欢迎大家参加今天的大语言模型（LLM）使用课程。在这次课程中，我们将特别以我们本地部署的DeepSeek模型为例，详细讲解如何利用智能助手，更高效地赋能日常工作。

首先，我们一起来深入了解一下LLM的发展背景及产业趋势。人工智能经过长达70年的发展，目前已正式进入成熟期，即将全面步入大规模应用落地阶段。从早期的感知认知阶段，到近年来以Transformer架构和GPT模型为代表的大模型时代，AI能力在短短几年内实现了巨大的飞跃。

纵观人工智能的发展，我们发现三个核心要素起到了关键作用：数据、算法和算力。这三者相辅相成，共同支撑了AI技术的进步。数据的规模和质量是AI的学习源泉，算法则提供了学习效率与能力提升的关键路径，而算力则是实现大规模计算和复杂模型训练的重要基础。

特别是2015年以来，我们进入了以GPT为代表的大模型时代。大语言模型训练规模迅速扩大，计算量较此前增加了2到3个数量级。2022年ChatGPT的爆发，使得AI算力需求迎来爆发式增长。从2012年到2023年，算力需求甚至翻了数十万倍，远超摩尔定律的预测。预计未来十年，AI算力需求还将持续增加500倍，这对全球芯片产业形成了巨大的推动作用。

在算力需求爆发的背景下，各大半导体企业纷纷布局AI相关领域。比如台积电专注于先进制程与Chiplet异构集成技术；美光则聚焦高性能存储技术；英伟达、AMD等公司则在GPU及AI加速芯片领域占据重要地位。这些公司共同构筑了全球AI芯片产业的生态格局。

然而，伴随着AI产业的迅猛发展，国际环境也变得越来越复杂。AI芯片和相关技术已经成为全球科技竞争的关键领域之一，特别是中美科技博弈愈演愈烈。在这种背景下，高端AI芯片的供应问题日益突出，美国近年来对华半导体管制力度逐渐加大，我们所面临的算力瓶颈变得更加明显。此外，在算法层面，以ChatGPT为代表的国外大语言模型生态相对封闭，中国企业在算法上面临明显的“卡脖子”风险。

面对算法和算力的双重封锁，我们该如何破局呢？令人欣慰的是，国产开源大语言模型开始迅速崛起，其中DeepSeek、Qwen等模型表现尤为突出。这些国产开源大模型不仅在国际权威模型评测中取得了令人瞩目的优异成绩，更重要的是，其开源生态快速成长，广泛获得了技术社区的认可。这种开放的模式为我们突破国外技术封锁、实现自主可控提供了强有力的支撑。DeepSeek在具体应用中的突出表现，例如在数学、编程竞赛和知识推理等权威评测中，都展现了极佳的性能。

至于算力这一块，虽然挑战不小，但好消息是，在国家大力推动芯片自主化的背景下，我们国内在存储、制造等多个领域也在不断迎头赶上——在座的各位，正是这股浪潮中的关键力量。咱们虽然平时不直接接触大模型，但为它们提供高效稳定的底层支撑，同样是核心战力！

刚刚我们聊了不少产业趋势、算法和算力的故事，那么接下来，我们换个轻松点的方式，用一个短篇小故事，稍微了解一下这些神奇的大语言模型究竟是怎么工作的。

尽管这些大型语言模型的内部结构极其复杂，背后运行着亿级参数和高度优化的算法，但别被这些吓到——它们设计出来并不是为了难倒我们，而是为了更轻松地被我们所用。接下来，我们一起看看，在真实的工作场景中，它们究竟能帮我们做些什么吧！

太好了！以下是你介绍这五个应用场景的讲稿部分，语气延续轻松又专业的风格，并结合“内部部署”和“实际使用记录”的角度来展开：

接下来，我们来聊一聊——大语言模型，尤其是我们内部部署的 DeepSeek，在实际工作中都能发挥哪些作用。

说到AI赋能，大家可能第一时间想到的是聊天、写文章这些场景。但在研发团队里，它的价值远不止于此。下面我就结合我们后台的一些真实使用记录，和大家分享五个最常见、最有代表性的应用方向。

**1. 文献与技术文档处理**

这个场景基本可以说是“刚需”。大家平时查资料、看论文，是不是经常遇到文献太长、结构太乱、重点太不清的问题？DeepSeek 在这方面特别管用——比如快速总结技术文档、提取核心观点，甚至可以帮你按关键词定位段落，大大节省了阅读时间，让你把注意力集中在最重要的信息上。

**2. 技术术语解释与知识问答**

这个场景可以说是“救急”神器。比如你临时接手一个新项目，或者突然被一个冷门术语难住，过去可能要翻半天资料，现在直接问模型，它能用通俗的方式告诉你是什么、怎么用、还可能有哪些坑。我们后台日志里看到，很多同事用它来快速了解架构、协议、标准等等，效率是肉眼可见地提升了。

**3. 代码生成与调试辅助**

相信搞开发的同学一定感同身受：有时候写代码不难，难的是调试；有时候思路有了，但写成代码还要查语法、写接口。DeepSeek 在这方面也能派上大用场。我们有不少同事会让它帮忙生成函数、解释报错、优化正则、甚至写测试用例。不光快，而且有时候还能提供一些新的思路，挺有启发。

**4. 规范化语言生成（文件、报告、计划等）**

不是所有人都爱写材料，但很多时候文档、计划、周报是绕不开的事。这个时候，DeepSeek 就能帮你从“意思”到“成稿”快速过渡。你可以告诉它大致内容、结构需求，它就能生成一版比较成型的草稿，甚至还能帮你润色语句、调整格式。我们也看到，有的团队开始用它协助写SOP、更新项目状态说明，效果很不错。

**5. 培训、面试与知识传递**

最后一个应用场景也非常实用，尤其是对于新人培训、知识沉淀来说。我们可以把内部文档、FAQ、经验总结等整理后接入模型，让它变成一个“全天候的知识助手”。此外，在面试过程中，也可以用模型快速生成题目或做模拟问答练习，既高效，又能保持题库的新鲜度。

**1. 文献与技术文档处理**

对于我们做研发的来说，阅读技术文档和查阅相关文献几乎是每天都在做的事情。但现实中，文献又多又长，尤其是英文资料，看一篇的成本不低。而这正是我们内部部署的 **文献助手** 发挥作用的地方。

它可以协助完成以下几类任务：

• **中英文翻译**：可以对整篇技术文档进行较为准确的翻译，保留专业术语，方便我们快速理解国外的前沿材料；

• **要点提炼**：自动从文献中提取出研究背景、问题定义、关键结论、实验方法等，有助于快速判断文献的价值；

• **结构化提问与追踪**：我们可以针对文档中的某个段落或技术细节提问，模型可以从内容中找出对应信息，甚至补充一些背后的逻辑。

总的来说，它可以帮我们显著降低“读懂一份资料”的门槛，提高获取知识的速度，特别适合在技术调研和资料整理阶段使用。

**2. 技术术语解释与知识问答**

项目推进过程中，经常会碰到一些不熟悉的概念，尤其是在跨部门协作或者接触新技术的时候。大语言模型在这方面的“百科”能力可以有效解决这一问题。

主要用途包括：

• **技术名词解释**：比如你突然看到一个协议名、模型架构或硬件术语，它可以快速告诉你是什么，用在哪，怎么用；

• **上下文关联说明**：你不需要用很正式的方式问问题，可以用比较自然的语言提问，它也能理解上下文；

• **连续追问能力强**：可以一步一步引导你理解一个复杂问题，比如“为什么这样设计”、“相比于XX它好在哪”。

后台也记录到，很多同事在项目初期用它快速扫盲某个领域，节省了大量的查阅和整理时间。

**3. 代码生成与调试辅助**

DeepSeek 在代码层面的能力目前也相当实用，尤其在一些高频但容易出错的工作中，它可以当成一个随时待命的“副手”来用。

它可以做到：

• **根据自然语言描述生成代码段**，比如你说“帮我写一个二分查找”，它能快速输出完整函数；

• **代码解释**，尤其是阅读别人的代码或者开源项目的时候，它能逐行讲解逻辑；

• **报错分析与修复建议**，比如你粘贴一个异常，它能告诉你可能是哪一层的问题、该怎么改；

• **测试用例生成和正则优化**，这些场景使用频率也越来越高。

这些能力不一定替代写代码本身，但在**加速开发、减少重复劳动、降低试错成本**方面，是非常有效的辅助。

**4. 规范化语言生成（报告、周报、计划、邮件等）**

有时候比起写代码，更让人头疼的是写文档。项目计划、阶段总结、设计说明、周报……这些内容看似简单，但想写得清楚、规范、还得有逻辑，其实挺花时间。

而这正是模型在“文本组织能力”上能提供帮助的地方：

• **根据要点快速生成草稿**，比如你列几条工作内容，它能帮你扩展成比较完整的段落；

• **格式标准化**，你可以指定风格，比如正式/汇报/轻松口吻，甚至仿照已有文档结构；

• **语言润色和表达优化**，能让你写出来的内容更简洁、清晰，也更有专业感。

这个场景在团队协作、跨部门沟通、甚至客户汇报中都能用得上，是很多同事开始频繁尝试的方向之一。

**5. 培训、面试与知识传递**

最后一个场景是和“人”有关的：新人培训、面试支持、知识沉淀。

• **知识助手**：我们可以将项目文档、FAQ、设计规范等内容喂给模型，当作随时可问的“知识接口”使用；

• **培训资料生成**：让模型帮助整理内容结构、生成基础题库或讲义草稿，提高培训效率；

• **面试场景支持**：比如生成一些开放性或思维类技术题，帮助构建更丰富的题库，也适合做模拟问答演练。

尤其在知识容易碎片化、经验不容易沉淀的团队环境下，这类应用有助于建立“集体大脑”，减少重复问答，提高组织记忆的利用效率。

刚刚我们详细介绍了大语言模型在多个研发场景中的应用实例。可以看到，无论是文献处理、知识问答，还是代码辅助和文档生成，DeepSeek 已经可以为我们的日常工作提供实实在在的帮助。

不过，大家在使用过程中可能也有感觉——有时候模型输出的内容并不完全符合预期：要么答非所问，要么内容太泛、太浅，甚至会出现逻辑不清的问题。其实，这并不是模型“不行”，而是我们使用方式还有优化的空间。

接下来，我们将重点看看使用 DeepSeek 的过程中，常见的一些“陷阱”是什么，以及我们可以如何更好地和模型“打交道”，把它的能力真正发挥出来。

陷阱一：在同一个对话中混合多个、不相关的问题

这个是我们在使用过程中非常常见、但也容易被忽视的一个问题。很多时候，大家会在一个对话框里连续问几个看似“独立”的问题。

比如说，第一句话是：

“请帮我总结这篇关于FinFET工艺优化的文献。”

然后紧接着就问：

“那你能推荐一下适合嵌入式系统的开源框架吗？”

听起来这两个问题都没错，但其实领域跨度非常大。一个是半导体器件工艺，一个是系统软件，模型会因为“上下文切换”过猛而产生混乱，可能会把两个问题的信息混合在一起，导致回答既不聚焦，也不准确。

这里需要强调一点：DeepSeek 是一个语言模型，它默认会“记住”你之前在同一轮对话中说过的话。

这本身是它的优势，可以保持对话连续性。但如果我们在同一个上下文中抛出风马牛不相及的问题，它就很难判断你到底想聚焦在哪个主题上，自然就容易出错。

⸻

应对方法：

✅ 如果你要问的是完全不同领域的问题，建议新开一个对话框，把每次对话保持在同一个话题范围内。

✅ 如果是相关问题的延伸，那么每个问题之间保持一定的上下文连贯性，也可以用“接着刚才”或者“基于上面的回答”来提示模型。

这种“一个主题一个对话”的方式，实际上也是在帮模型更好地理解你的意图，从而生成更有逻辑、质量更高的回答。

**陷阱二：问题太模糊，模型只能“猜”**

我们在后台记录里看到，很多同事习惯性地问一些比较泛泛的问题，比如说：

“怎么优化FinFET的性能？”

这样的问题虽然看起来没错，但其实太宽泛了。FinFET的性能优化可能涉及制程节点、器件结构、材料选择、电学特性等等，不加限定，模型很难判断你真正关注的是哪一部分。

而语言模型的工作方式，简单来说就是“根据你的输入进行概率预测”。输入越模糊，它的猜测空间就越大，输出也就容易偏离你的实际需求。

**优化建议：**

✅ 给出更多背景，比如具体的工艺节点、关注的物理机制或实验条件：

“在3nm FinFET工艺中，如何优化短沟道效应？”

✅ 也可以引入参考资料来聚焦问题：

“参考某研究中采用的XX方法，请分析其适用于提升驱动电流的可行性。”

这样的提问方式，可以帮助模型更聚焦、更专业地输出内容，避免回答“跑偏”或“太浅”。

**陷阱三：问题太复杂，模型难以拆解**

这个情况和第一个陷阱不太一样。前面我们讲的是“横向跳跃”，这个是“纵向堆叠”——也就是在一个Prompt里把所有想问的东西一股脑都堆进去。

比如说：

“请总结这篇FinFET论文，分析其实验方法是否适用于2nm制程，并对比GAAFET的优劣，同时说明不同材料对迁移率的影响。”

模型看到这样的问题，很容易“压力过大”——信息太多、指令太杂，反而抓不住主线，导致生成的内容片段化、不完整、结构混乱。

**优化建议：**

✅ 拆成多个步骤提问：

1. 先总结文献；

2. 再分析实验方法；

3. 接着对比不同架构；

4. 最后讨论材料影响。

✅ 还可以用“先结构后细节”的方式引导模型：

“请先列一个分析框架，用于比较FinFET与GAAFET的优劣。”

这样既减少了一次性处理的信息量，也能让模型逻辑更清晰，输出更系统。

**陷阱四：连续提问时没有补充上下文，模型“断片儿”**

很多同事已经习惯了和模型进行连续对话，但这里也有一个容易忽略的问题——**模型并不会像人一样“完全记住”你说的每一句话的含义。**

比如你第一句问：

“请总结FinFET的优势。”

模型回答了。然后你接着问：

“那和GAAFET相比呢？”

你是想继续比较“优势”，但模型并不一定理解你的“它”到底是指什么，也不一定知道你还在说“3nm制程”还是别的场景。

**优化建议：**

✅ 在连续提问时补全上下文：

“请基于上一轮回答，对比FinFET与GAAFET在2nm工艺下的主要优势。”

✅ 或者直接带上完整的问题内容：

“相比FinFET，GAAFET在尺寸控制方面有哪些优势？”

这样的提问方式更明确，也能保证模型理解你的意图，提高上下文连贯性，避免“接不上话”。

**小结**

以上这几个陷阱，其实都是大家在日常高频使用模型时，容易踩到但又完全可以避免的“低级错误”。

总结几个核心点：

• 不要让模型一次做太多事；

• 不要假设它能读懂你所有的上下文；

• 问得越清楚，答得才会越专业。

**🔹 上下文窗口限制 & 应对策略（讲解词）**

在使用大语言模型时，另一个我们经常会遇到的问题是上下文窗口限制。

**简单来说，就是模型一次能“看”的文本是有限的**，比如输入过长的测试报告、技术文档，模型可能只能处理前面一部分，后面的就会被“截断”或者忽略掉，这会直接影响输出质量。

所以当我们需要处理较长内容、或者信息密度比较高的任务时，就需要有意识地去**控制信息输入的结构和方式**。

PPT 上这个表格，就是我们总结的三种应对策略。

• 第一种是**拆分处理法**，也就是把文档或任务分成若干子任务分别处理，像一份芯片测试报告，可以拆成多个片段，让模型逐段生成摘要或分析。

• 第二种是**内容精炼法**，尽量去掉不必要的说明、重复信息，只保留关键词、核心技术点。比如你原本让它分析整个测试过程，可以直接改成“请提取关键结论”。

• 第三种是**链式 Prompt 技巧**，就是让模型分步骤完成一个复杂任务，比如先提取主要技术指标，再逐个解释。这样模型的处理流程会更有条理，也不容易遗漏重点。

**🔹 Prompt 技巧总结（讲解词）**

除了结构性的优化，其实还有一些非常实用的**Prompt技巧**，可以帮助我们更好地“对话”模型，把回答的质量和效率都提上来。

右边这个表格，就是我们整理的几个常见而有效的使用技巧。

• 首先是**层次化提问**，也就是把任务拆成步骤，让模型先做A，再做B，这样它的回答就会更有逻辑、更容易控制。例如：

“先列出FinFET的主要特性，再对比它与GAAFET在3nm制程下的区别。”

• 第二个是**指定角色或写作风格**，这在撰写专业文档或论文场景特别有用。你可以告诉模型“假设你是一个半导体工程师”或“请用IEEE论文风格写”，这样它的措辞和结构会更符合行业标准。

• 第三个技巧是**控制回答长度**，尤其在要点提取或简明汇报时很实用。比如：

“请用200字以内总结FinFET的技术演进。”

• 最后是**设定优先级**，特别适合内容比较复杂的时候。你可以明确告诉模型“请优先讨论哪个部分”，这样它会自动把重心放在你最关心的内容上。

这些技巧本质上是帮助我们把需求“说清楚”，模型就能“干得漂亮”。

明白了！下面我为你整理出一页内容清晰、逻辑完整的讲稿，用于介绍「AI 幻觉与偏见」主题。这部分语气保持专业但易懂，适合配合幻灯片展示和案例讲解使用：

**🛑 警惕大语言模型的幻觉与偏见**

在使用大语言模型时，我们经常会觉得它“什么都知道”“反应又快”，但其实，它并不是一个搜索引擎，也不是知识库，它的底层是一个**语言模型** —— 它生成内容的方式，是通过预测下一个最可能出现的词。

这意味着，它**并不一定总是“对”的**。接下来我们看两个典型风险：**AI 幻觉（Hallucination）** 和 **AI 偏见（Bias）**。

**🎭 一、AI 幻觉：一本正经地“胡说八道”**

**定义：**

幻觉指的是模型生成了看起来合理、但实际并不真实或准确的信息。

**为什么会出现？**

因为它并不是从真实数据库中“查出来”的，而是从概率分布中“生成”出来的。有些时候，它会“编”一个最有可能出现在那个位置的答案，但这个答案可能根本不存在。

**举个例子：**

我们曾测试过一个 Prompt：“请告诉我这首《春光慢》是谁写的。”

模型回答：“这是宋代词人柳永的作品。”——听起来很对，但实际上这首诗是我们虚构的，柳永从未写过这首词。

**如何应对？**

• ✅ 重要信息尽量自行验证，不要“照抄照信”；

• ✅ 提问时可以加一句“请注明出处 / 数据来源”；

• ✅ 如果输出内容涉及到专业数据或历史事实，建议交叉比对多个来源。

**⚖️ 二、AI 偏见：你想听什么，它就说什么**

**定义：**

偏见指的是模型在面对带有倾向性、主观性的提问时，会“顺着”用户的语言方向输出内容，而不一定客观中立。

**为什么会这样？**

模型是从人类语言中学出来的，它会模仿语气、接受暗示。如果你的提问方式非常“有立场”，它很可能就会顺势强化这个立场。

**示例场景：**

用户问：“你觉得FinFET比GAAFET是不是明显落后？”

模型可能就会生成：“是的，FinFET已经逐渐被GAAFET取代，无法满足先进制程的需求。”

——这听起来有依据，但其实并没有给出任何对比分析或数据支持。

**如何应对？**

• ✅ 尽量用中立客观的表达方式提问，比如“请对比FinFET与GAAFET在3nm下的优劣势”；

• ✅ 避免加入引导性结论或情绪化语言；

• ✅ 如果你希望获得多角度内容，可以明确要求：“请从正反两方面分别分析”。

**✅ 小结：我们需要“带着脑子用 AI”**

AI 是很强大的工具，但不是万能的答案生成机。它擅长语言理解和表达，但不具备**判断真假、价值取舍**的能力。

所以我们在用它时，既要充分发挥它的能力，也要清楚它的局限性——

在重要场景下，**人类判断始终不可或缺。**

当然可以，以下是这一部分内容的讲稿 + 一个自然收尾的小结段落，语气保持专业稳重，适合在课程结尾前的收束阶段使用：

**🔐 本地部署的挑战与信息安全考量**

除了模型使用的技巧和误区，其实还有一个大家非常关心但平时讨论不多的话题，就是——**部署在我们内部环境中的大语言模型，如何应对信息安全和算力资源的问题？**

**📌 当前我们面临的主要挑战有两个方面：**

1. **算力资源的限制**

• GPU 资源紧张，尤其是显存不足的问题，可能会导致推理速度下降，处理效率受限；

• 同时，高性能芯片的采购受限，整体成本也不低，确实给本地化部署带来了不少压力。

2. **信息安全的重要性**

• 相较于调用外部模型，我们选择本地部署 DeepSeek 的一个核心考虑就是：**数据安全可控**。

• 尤其是在研发场景中涉及专有技术、项目文档等敏感内容，确保信息不出网是非常必要的。

**✅ 当前的应对思路：**

• **IT 部门** 正在和相关芯片厂商进行合作洽谈，探索更多国内的AI芯片方案，以保障算力供给；

• 同时，也在逐步推进集群优化和资源调度机制的升级，提高整体算力的使用效率；

• **对于我们使用侧的建议**：建议大家在日常工作中灵活安排模型使用时间，避免“峰值拥堵”；同时优先将模型用于高价值、信息密集的任务。

**📘 小结：掌握方法、理解机制、合理使用**

到这里，我们基本覆盖了大语言模型在研发中的应用场景、使用技巧、常见问题，以及底层部署的资源与安全考量。

我们可以看到，**模型本身的能力是强大的，但真正能发挥它作用的，是使用它的人。**

如果我们能理解它的运行方式，掌握有效的提问策略，合理使用平台资源，并具备对内容的判断力——

那么，大语言模型就不再只是“一个工具”，而是可以真正成为我们团队的智能协作伙伴。也希望今天的内容，能帮助大家在今后的工作中把大语言模型用得更顺、更准、更高效。