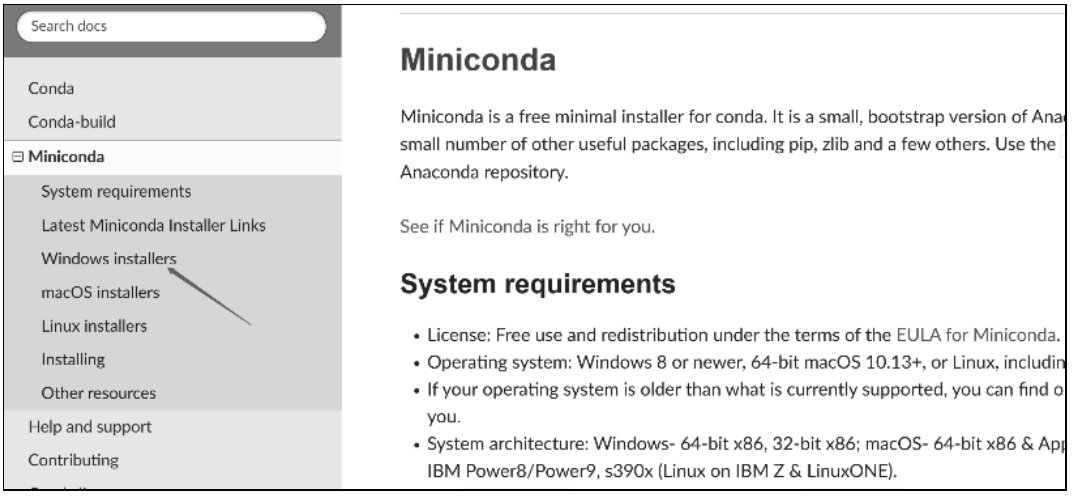

图 2-1 Miniconda 下载页面

目前提供的是最新集成了 Python 3.11 64-bit 版本的 Miniconda，如果读者使用的是以前的 Python 版本，例如 Python 3.10，也是完全可以的，读者可以根据自己的操作系统选择下载。

这里笔者推荐使用 Windows Python 3.11 64-bit 版本，可以到 Miniconda 官网下载，如图 2-2 所示。

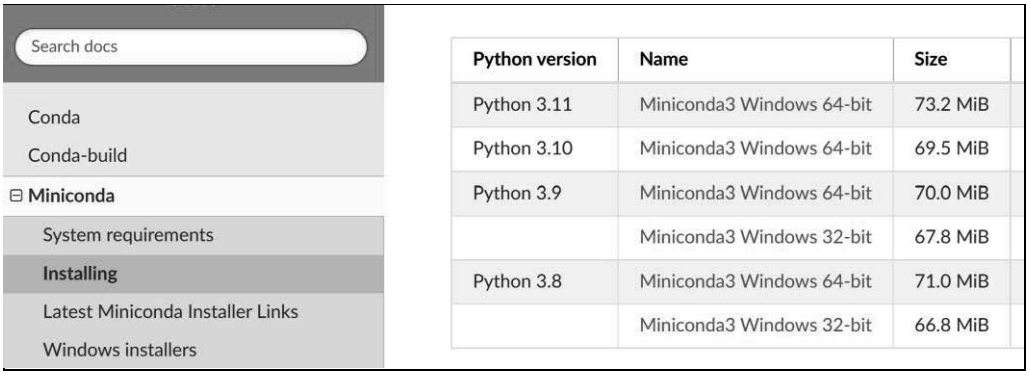

图 2-2 Miniconda 官网提供的下载

（2）下载完成后得到的是 EXE 文件，直接运行即可进入安装过程。安装完成以后，出现如图 2-3 所示的目录结构，说明安装正确。

图 2-3　Miniconda 安装目录

第二步：打开控制台

在计算机桌面依次单击“开始”→“所有程序”→“Miniconda3”→“Miniconda Prompt (Miniconda3)”，打开 Miniconda Prompt 窗口，它与 CMD 控制台类似，输入命令就可以控制和配置 Python。在 Miniconda 中最常用的是 conda 命令，该命令可以执行一些基本操作，读者可以自行测试一下这个命令。

第三步：验证 Python

在 Miniconda Prompt 窗口中输入 python，如果安装正确，会打印出 Python 版本号以及控制符号。在控制符号下输入代码：

```
print("hello Python")
```

输出结果如图 2-4 所示。

```
(base) C:\Users\xiaohua>python
Python 3.11.4 | packaged by Anaconda, Inc. | (main, Jul  5 2023, 13:47:18) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello Python")
hello Python
>>>
```

图 2-4　验证 Miniconda Python 是否安装成功

第四步：使用 pip 命令

使用 Miniconda 的好处在于，它能够很方便地帮助读者安装和使用大量第三方类库。本书中，我们将使用 pip 命令安装第三方类库。查看已安装的第三方类库的命令如下：

```
pip list
```

注意：如果此时命令行还处于>>>状态，可以输入 exit()退出。

在 Miniconda Prompt 控制台输入 pip list 命令，结果如图 2-5 所示。

```
(base) C:\Users\xiaohua>pip list
WARNING: Ignoring invalid distribution -qdm (c:\miniforge3\lib\site-packages)
WARNING: Ignoring invalid distribution -harset-normalizer (c:\miniforge3\lib\site-packages)
WARNING: Ignoring invalid distribution -ensorflow-gpu (c:\miniforge3\lib\site-packages)
Package                       Version
----------------------------- -----------------------
absl-py                       1.0.0
aiofiles                      0.8.0
aiohttp                       3.8.1
aiosignal                     1.2.0
alabaster                     0.7.12
altair                        4.2.0
altgraph                      0.17.2
anyio                         3.5.0
argon2-cffi                   21.1.0
arrow                         1.1.1
```

图 2-5　列出已安装的第三方类库

在 Miniconda 中安装第三方类库的命令如下：

```
pip install name
```

这里的 name 是需要安装的第三方类库名，假设需要安装 NumPy 包（这个包已经安装过），那么输入的命令就是：

```
pip install numpy
```

这个安装过程略去，请读者自行尝试。使用 Miniconda 的好处就是默认已安装好了大部分学习所需的第三类库，这样避免了使用者在安装和使用某个特定类库时，可能出现的依赖类库缺失的情况。

2.1.2　PyCharm 的下载与安装

和其他语言类似，Python 程序的编写可以使用 Windows 自带的编辑器。但是这种方式对于较为复杂的程序工程来说，容易混淆相互之间的层级和交互文件，因此在编写程序工程时，我们建议使用专用的 Python 编译器 PyCharm。

第一步：PyCharm 的下载和安装

（1）进入 PyCharm 官网的 Download 页面，选择不同的版本，如图 2-6 所示，PyCharm 有收费的专业版和免费的社区版，这里建议读者选择免费的社区版即可。

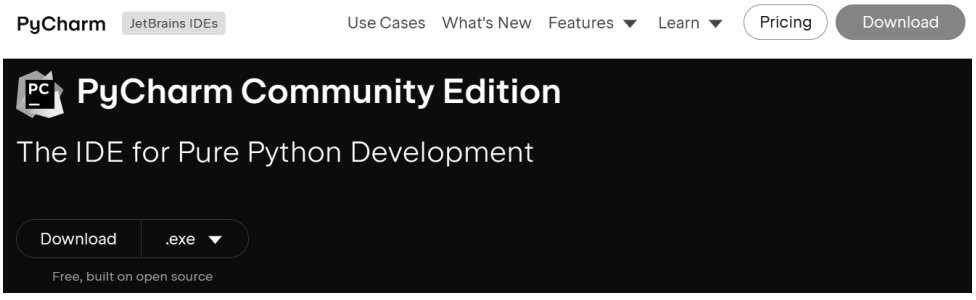

图 2-6　PyCharm 的免费版

（2）下载 PyCharm 安装文件后，双击运行进入安装界面，如图 2-7 所示。直接单击 Next 按钮，

采用默认安装即可。

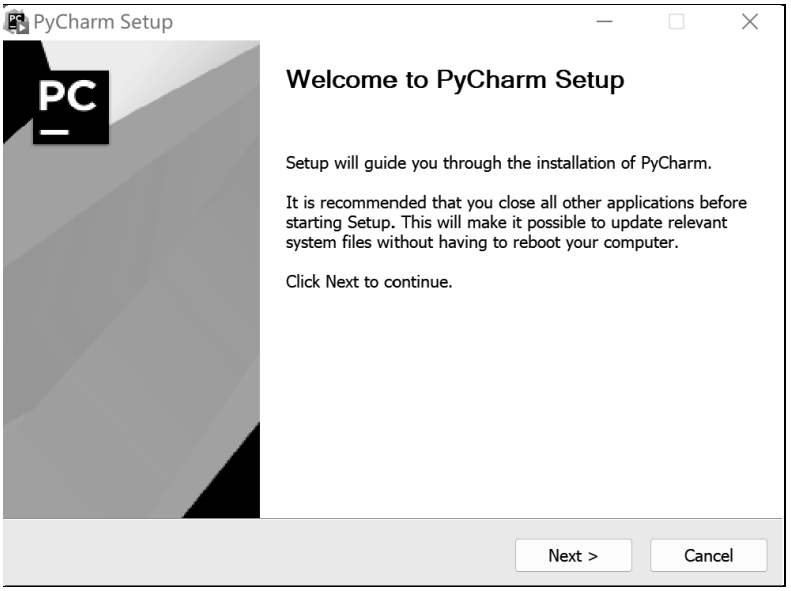

图 2-7　PyCharm 的安装文件

（3）在安装 PyCharm 的过程中需要对安装的参数进行选择，如图 2-8 所示，这里建议直接使用默认安装即可。

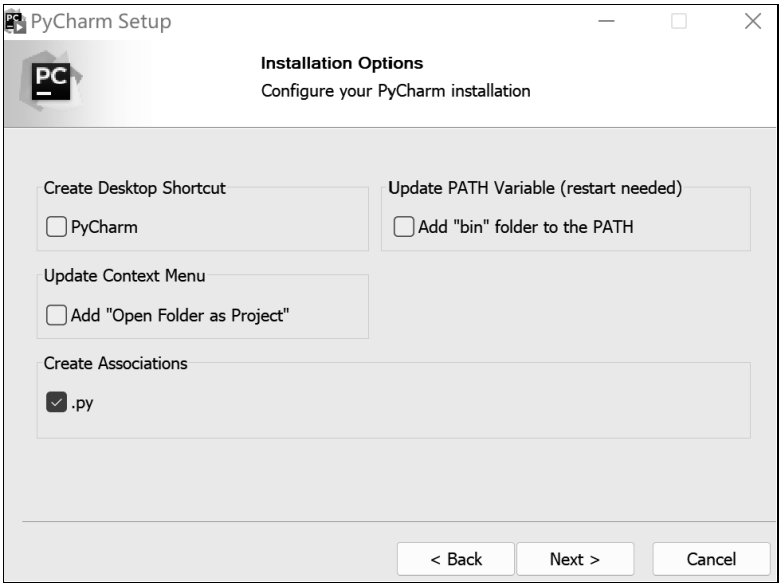

图 2-8　PyCharm 的配置选择（按个人真实情况选择）

（4）安装完成后出现 Finish 按钮，单击该按钮安装完成，如图 2-9 所示。最后将在桌面上显示一个 PyCharm 程序图标，双击该图标可运行 PyCharm。

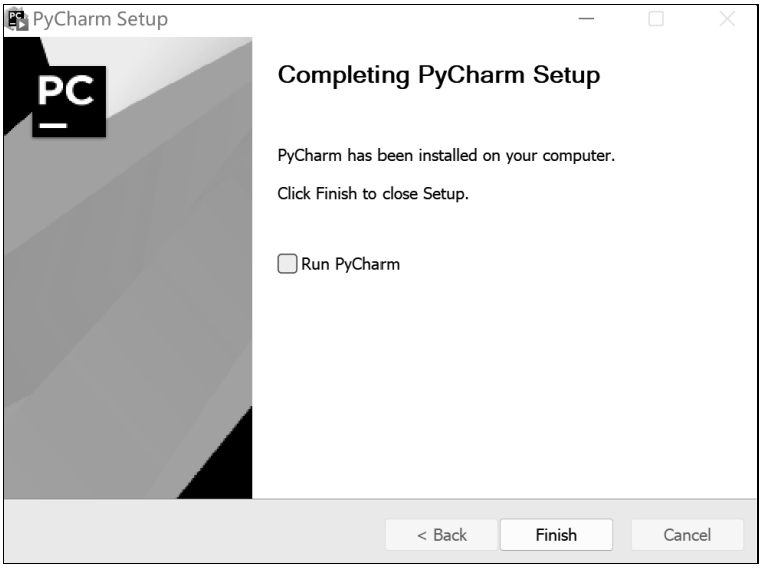

图 2-9　PyCharm 安装完成

第二步：使用 PyCharm 创建程序

（1）单击桌面上新生成的图标进入 PyCharm 程序界面。由于是第一次启动 PyCharm，需要接受相关的协议，在勾选界面下方的复选框后单击 Continue 按钮，进行下一步操作。因为操作比较简单，这里就不截图显示了。

（2）进入 PyCharm 工程创建界面创建新的项目，可以直接创建一个新项目（New Project），或者打开一个已有的项目文件夹（Open），如图 2-10 所示。

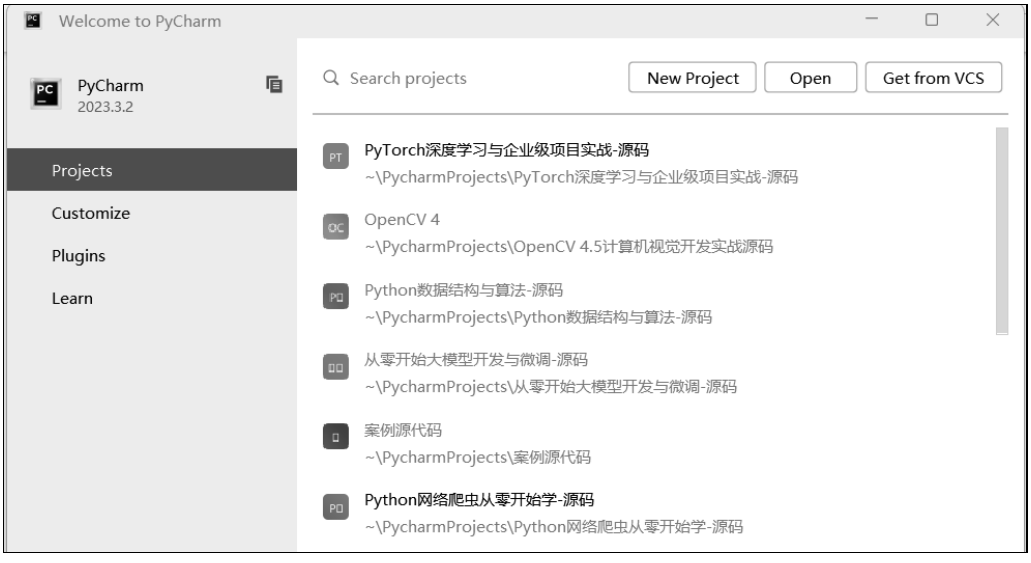

图 2-10　PyCharm 工程创建界面

（3）这里单击 New Project 按钮创建一个新项目，下面就是配置 Python 环境路径，填写好 python.exe 地址后（就是上一步安装的 c:\miniconda3 目录下的 python.exe），单击 Create 按钮，将

在 PyCharm 项目管理目录 PycharmProjects 下面创建一个新项目，如图 2-11 所示。

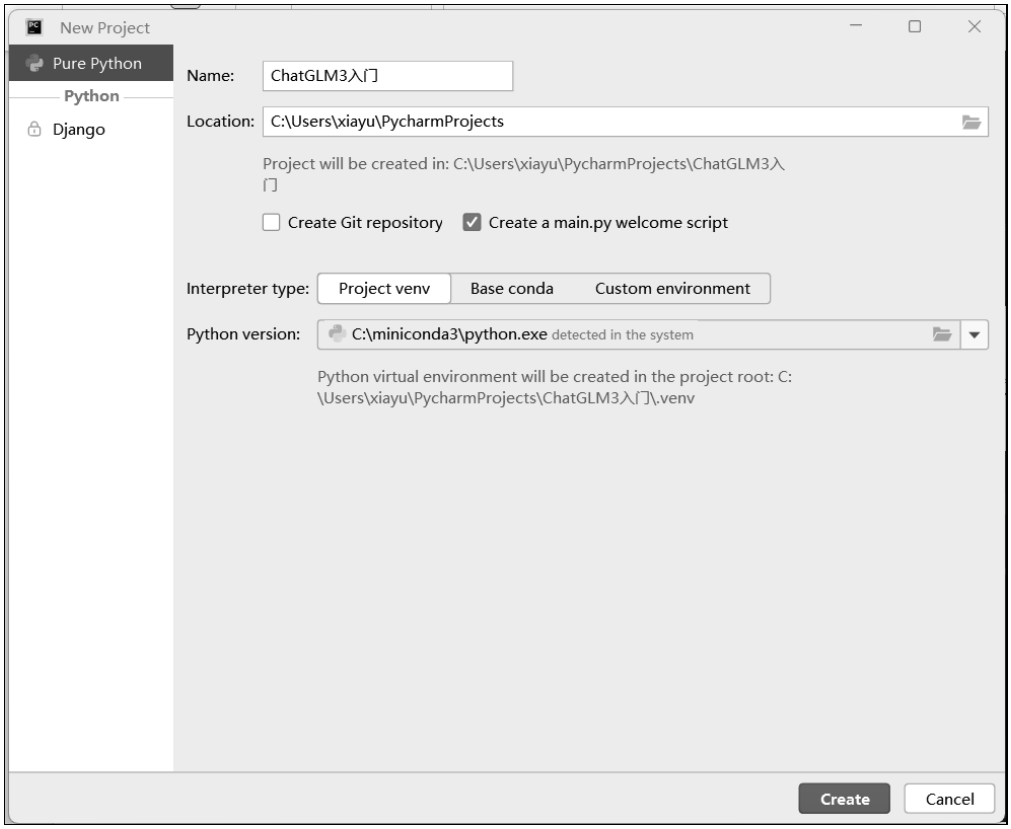

图 2-11　PyCharm 新建文件界面

（4）对于创建的新项目，PyCharm 默认提供了一个测试程序 main.py，内容如图 2-12 所示。

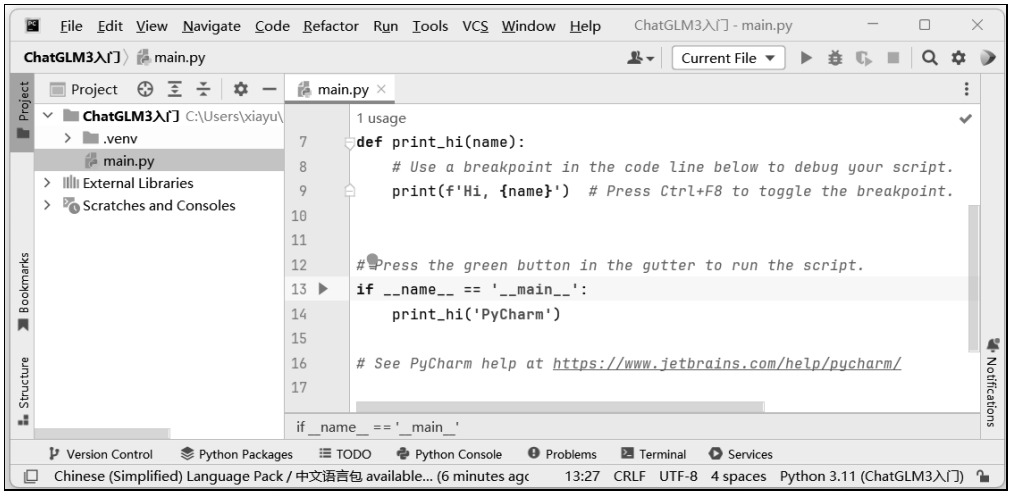

图 2-12　PyCharm 工程运行界面

选中 main.py，单击菜单栏中的 Run|run…运行代码，或者直接右击 main.py 文件名，在弹出的

快捷菜单中选择 run 命令。如果成功，将输出“Hi, PyCharm”，如图 2-13 所示。

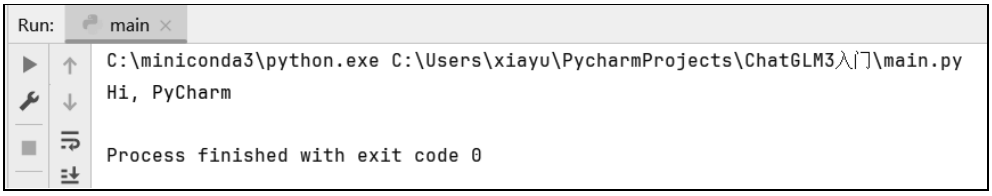

图 2-13　运行成功

至此，Python 与 PyCharm 的配置就完成了。

2.1.3　softmax 函数练习

对于 Python 科学计算来说，最简单的想法就是可以将数学公式直接表达成程序语言，可以说，Python 满足了这个想法。本小节将使用 Python 实现和计算一个深度学习中最为常见的函数——softmax 函数。至于这个函数的作用，现在不加以说明，笔者只是带领读者尝试实现其程序的编写。

softmax 函数的计算公式如下：

$$\text{softmax}(i)=\frac{\exp^{x\_i}}{\sum_{i=0}^{j=N}\exp^{x\_j}}$$

其中，*x_i* 表示输入向量 *x* 中的第 *i* 个元素，*N* 为数据总量，Σ 表示求和符号，exp 表示自然指数函数。

带入 softmax 的结果其实就是先对每一个 *x_i* 进行以 e 为底的指数计算，变成非负，然后除以所有项之和进行归一化，之后每个 *x_i* 就可以解释成在观察到的数据集类别中，特定的 *x_i* 属于某个类别的概率，或者称作似然（Likelihood）。

提示：softmax 用以解决概率计算中概率结果大而占绝对优势的问题。例如函数计算结果中有两个值 a 和 b，且 a>b，如果简单地以值的大小为单位进行衡量，那么在后续的使用过程中，a 永远被选用而 b 由于数值较小而不会被选择，但是有时候也需要使用数值小的 b，softmax 就可以解决这个问题。

softmax 按照概率选择 a 和 b，由于 a 的概率值大于 b，因此在计算时 a 经常会被取得，而 b 由于概率较小，因此取得的可能性也较小，但是有概率被取得。

softmax 函数的代码如下：

```
import numpy
def softmax(inMatrix):
m,n = numpy.shape(inMatrix)
outMatrix = numpy.mat(numpy.zeros((m,n)))
soft_sum = 0
for idx in range(0,n):
    outMatrix[0,idx] = math.exp(inMatrix[0,idx])
    soft_sum += outMatrix[0,idx]
```

```
    for idx in range(0,n):
        outMatrix[0,idx] = outMatrix[0,idx] / soft sum
    return outMatrix
```

可以看到，当传入一个数列后，分别计算求出每个数值所对应的指数函数值，之后将其相加，再计算每个数值在数值和中的概率。例如：

```
a = numpy.array([[1,2,1,2,1,1,3]])
```

结果请读者自行打印验证。

2.2　安装 PyTorch 2.0

Python 运行环境调试完毕后，本节的重点就是安装本书的主角——PyTorch 2.0。

2.2.1　NVIDIA 10/20/30/40 系列显卡选择的 GPU 版本

目前市场上有 NVIDIA 10/20/30/40 系列显卡，对于需要调用专用编译器的 PyTorch 来说，不同的显卡需要安装不同的依赖计算包。笔者在此总结了不同显卡的 PyTorch 版本以及 CUDA 和 cuDNN 的对应关系，如表 2-1 所示。

表 2-1　NVIDIA 10/20/30/40 系列显卡的版本对比

显卡型号	PyTorchGPU 版本	CUDA 版本	cuDNN 版本
10 系列及以前	PyTorch 2.0 以前版本	11.1	7.65
20/30/40 系列	PyTorch 2.0 向下兼容	11.6+	8.1+

注意：这里的区别主要在于显卡运算库 CUDA 与 cuDNN 的区别，当在 20/30/40 系列显卡上使用 PyTorch 时，可以安装 CUDA11.6 版本以上以及 cuDNN8.1 版本以上的计算包。

注意：由于本书使用 PyTorch 2.0，因此要求读者的计算机配有 20/30/40 系列的显卡，用来顺利运行本书的示例代码。

下面以 PyTorch 2.0 为例，演示完整的 CUDA 和 cuDNN 的安装步骤，不同的版本的安装过程基本一致。

2.2.2　PyTorch 2.0 GPU NVIDIA 运行库的安装

本小节讲解 PyTorch 2.0 GPU 版本的前置软件的安装。对于 GPU 版本的 PyTorch 来说，由于调用了 NVIDIA 显卡作为其代码运行的主要工具，因此额外需要 NVIDIA 提供的运行库作为运行基础。

我们选择 PyTorch 2.0.1 版本进行讲解。对于 PyTorch 2.0 的安装来说，最好的方法是根据官方提供的安装命令进行安装，具体参考官方文档 https://pytorch.org/get-started/previous-versions/。从页面上可以看到，针对 Windows 版本的 PyTorch 2.0.1，官方提供了几种安装模式，分别对应 CUDA 11.7、CUDA 11.8 和 CPU only。使用 conda 安装的命令如下：

```
    # CUDA 11.7
    conda install pytorch==2.0.1 torchvision==0.15.2 torchaudio==2.0.2
pytorch-cuda=11.7 -c pytorch -c nvidia
    # CUDA 11.8
    conda install pytorch==2.0.1 torchvision==0.15.2 torchaudio==2.0.2
pytorch-cuda=11.8 -c pytorch -c nvidia
    # CPU Only
    conda install pytorch==2.0.1 torchvision==0.15.2 torchaudio==2.0.2 cpuonly -c
pytorch
```

下面以 CUDA 11.8+cuDNN 8.9 为例讲解安装的方法。

（1）首先是 CUDA 的安装。在百度搜索 CUDA 11.8 download，进入官方下载页面，选择适合的操作系统安装方式（推荐使用 exe(local)本地化安装方式），如图 2-14 所示。

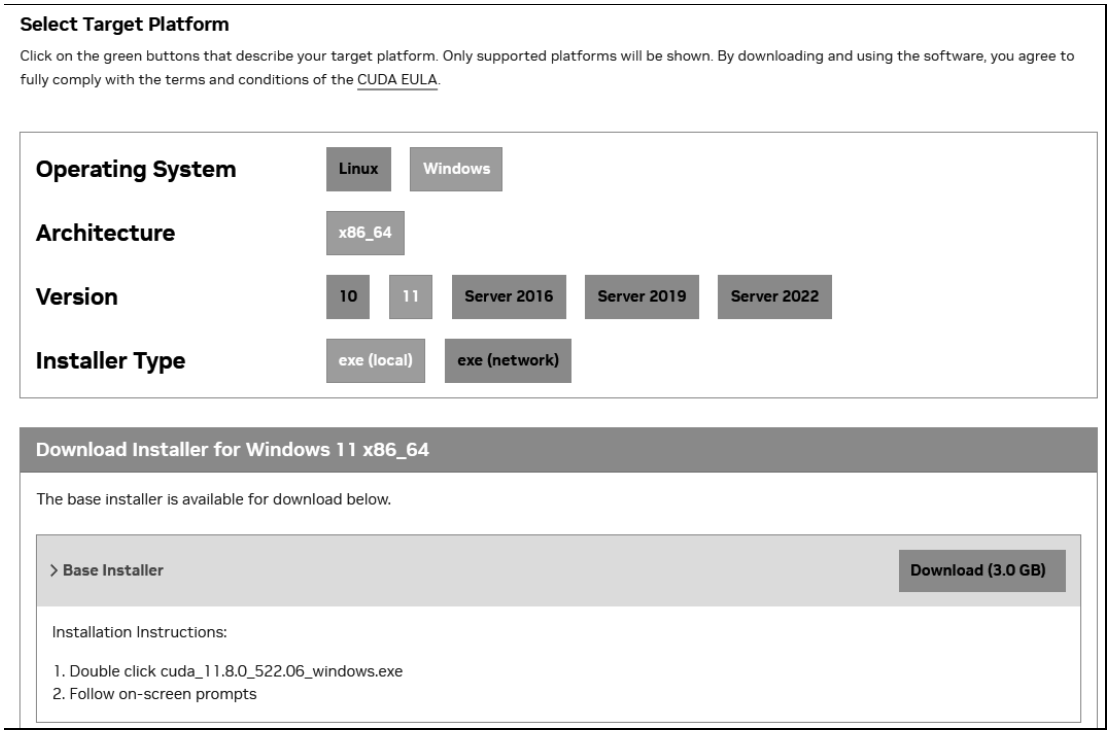

图 2-14　CUDA 11.8 下载页面

此时下载下来的是一个 EXE 文件，读者自行安装，不要修改其中的路径信息，完全使用默认路径安装即可。

（2）下载和安装对应的 cuDNN 文件。要下载 cuDNN，需要先注册，相信读者可以很快完成，之后直接进入下载页面，如图 2-15 所示。

注意：不要选择错误的版本，一定要找到对应 CUDA 的版本号。另外，如果使用的是 Windows 64 位的操作系统，需要下载 x86_64 版本的 cuDNN。

图 2-15　cuDNN 8.9 下载页面

（3）下载的 cuDNN 是一个压缩文件，将它解压并把所有的目录复制到 CUDA 安装主目录中（直接覆盖原来的目录），CUDA 安装主目录如图 2-16 所示。

图 2-16　CUDA 安装主目录

（4）确认 PATH 环境变量，这里需要将 CUDA 的运行路径加载到环境变量的 PATH 路径中。安装 CUDA 时，安装向导能自动加入这个环境变量值，确认一下即可，如图 2-17 所示。

（5）最后完成 PyTorch 2.0.1 GPU 版本的安装，只需在终端窗口中执行本小节开始给出的 PyTorch 安装命令即可。

```
# CUDA 11.8
conda install pytorch==2.0.1 torchvision==0.15.2 torchaudio==2.0.2
```

```
pytorch-cuda=11.8 -c pytorch -c nvidia
```

图 2-17　将 CUDA 路径加载到环境变量 PATH 中

2.2.3　Hello PyTorch

到这里我们已经完成了 PyTorch 2.0 的安装。下面使用 PyTorch 2.0 做一个小练习。打开 CMD，依次输入如下命令，验证安装是否成功。

```
import torch
result = torch.tensor(1) + torch.tensor(2.0)
result
```

结果如图 2-18 所示。

```
Python 3.11.4 | packaged by Anaconda, Inc. | (main, Jul  5 2023, 13:47:18)
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> result = torch.tensor(1) + torch.tensor(2.)
>>> result
```

图 2-18　验证安装是否成功

或者打开前面安装的 PyCharm IDE，先新建一个项目，再新建一个 hello_pytorch.py 文件，输入如下代码：

```
import torch
```

```
result = torch.tensor(1) + torch.tensor(2.0)
print(result)
```

最终结果请读者自行验证。

2.3　Hello ChatGLM3

ChatGLM 系列是智谱 AI 发布的一系列大语言模型，因为其优秀的性能和良好的开源协议，在国产大模型和全球大模型领域都有很高的知名度。其开源的第三代基座大语言模型 ChatGLM3-6B 的性能较前一代有了大幅提升，可以认为是十分强大的中文基础大模型。

2.3.1　ChatGLM3 简介与安装

自诞生以来，ChatGLM 便以其卓越的性能和广泛的应用而闻名于世，成为国产大语言模型中最强大、最著名的模型。2023 年 3 月，第一代 ChatGLM-6B 面世，开源之后便获得了广泛的关注和使用。短短 3 个月后，ChatGLM2 的发布再次引发了科技界的热议。2023 年 10 月 27 日，智谱 AI 再次发布了第三代基础大语言模型 ChatGLM3 系列，这一系列共包含了 3 个模型：基础大语言模型 ChatGLM3-6B-Base、对话调优大语言模型 ChatGLM3-6B 和长文本对话大语言模型 ChatGLM3-6B-32K。

1. ChatGLM3 简介

ChatGLM3 系列的发布，标志着中国在自然语言处理领域的研究取得了新的突破。这一系列模型不仅具备了更强大的语言理解能力，更能在不同场景下进行高效、精准的应用。其中，ChatGLM3-6B-Base 作为基础大语言模型，凭借其出色的性能表现，成为众多领域中的得力助手；ChatGLM3-6B 通过对话调优，实现了更加智能、自然的交互体验；ChatGLM3-6B-32K 作为长文本对话大语言模型，以其在长文本处理方面的卓越表现，广泛应用于新闻媒体、科技文献等领域。ChatGLM3 模型说明如表 2-2 所示。

表2-2　ChatGLM3模型说明

版本模型	模型介绍	上下文长度	备注
ChatGLM3-6B-Base	基础大语言模型，预训练结果	8K	基础模型
ChatGLM3-6B	基础大语言模型，针对对话微调调优，适合对话	8K	
ChatGLM3-6B-32K	对话调优的大语言模型，但是支持 32K 上下文	32K	不支持工具函数调用

ChatGLM 系列的持续升级和优化，不仅提升了国产大语言模型的国际竞争力，更为人工智能领域的发展注入了新的活力。未来，我们有理由相信，ChatGLM 将继续秉持开放、共享的精神，推动人工智能技术的创新和发展，为人类社会的进步贡献更多的力量。

需要注意，ChatGLM3 的功能不局限于生成对话，它在工具调优、Prompt（提示）调优、代码执行等方面都有很大提升。具体来说，ChatGLM3 的功能提升主要集中在以下几点：

- 更强大的基础模型：ChatGLM3-6B 的基础模型 ChatGLM3-6B-Base 采用了更多样的训练数据、更充分的训练步数和更合理的训练策略。在语义、数学、推理、代码、知

识等不同角度的数据集上测评显示，ChatGLM3-6B-Base 在 10B 以下的预训练模型中具有最强的性能。

- 更完整的功能支持：ChatGLM3-6B 采用了全新设计的 Prompt 格式，除正常的多轮对话外，同时原生支持工具调用（Function Call）、代码执行（Code Interpreter）和 Agent 任务等复杂场景。
- 更全面的开源序列：除了对话模型 ChatGLM3-6B 外，还开源了基础模型 ChatGLM-6B-Base、长文本对话模型 ChatGLM3-6B-32K。以上所有模型的源码权重对学术研究完全开放，在填写问卷进行登记后还可免费商业使用。

2. ChatGLM3 安装

在介绍完 ChatGLM3 激动人心的新功能之后，下面开始使用 ChatGLM3，读者有两种方式获取 ChatGLM3。

第一种方式是从 GitHub 上获取 ChatGLM3 源码（笔者不推荐）。读者可以登录 GitHub，搜索关键字“ChatGLM”，打开对应的链接，结果如图 2-19 所示。

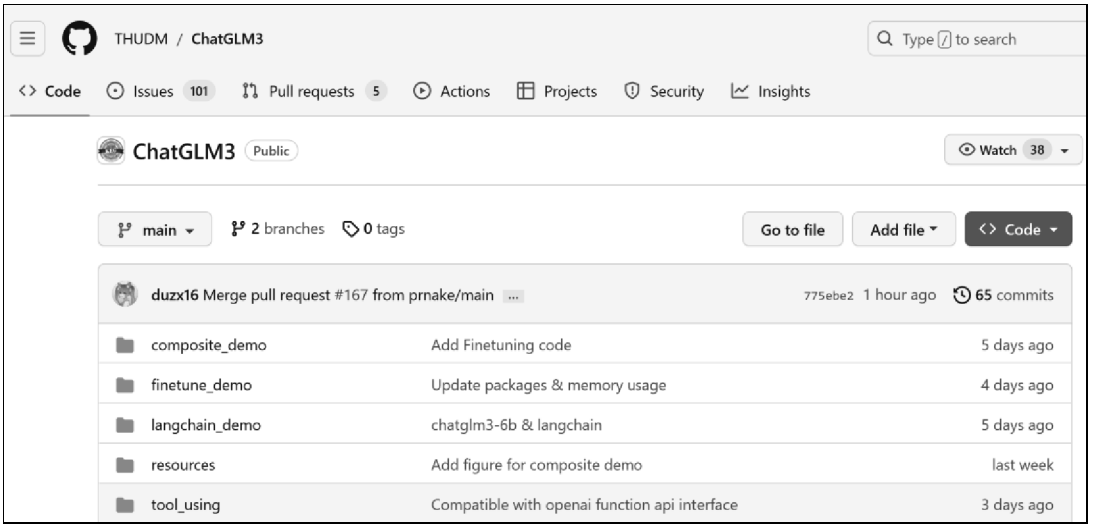

图 2-19　GitHub 上的 ChatGLM3 源码

这里读者可以使用 GitHub 自行下载对应的源文件，但需注意，这里的源码权重并不直接提供，而是需要单独下载。因此，读者可以采用第二种方式（笔者推荐）来完成 ChatGLM3 的配置和使用。

第二种方式是通过魔塔社区（ModelScope）（见图 2-20）的模型库，安装对应的库包来直接下载和使用完整的 ChatGLM3。

魔塔社区的库包可以使用 pip 命令进行安装，如下所示。

```
pip install modelscope
```

待安装结束后，即可在 PyCharm 中新建一个空的项目，通过在代码中调用 ChatGLM3-6B 模型来生成对话。

图 2-20　魔塔社区首页

为了便于读者选择适合自己的 ChatGLM3 版本，下面我们将依次使用 CPU 版本、GPU（INT4 或 INT8 量化）版本、GPU（half 或 float 量化）版本对本示例进行推演。

2.3.2　CPU 版本的 ChatGLM3 推演

我们首先完成 CPU 版本的 ChatGLM3 的推演。由于采用 CPU 进行推演，因此在自动下载完模型参数后，推演的耗时较长，这一点请读者注意。

```
    import torch
    from modelscope import AutoTokenizer, AutoModel, snapshot_download
    model_dir = snapshot_download("ZhipuAI/chatglm3-6b", revision = "v1.0.0")
    tokenizer = AutoTokenizer.from_pretrained(model_dir, trust_remote_code=True)

    with torch.no_grad():
       model = AutoModel.from_pretrained(model_dir,
trust_remote_code=True) .cpu().float()

    model = model.eval()
    response, history = model.chat(tokenizer, "你好", history=[])
    print(response)
    response, history = model.chat(tokenizer, "晚上睡不着应该怎么办", history=history)
    print(response)
```

由于第一次调用，需要从网上直接下载对应的权重文件，这个权重文件比较大，因此读者需要根据自身的网络带宽情况等待文件下载结束。

下载过程如图 2-21 所示，可以看到此时的下载较烦琐，权重文件被分成了 8 个部分依次下载。

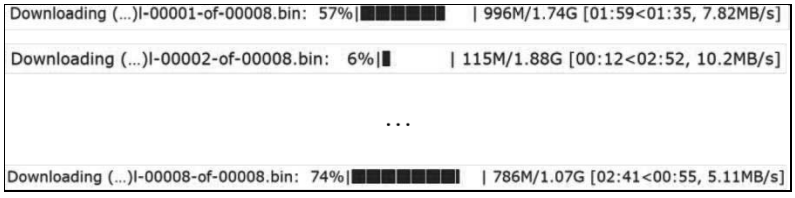

图 2-21　下载过程展示

此时需要注意，对于下载的内容，还需要进行一次合并处理的过程，如图 2-22 所示。

图 2-22　合并分割的权重

如果有报错，可以等待 1 分钟后重新连接，再进行下载。

全部下载完成后，ChatGLM3 自动进入运行模式，在代码中可以设置需要让模型回答的普通问题，上述代码段的回答结果如图 2-23 所示。注意：返回回答结果的时间会长一点，请读者耐心等待。

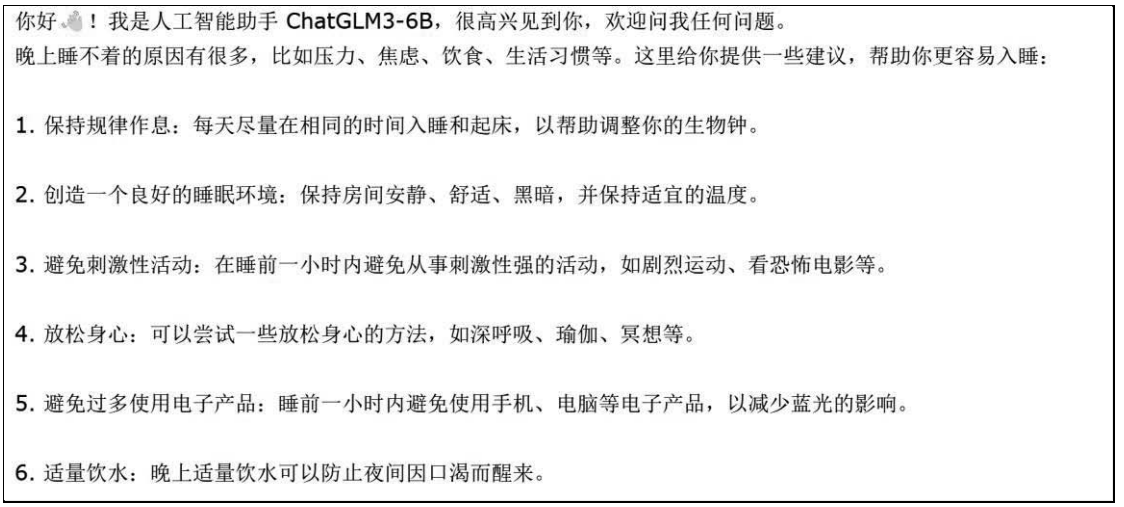

图 2-23　ChatGLM 的回答示例

请读者自行运行代码进行验证。需要读者注意的是，即使问题是一样的，但是每一次运行代码得到的回答也有可能不一样。因为我们所使用的 ChatGLM 是生成式模型，前面的生成会直接影响后面的生成，而这点也是生成模型相对于一般模型不同的地方。前面的结果有了波动，后面就会发生很大的变化，会有一个滚雪球效应。

另外，CPU 版本的 ChatGLM3 推演耗时较长，因此推荐读者尽量采用 GPU 版本的模型进行后续的学习。

2.3.3　GPU（INT4 或 INT8 量化）版本的 ChatGLM3 推演

在 PyTorch 中，模型量化是一种优化技术，它可以将模型的权重和激活值从浮点数转换为低精度的整数格式，如 INT4 和 INT8。这种转换可以显著减少模型的大小和计算复杂度，同时加快推理速度，特别是在资源受限的设备上，如智能手机、嵌入式系统和物联网设备等。现在，我们来详细了解一下 INT4 和 INT8 这两种 GPU 模型量化推演方式。

1. INT8 量化

INT8 量化是将模型的权重和激活值从浮点数转换为 8 位整数的过程。与浮点数相比，虽然 INT8 整数表示的数值范围较小，精度较低，但它可以显著减少存储和计算的需求。

在 INT8 量化中，模型的权重和激活值会经过一个量化过程，该过程包括缩放和偏移，以确保量化后的值能够尽可能地保留原始浮点数的信息。在推理时，这些量化值会被反量化回浮点数进行计算，然后再量化回 INT8 进行下一步操作。

2. INT4 量化

INT4 量化是一种更为激进的量化方式，它将模型的权重和激活值量化为 4 位整数。由于表示范围更小，精度更低，INT4 量化通常会导致更大的精度损失。然而，与 INT8 相比，INT4 量化可以进一步减少模型的存储需求和计算复杂度。

需要注意，INT4 量化在实际应用中相对较少见，因为过低的精度可能导致模型性能显著下降。此外，不是所有的硬件都支持 INT4 操作，因此在选择量化方式时需要考虑硬件的兼容性。

3. 量化的优势

量化具有以下优势：

- 减少模型大小：量化可以显著减少模型的存储需求，使得部署更加轻便。
- 提高推理速度：低精度的计算通常比浮点数计算更快。
- 降低能耗：在移动和嵌入式设备上，低能耗是至关重要的，量化有助于减少计算时的能耗。

4. 注意事项

量化的注意事项如下：

- 精度损失：量化通常会导致一定程度的精度损失，需要在性能和精度之间进行权衡。
- 模型需重新训练：有些量化方法可能需要重新训练或微调模型以恢复性能。
- 硬件支持：不同的硬件对量化的支持程度不同，需要确保所选的量化方式与目标硬件兼容。

下面来完成 INT4 量化的 ChatGLM3 推演代码，如下所示（注意 model 的构建方式）：

```
    import torch
    from modelscope import AutoTokenizer, AutoModel, snapshot_download
    model_dir = snapshot_download("ZhipuAI/chatglm3-6b", revision = "v1.0.0")
    tokenizer = AutoTokenizer.from_pretrained(model_dir, trust_remote_code=True)

    with torch.no_grad():
        # 采用量化方案的 ChatGLM3
        model = AutoModel.from_pretrained(model_dir,
trust_remote_code=True).quantize(4).cuda()

    model = model.eval()
    response, history = model.chat(tokenizer, "你好", history=[])
    print(response)
    response, history = model.chat(tokenizer, "晚上睡不着应该怎么办", history=history)
    print(response)
```

请读者自行打印结果进行验证和比较。

如果想换成 INT8 格式的模型构建方法，则可以采用如下的 model 构建：

```
   model = AutoModel.from_pretrained(model_dir,
trust_remote_code=True).quantize(8).cuda()
```

可以看到，只需简单地将 quantize(4)中的数字 4 改成 8 即可。具体请读者自行验证。

2.3.4 GPU（half 或 float 量化）版本的 ChatGLM3 推演

使用 GPU 对 ChatGLM3 进行推演，除了前面提到量化方法之外，常规方案的 GPU 推演中 half 和 float 是两种最常用的格式。它们分别对应 16 位浮点数（float16）和 32 位浮点数（float32）。half 格式占用 13GB 显存，float 格式占用 40GB 显存。下面来具体介绍。

1. half 格式

half 格式即 16 位浮点数（float16）。与 float 相比，half 格式占用的内存减半，这在大规模深度学习应用中具有显著优势。它允许我们在相同的 GPU 内存限制下加载更大规模的模型或处理更多数据。此外，随着现代 GPU 硬件对 float16 操作的支持不断增强，使用 half 格式还可能带来计算速度的提升。然而，half 格式有一个固有的缺点，即降低精度，这可能导致在某些情况下出现数值不稳定或精度损失的情况。

2. float 格式

接下来谈谈 float 格式。这种格式提供了较高的精度，能够准确表示大范围的数值。在进行复杂的数学运算或需要高精度结果的场景中，float 格式是首选。然而，高精度也意味着更多的内存占用和可能更长的计算时间。对于大规模深度学习模型，尤其是当模型参数众多、数据量巨大时，float 格式可能会导致 GPU 内存不足或推演速度下降的情况。

选择是使用 float 还是 half 格式时，需要考虑以下几个因素：

- 模型要求：某些模型对精度非常敏感，可能需要使用 float32。
- 硬件支持：不是所有 GPU 都支持 float16 操作，或者支持的程度不同。
- 内存限制：如果 GPU 内存有限，使用 float16 可以显著减少内存占用。
- 性能要求：对于需要快速推演的应用，float16 可能是一个更好的选择。

选择 float 还是 half 格式是一个权衡精度、内存和性能的过程。如果应用场景对精度要求较高，且 GPU 内存充足，那么 float 格式可能是更好的选择；如果面临内存限制或希望加快推演速度，那么使用 half 格式或混合精度训练可能会带来显著的好处。对于 ChatGLM3 这样的大型模型，混合精度推演可能是一个折中的好选择，它能够在保持一定精度的同时，充分利用 GPU 的计算能力。

下面是使用 half 格式完成 ChatGLM3 模型推演的示例，代码如下：

```
    import torch
    from modelscope import AutoTokenizer, AutoModel, snapshot_download
    model_dir = snapshot_download("ZhipuAI/chatglm3-6b", revision = "v1.0.0")
    tokenizer = AutoTokenizer.from_pretrained(model_dir, trust_remote_code=True)

    with torch.no_grad():
        # 采用 GPU half 方案的 ChatGLM3
        model = AutoModel.from_pretrained(model_dir,
trust_remote_code=True).half().cuda()

    model = model.eval()
```

```
    response, history = model.chat(tokenizer, "你好", history=[])
    print(response)
    response, history = model.chat(tokenizer, "晚上睡不着应该怎么办", history=history)
    print(response)
```

如果想要使用 float 格式下的 GPU 推演模式，则只需要将 model 的构建代码改为如下方式：

```
   model = AutoModel.from_pretrained(model_dir,
trust_remote_code=True).float().cuda()
```

此时，对于普通用户来说应该会产生一个爆显存的提示。因此在具体选择上，读者应该遵循自己所使用的硬件资源合理选择不同的模型。如果 GPU 具有较大的显存容量，那么使用 half 版本的模型可能不是问题；对于显存有限的情况，选择量化版本将是一个明智的决策。

为了保证 ChatGLM3 模型在后续学习过程中的效率和准确率之间的平衡，笔者将主要使用 GPU 的 half 版本来进行讲解。half 版本通过降低数据精度来减少显存占用，同时在一定程度上保留了模型的准确性。这种折中方案使得更多用户能够在有限的硬件资源上运行和实验 ChatGLM3 模型。

在具体学习过程中，读者还应该根据自己的具体条件和需求来调整模型的构建方式，包括修改模型的层数、神经元数量、激活函数等。通过合理调整这些参数，读者可以获得一个既高效又准确的推断结果。

总之，读者在使用 ChatGLM3 模型时，应该充分了解自己的硬件资源限制，并根据实际情况选择合适的模型版本和构建方式。这样做不仅可以避免显存溢出等问题，还可以确保模型在推演过程中保持高效和准确。

2.3.5　离线状态的 ChatGLM3 的使用

对于部分有特殊需求的读者来说，一个能够离线使用或者部署在私有云架构上的大模型是必不可少的。ChatGLM3 也提供了可完全离线的 ChatGLM 部署和使用方法。首先，需要打开魔塔社区在本地的下载地址，一般这个存储地址为：

```
    C:\Users\你当前登录用户名\.cache\modelscope\hub\ZhipuAI
```

此时，在这个文件夹下，我们可以看到从魔塔社区下载的全部 ChatGLM3 的源码与权重文件，局部界面如图 2-24 所示。

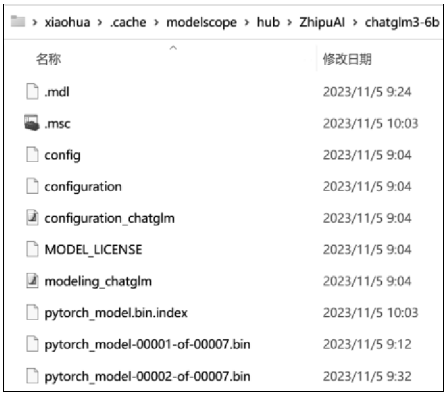

图 2-24　ChatGLM3 源码与权重文件

要离线使用 ChatGLM3，读者可以将 chatglm3-6b 目录直接复制到 PyChram 的当前项目目录下，如图 2-25 所示。然后在代码中修改模型存档地址。

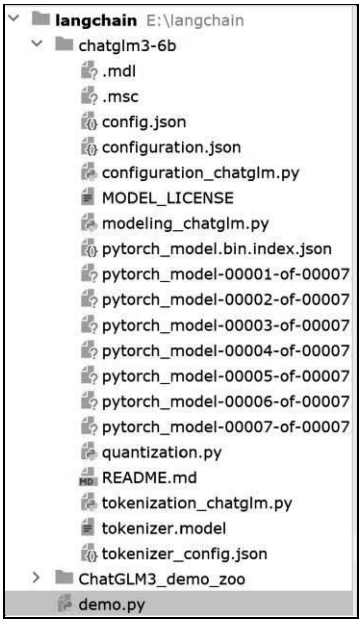

图 2-25　把 ChatGLM3 源码放到当前项目目录下

对于新的 ChatGLM3 模型的调用，也可以简单地使用如下代码完成：

```
    # model_dir = snapshot_download("./chatglm3-6b", revision = "v1.0.0") # 注销联网验证代码
    model_dir = "../chatglm3-6b"          # 直接提供 ChatGLM3 的存储地址
    …
    tokenizer = AutoTokenizer.from_pretrained(model_dir, trust_remote_code=True)
    model = AutoModel.from_pretrained(model_dir, trust_remote_code=True).half().cuda()
```

在上面代码中，我们注释了原始的 snapshot_download 下载和验证函数，替代为直接使用本地模型，相对于当前 demo 位置的源码和权重存储相对地址。

最终结果请读者自行打印完成。

2.3.6　ChatGLM 的高级使用

前面我们进行了一次非常小的演示，介绍了 ChatGLM 的使用。除此之外，ChatGLM 还可以有更多用法，例如进行文本内容的抽取，读者可以尝试如下任务：

```
    content="""ChatGLM3-6B 是一个最强开源的、支持中英双语的大语言模型，
    基于 General Language Model(GLM)架构，具有 62 亿参数。
    手机号 18888888888
    结合模型量化技术，用户可以在消费级的显卡上进行本地部署（INT4 量化级别下最低只需 6GB 显存）。
    ChatGLM-6B 使用了较 ChatGPT 更为高级的技术，针对中文问答和对话进行了优化。
    邮箱 123456789@qq.com
```

```
    经过约 1T 标识符的中英双语训练，辅以监督微调、前馈自助、人类前馈强化学习等技术的加持，
    账号:root 密码:xiaohua123
    62 亿参数的 ChatGLM-6B 已经能生成相当符合人类偏好的回答，更多信息请参考我们的博客。
    """
    Prompt='从上文中，提取"信息"(keyword,content)，包括:"手机号"、"邮箱"、"账号"、"密码"
类型的实体，只输出这些实体，不要其他的。输出 JSON 格式内容'
    input ='{}\n\n{}'.format(content,Prompt)
    print(input)
    response, history = model.chat(tokenizer, input, history=[])
    print(response)
```

这是一个经典的文本抽取任务，希望通过 ChatGLM 抽取其中的内容。这里使用了一个 Prompt（中文暂时称为“提示”），它是研究者们为了下游任务设计出来的一种输入形式或模板，能够帮助 ChatGLM“回忆”起自己在预训练时“学习”到的东西。

Prompt 也可以帮助使用者更好地“提示”预训练模型所需要去做的任务，这里通过 Prompt 的方式向 ChatGLM 传达了一个下游任务目标，即需要对文本进行信息抽取，抽取其中蕴含的手机号、邮箱、账号密码等常用信息。最终显式结果如图 2-26 所示。

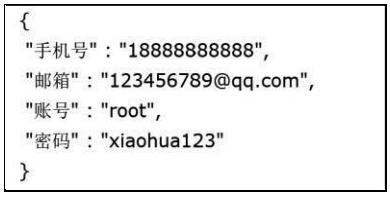

图 2-26　文本信息抽取结果

可以看到，这是一个使用 JSON 表示的抽取结果，其中的内容还根据 Prompt 中定义的键值，直接抽取了对应的内容。

除此之外，我们还可以使用 ChatGLM 进行一些常识性的文本问答和编写一定量的代码。当然，要完成这些内容，还需要设定好特定的 Prompt，从而使得 ChatGLM 能更好地理解用户所提出的问题和所要表达的意思。

2.4　本章小结

在本章中，首先详细介绍了 PyTorch 2.0 框架、开发环境和第三方软件的安装：先讲解了 PyTorch 的独特性和其在机器学习领域的重要地位，然后逐步展开，详细阐述了如何进行环境配置和软件安装，并通过一个非常简单的示例验证了安装环境。

在接下来的代码演示部分，我们以 ChatGLM3 为例，向读者展示了大语言模型的基本使用方法。ChatGLM3 是一种基于 Transformer 的大语言模型，具有进行自然语言对话和生成连贯文本的能力。我们详细介绍了如何使用 ChatGLM3 进行对话和文本生成，从而为后续的大模型应用提供了坚实的基础。

本章为后续的深度学习应用提供了必要的基础。希望读者能够充分理解和掌握本章的内容，为后续的学习和实践打下坚实的基础。