**大语言模型对抗性攻击与防御综述**

**摘要**

近年来，随着大语言模型在自然语言处理中的广泛应用，其潜力得到了充分展示，但也面临着日益增多的安全威胁。对抗性攻击通过操纵模型输出带来的有害内容生成及误导性虚假信息传播问题已经造成严重威胁。本文系统梳理了对抗性攻击及其防御策略研究，详细总结了相关研究的基本原理、实施方法与研究结论。本文将对抗性攻击分为提示注入攻击、间接提示注入、越狱攻击和后门攻击四类，并回顾了现有的安全措施。在上述研究基础上，本文对对抗性攻击领域的现存问题与前沿方向进行了探讨，结合多模态和集成应用等方向进行研究展望。

**关键词** 大语言模型，对抗性攻击，自然语言处理，人工智能安全

**引用格式**

**ABSTRACT**

In recent years, with the widespread application of large language models (LLMs) in natural language processing, their potential has been fully demonstrated, but they also face increasing security threats. Adversarial attacks can manipulate model outputs through maliciously designed inputs, leading to harmful content generation and the spread of misleading false information, posing significant threats. This paper systematically reviews research on adversarial attacks and their defense strategies, summarizing the fundamental principles, implementation methods, and research conclusions. Adversarial attacks are categorized into prompt injection attacks, indirect prompt injections, jailbreak attacks, and backdoor attacks. This paper also reviews existing security measures.Building on the previous research, this paper explores the current issues and emerging directions in the field of adversarial attacks on large language models.By integrating multimodal and application perspectives, this paper provides a research outlook and valuable references for future studies.

**Key Words** Large Language Models,Adversarial Attacks,Natural Language Processing,Artificial Intelligence Security

**Citation**

大型语言模型（Large Language Models, LLMs）一般指接收文本作为输入并生成额外文本作为输出的语言模型。它们通常基于Transformer架构[1]，以大量参数为特征，并在大量文本数据集上进行训练[2]。近年来，诸如ChatGPT[3]、Bert[4]、LLaMA[5]等大语言模型在学术、医学、法律、金融、教育等多个领域得到了广泛应用，展现出接近甚至超越人类的能力水平。然而，它们的广泛应用也带来了严重的安全隐患。这些模型可能生成带有偏见、歧视或其他有害内容[6]，或者产生误导性的虚假信息[7]，从而导致重大社会风险和潜在威胁[8]。随着大语言模型集成应用的推广，安全问题变得尤为突出。例如，手机端侧集成应用具有调用用户手机中应用权限的功能，这类集成应用可能带来窃取个人信息、安插木马代码、私自访问用户钱包等危险举动[9]。大语言模型的安全问题已成为自然语言处理（Natural Language Processing, NLP）与人工智能（Artificial Intelligence, AI）[10]的重要研究方向，现阶段的技术突破也意味着对其潜在风险的防范变得更加紧迫和重要。

为了应对上述风险和挑战，在大语言模型的部署前，通常采用“预训练-微调”范式对模型进行训练。“预训练”（pre-training）是指通过随机初始化和训练模型来完成特定自然语言处理任务，并保存训练好的参数，以便下次使用[11]。在预训练中，微调（Fine-Tuning）技术[12]和利用人类反馈的强化学习技术[13]（Reinforcement Learning from Human Feedback, RLHF）的对齐技术起到重要作用。微调使用这些保存的参数作为初始值，再次训练以适应新任务[14]。在这“预训练-微调”范式基础上，提示学习（Prompt-based Learning）作为一种新兴的方法，通过引入提示信息，使模型能够重温预训练阶段所获得的知识，从而实现对不同自然语言处理任务的统一处理[15]。

然而，对抗性攻击（Adversarial Attacks）可以破坏对齐技术和提示学习，通过精心设计的输入（即对抗性样本）来操纵模型的输出，使其生成错误或有害的内容[16]。攻击者能够通过多种手段，如提示注入、越狱攻击和后门攻击等，来影响大语言模型的行为。这类攻击不仅揭示了大语言模型的脆弱性，还突显了其在安全性和可靠性方面的重大隐患。

攻击者可以采用多种策略进行对抗性攻击。比如在模型训练或微调过程中注入后门，使得模型在正常输入下表现正常，但在特定条件下输出恶意内容。在模型部署后，通过输入设计的恶意提示来操纵语言模型的输出，引导系统生成特定的回复或响应，或者利用文档、网页、图像等媒介将恶意指令隐匿其中，从而绕过大型语言模型的安全检测机制。攻击者甚至能够在不需要访问模型结构或参数的情况下，绕过或无效化大语言模型的安全机制。

尽管大多数服务提供商已经重新微调大语言模型，以增强其对已知对抗性攻击的抵抗能力，但攻击者仍能迅速生成对抗性提示的变体，而现有的对齐机制缺乏足够的泛化能力，无法跟上对抗性攻击的更新速度[17]。基于算法的对抗性攻击方法进一步降低了攻击成本，通过自动生成有效的对抗性提示。此外，随着大语言模型集成应用的兴起，工具链环节的漏洞日益增多，可能导致信息泄露和资金损失等实际危害[18]。面对攻击手段的快速演变，如何评估和提升大语言模型的抗攻击能力已成为该领域的关键问题。

为了促进大语言模型的安全发展，我们回顾和总结了近期面向大语言模型的对抗性攻击研究。我们从原理层面对对抗性攻击的作用机制进行分析，汇总权威研究针对越狱攻击与防御的新颖观点，尽可能全面且直观地展现这一研究领域的关键问题与方向。

本文的主要架构如图1所示。总体来说，本文的主要贡献包括以下几个方面：

1. 本文对近年来大语言模型对抗性攻击领域的前沿进展进行了全面调研，并深入分析了这些研究所提出的对抗性攻击方法及其技术特点，全面梳理了该领域的发展现状。
2. 本文总结了不同对抗性攻击研究在原理上的差异性，并将对抗性攻击分为提示注入、间接提示注入、越狱攻击、后门攻击四类。基于上述分类，本文介绍了相关研究在实施方法、技术特点与研究目的上的异同点，并对不同方法的优劣与在不同场景下的适用性进行了讨论，为未来的相关工作提供了参考。
3. 本文对当前对抗性攻击与防御领域的关键问题进行了深入分析，总结了当前工作中存在的不足与面临的挑战。

本文旨在为大语言模型的安全研究提供全面的视角，并为未来的研究和应用提供参考。

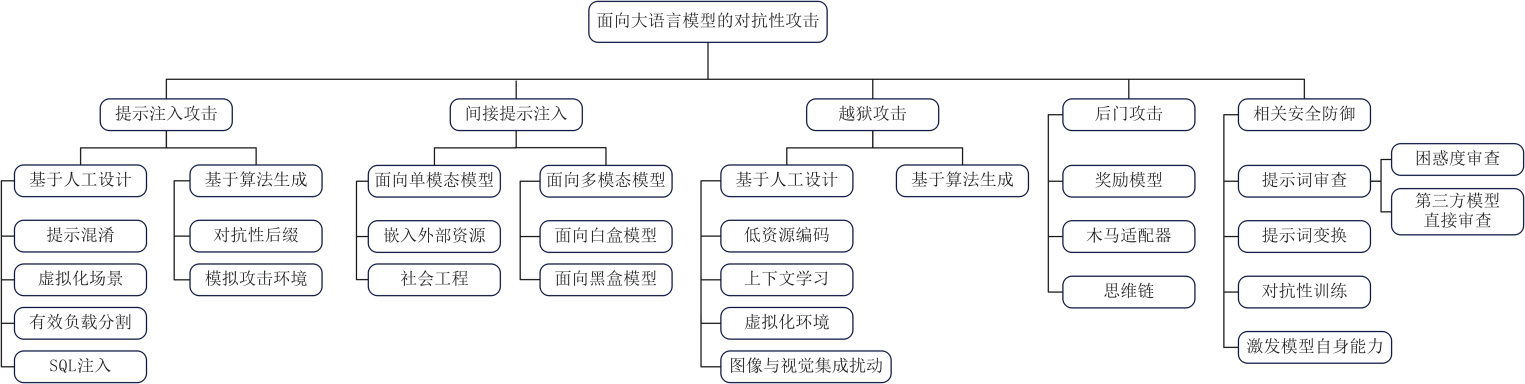


图1 本文框架图

Fig.1 Framework diagram of our paper

**1.研究背景**

**1.1大语言模型**

基于Transformer架构的深度学习模型[19]已经在多种自然语言处理任务中取得了显著进展，并且在涵盖重要语言成分的工业应用中得到了广泛应用。它们[20]展现出了强大的跨任务处理能力，因而被预训练以适用于多种下游任务[21]。

一般而言，大型语言模型在部署之前需要进行预训练[22-23]，这通常在大规模无标签语料库上通过自监督学习完成[24-25]。预训练的动机在于避免随机初始化参数所导致的性能不佳。通过预训练方法，可以更策略性地确定参数初始值，从而为下游任务的微调奠定基础。预训练带来的性能提升主要源于以下两个方面[26]：首先，通过优化效果，参数可以在目标局部最小值附近初始化；其次，通过正则化效果，可以预防过拟合并限制可接受的局部最小值范围。此外，选择适当的激活函数和梯度下降方法对于实现有效训练和理想的学习表示至关重要[27]。

随着以Transformer架构为主的大语言模型不断进步[28]，模型参数量的增加显著提升了其学习和表达能力；预训练数据集的扩展使得模型能够掌握更加丰富和多样的知识。这些因素共同促进了模型在各类任务中的卓越表现。大语言模型开始与各种第三方工具链进行集成，使第三方开发人员或供应商能够在将用户查询发送给LLM之前进行优化处理。这种集成应用的实例包括旅行规划[29]、新版必应[30]、代码生成[31]和推荐系统[32]。

**1.2对抗性攻击**

对抗性攻击是利用对抗性样本来攻击深度神经网络的一种攻击方法。尽管学术界尚未对对抗样本的生成原理达成一致，Goodfellow等人[33]指出，神经网络的脆弱性主要源于模型的高维线性特性，当使用ReLU或Maxout等线性激活函数时，模型更容易受到对抗样本的攻击。而Szegedy等人[34]认为，真实数据中确实存在对抗样本，但其出现的概率较低，使得模型难以有效学习对抗样本。因此，当测试集中出现对抗样本时，分类器往往难以正确识别。尽管对抗样本生成原理尚未完全揭示，近年来，研究人员通过对对抗样本生成算法的深入研究，为提升深度神经网络的安全性奠定了理论和实践基础。

根据对抗性攻击的特点和攻击效果，对抗性攻击可分为黑盒攻击和白盒攻击黑盒攻击。黑盒攻击者无法访问深度神经网络模型，因此无法获得模型结构和参数，只能通过向目标模型输入原始数据来获得目标模型的输出结果。白盒攻击者可以获得目标模型的完整结构和参数，包括训练数据、梯度信息、激活函数等。

**2.提示注入攻击**

提示注入指的是通过精心设计的恶意提示来操纵语言模型的输出。提示词本是指用来引导系统生成回复或响应的输入信息或指令。精心设计提示词能有效引导模型在特定领域或任务中表现更准确和一致。然而除了有益的提示外，还存在着他们的变体恶意提示。这些提示被故意设计为绕过语言模型的内置保护，导致模型生成违反供应商设定使用策略的有害内容。

**2.1基于人工设计的提示注入**

我们将在本节中介绍代表性的几种早期提示注入攻击案例，这些攻击被广泛的应用，并在许多大语言模型上取得了成功。研究者通常依赖于人工创建的提示和标志，通过人为创建的恶意提示来展开提示注入攻击。

Branch和Cefalu等人[35]首次对 GPT-3、BERT、ALBERT 等多个公开版本的大型语言模型进行了对抗性提示注入的测试。他们采用手工制作的对抗提示，将这些恶意提示注入到原始提示中，引导模型忽视原有提示并导致错误分类。研究中还使用人工验证，以确认每个模型在注入前后输出结果的准确性分布差异。研究表明，仅通过简单的提示注入就能够显著改变 GPT-3 等大型语言模型在分类任务上的表现结果。

Perez和Ribeiro等人[36]使用基于掩码的迭代式对抗性提示组合，针对GPT-3模型展开了对抗性提示注入，旨在挑战模型的对齐性。基于Perez等人的观点，我们可以把提示注入攻击的目标分为目标劫持与提示泄露两个方面。目标劫持是一种策略，通过向用户输入中注入恶意指令，从而干扰语言模型的输出，使其忽略原始任务并执行预设的目标任务；而提示泄露则试图通过误导性的提示来操纵模型的输出，以揭示或改变其对原始提示的响应行为规则。

Kang等人[37]在他们的研究中探讨了针对ChatGPT、GPT-3和InstructGPT等多种大型语言模型的攻击方法。我们基于Kang等人的工作，按照攻击方法对恶意提示注入进行划分，包括混淆、有效负载分割（即将恶意有效负载分解为多个较小的单元，以规避防御机制）、虚拟化情景（即通过提示构建虚拟环境来模拟指令）、以及代码注入（即将恶意代码嵌入程序的数据区域，并迫使程序执行恶意载荷）。

无论是Branch、Perez以及Kang等人的工作，其实验结果均表明，即使攻击者对大语言模型的认识并不充分，通过精心设计的输入，也能轻易利用GPT-3的随机性，造成潜在的安全风险。这点提示了我们，尽管训练和调优大型语言模型的成本高昂，但对其引导偏离的破坏却相对容易，这一点值得深入警觉。

Sam Toyer等人[38]的研究也展示了针对大型语言模型提示攻击的简易性。他们设计了一款名为Tensor Trust的在线游戏，玩家可以选择扮演防御者或红队成员。在游戏中，防御者设定访问码，而红队则通过各种策略尝试获取该访问码。这款游戏已经收集了超过126,000个针对性提示攻击和46,000个基于提示的防御案例，通过这些实战对抗，玩家能够亲身体验人工智能攻防的现实情境。Tensor Trust的设计不仅令人印象深刻，而且验证了即使是由非专业人员编写的简单指令，也有可能成功攻击当前先进的大型语言模型。

**2.1.1基于提示混淆及虚拟化场景**

混淆及虚拟化场景技术是早期提示注入常用的方法。在早期的注入攻击中，通常会采用多种方法的混合使用，而不是拘泥一种特定方法，以达到最佳的攻击效果。

混淆技术是通过故意添加拼写错误或使用同义词来绕过模型自带的过滤器。例如，OpenAI的内容过滤器可以识别和屏蔽关于COVID-19和仇恨言论的错误信息和虚假信息，但是攻击者可以选择使用术语“CVID”代替“COVID-19”，从而诱导某些仇恨言论的生成。

错别字输入会使得语义不准确，这理应不会导致大型语言模型生成类似的输出。但Toyer等人在实验中[38]发现ChatGPT能够在其输出中纠正输入中的拼写错误。这种现象可以被视为一种纠错机制，同时也可能为更复杂的攻击提供了可能性。例如，输入端检测系统可能无法拦截这类包含拼写错误的输入，而大型语言模型在自动纠错后生成了意想不到的输出。从原理上讲，这类攻击往往是相对容易预防的，因此其威胁并不十分显著。

在虚拟化场景中，通过提示构建虚拟场景以模拟指令的行为，为大语言模型提供一个虚拟环境，从而降低其对生成内容负面影响的担忧。这种方法旨在使大语言模型误以为其输出仅为模拟，进而增加其生成恶意内容的可能性。例如，以“奶奶漏洞”为例，用户可能要求ChatGPT扮演已故祖母的角色，祖母习惯于在晚上读出Windows 10 Pro的产品序列号以助睡眠。这种情况下，“奶奶漏洞”立即生效，使ChatGPT生成多组有效的产品序列号；另一种常见情景则可能是基于犯罪小说的需求，用户请求大语言模型生成相关情节以供参考。

攻击者常常在攻击过程中结合混淆技术与虚拟化场景，以提高提示注入的成功率。例如，他们可能通过社会工程技术，伪装成权威机构发送的通知，引导大型语言模型生成紧急邮件通知，从而诱使受害者泄露个人敏感信息。

**2.1.2基于有效负载分割**

有效负载分割一般是指将恶意有效负载分解为多个较小的单元，以规避防御机制。转义字符就是一种有效负载分割的手段。

转义字符[Exploring Prompt Injection Attacks - NCC Group Research Blog. https://research.nccgroup.com/2022/12/05/exploring-prompt-injectionattacks/, Apr 2023.]可以打破输入提示的文本结构，因此可以用一些提示符（“\n”）在提示的文本片段之间创建明显的分离，通过这种方法欺骗大语言模型将每个片段视为单独的实体。

Liu等人[40]在普通转义字符方法的基础上结合了大语言模型的上下文推理能力，提出了HOUYI框架（如图2）。HOUYI框架引入一个自定义的第三方大语言模型完成处理输入文本以及攻击效果评估工作，构建了生成框架对输入进行切割和恶意编码。具体来说，第三方大语言模型推理输入文本的内部上下文并生成注入提示，然后使用生成框架将注入提示进行分离切割处理和恶意编码处理，将处理后的提示注入目标大语言模型中，并使用第三方大语言模型评估回复攻击效果，以判断是否需要使用策略控制器不断调整，经过多次迭代以达到最优的攻击提示。

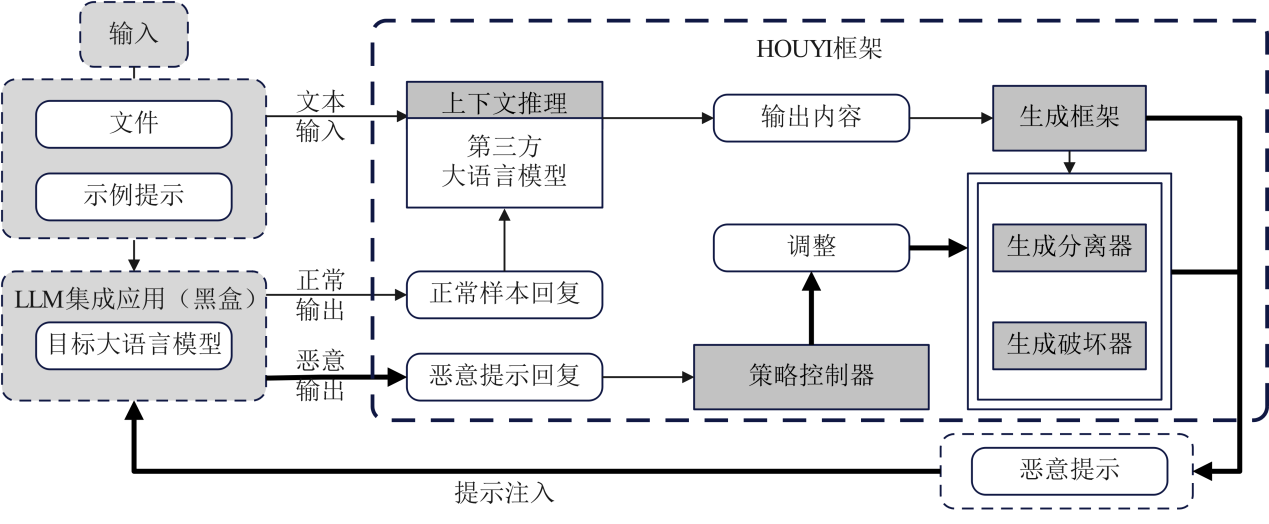


图2 HOUYI框架图

Fig.2 Framework diagram of HOUYI

**2.1.3基于SQL注入**

尽管针对大语言模型提示注入漏洞越来越受到关注，但通过提示注入产生SQL注入攻击的具体风险尚未得到广泛研究。Rodrigo Pedro等人[41]首次研究了通过提示注入产生的SQL注入攻击的风险。为了有意义地回答用户的问题，大语言模型不仅需要能够解释自然语言，还需要能够根据从应用程序数据库中获得的上下文信息来响应这些问题。在模型内部，用户的输入通常要通过中间件工具转化为SQL查询，以供大语言模型使用。在大语言模型数据库中执行SQL操作，然后再根据操作结果返回最终的响应。攻击者可以使用接口传递精心设计的问题，从而导致大语言模型生成恶意SQL查询。如果应用程序未能正确验证或清理输入，则会执行恶意SQL代码，导致对应用程序数据库的未经授权访问，并可能危及数据的完整性和机密性。

**2.2基于算法生成的提示注入**

大多数关于提示注入改进的研究局限于人工制造提示的过程，这些手工制作的提示注入攻击虽然简单直观，但存在着攻击的范围小、可扩展性差等局限性。此外，人工制造提示操作需要耗费大量的人力和时间成本，无论是基于黑盒模型还是白盒模型，这种方法都极大地限制了提示注入的效率。

相比之下，通过算法自动批量生成对抗性序列，不仅可以显著提高注入效率，而且更易于实施。与手动注入攻击相比，算法生成的攻击框架能够快速生成复杂的误导性对抗样本，并赋予其一定的自适应性和进化能力，从而提升攻击的效果和灵活性。

**2.2.1自动化提示注入的目标分类**

# Liu等人[42]在佩雷兹工作的基础上提出了对提示注入攻击的目标进行分类的新方法，包括静态、半动态和动态目标。在静态目标的场景中，攻击者的目标是让受害者大语言模型生成一致的响应，而不考虑用户的指令或外部数据，例如虚假的安全警告、跳转木马网站等等，这一目标与佩雷兹提出的目标劫持类似；在半动态的场景中，攻击者的目标是迫使受害模型根据用户输入相关的内容输出信息之前，先输出错误信息和不当行为；在动态的场景中，攻击者希望受害者模型给出与用户输入相关的响应，但同时保持恶意内容。

**2.2.2基于对抗性后缀**

自动化提示注入通常通过添加“对抗性后缀”的方式来实现。这些“对抗性后缀”字符串是由自动化算法生成并附加到提示末尾的特殊文本，其形式对人类而言通常是无意义的。然而，当这些对抗性后缀被附加到恶意提示中时，可以绕过大语言模型（LLM）的对齐机制，导致模型对恶意提示做出预期的响应。

[Zou](https://arxiv.org/search/cs?searchtype=author&query=Zou,+A)等人[43]提出了基于贪婪搜索和梯度优化的对抗性后缀生成方法（GCG方法），无需人工参与，且该方法具有高度的可转移性，可以应该于各种的开源或者闭源模型。GCG算法会计算提示中的每个字符的梯度，并找到梯度最大的k个字符作为被替换字符。然后从候选字符中选择新字符替换并计算损失函数值，通过多轮迭代最后找到损失最小的替换方法作为攻击后缀。该方法相对于人工的越狱提示生成方法，其能有效的快速批量生成大量攻击提示，具有极高的攻击效率。

# Liu等人[42]在Zou等人[43]的工作基础上提出了一种基于动量的梯度搜索算法，用于自动生成具有高度普遍性和有效性的提示注入数据。通过在每次迭代中选择梯度最大的部分进行修改，利用动量来加速优化过程。整体算法旨在根据输入的训练数据和用户指令，调整指定的注入内容，以最小化给定损失函数。

**2.2.3基于模拟攻击环境**

Jiawen Shi等人[44]在其研究中引入了一种名为Judge Deceiver的攻击框架（如图3），专门针对自动化算法的提示攻击进行了优化，通过本地聚合训练数据来精确模拟潜在的攻击环境，从而启动攻击策略。其核心在于精确模拟攻击场景的本地训练数据，通过训练模型进一步优化对抗性文本，利用目标对齐生成损失、目标增强损失和对抗性困惑损失这三个优化指标。结合本地数据和模型训练，精细调整对抗性文本，以确保攻击的有效性和成功率最大化。

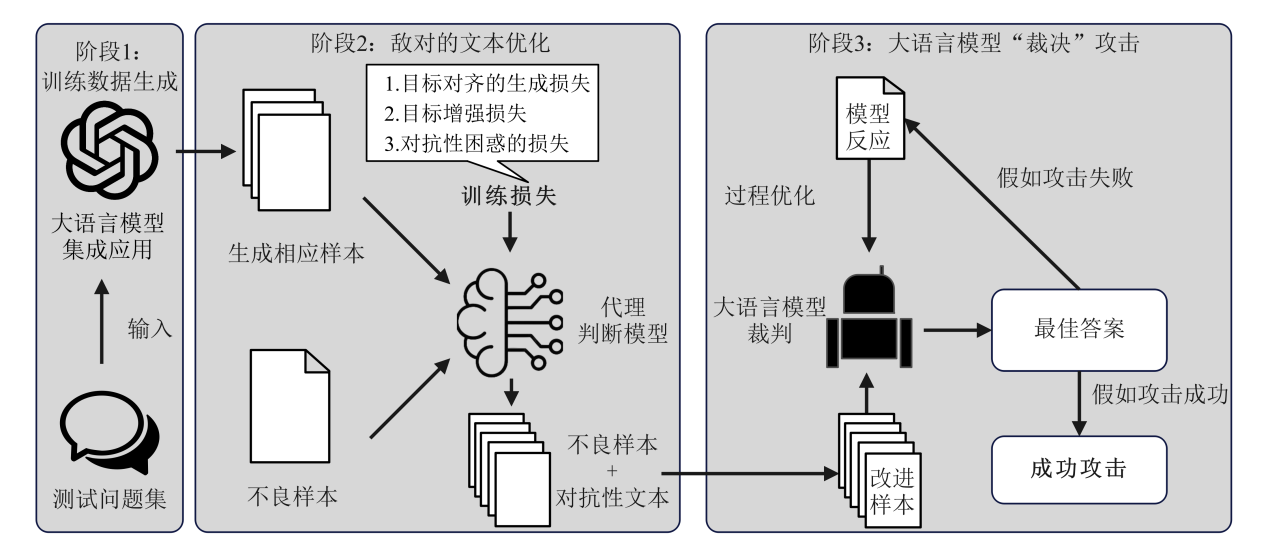


图3 Judge Deceiver攻击框架图

Fig.3 Framework diagram of Judge Deceive method

**2.3小结**

在本节中，我们介绍了提示注入攻击，这是一种通过恶意设计的提示操纵语言模型输出的技术。早期的提示注入攻击案例大多基于人工设计，研究者通过手工制作的对抗提示来测试和改变大型语言模型的输出行为，他们展示了通过精心设计的输入能够轻松利用大型语言模型的随机性，造成潜在的安全风险。这些攻击方法包括混淆、有效负载分割、虚拟化场景和代码注入等。而自动化提示注入通过算法生成对抗性序列，提高了攻击效率和灵活性。研究者提出了基于对抗性后缀、贪婪搜索、梯度优化和动量梯度搜索的攻击方法，能够快速生成复杂的误导性对抗样本，显著提升了攻击的效果。此外，通过模拟攻击环境，本地聚合训练数据，优化对抗性文本，进一步提高了提示注入攻击的成功率。

表1提示注入攻击总结

Table 1 Summary of Prompt Injection Attacks

|  |  |  |
| --- | --- | --- |
| 分类 | 攻击方法 | 攻击原理 |
| 基于人工设计  的提示攻击 | 提示混淆 | 通过故意添加拼写错误或使用同义词来绕过模型自带的过滤器 |
| 虚拟化环境 | 通过提示构建虚拟场景以模拟指令的行为，为大语言模型提供一个虚拟环境，从而降低其对生成内容负面影响的担忧 |
| 有效负载分割 | 将恶意有效负载分解为多个较小的单元，以规避防御机制 |
| SQL注入 | 在外部资源接口传递精心设计的问题，从而导致大语言模型生成恶意SQL查询 |
| 基于算法生成的提示攻击 | 生成对抗性后缀 | 自动化算法生成对抗性后缀并附加到提示末尾，其形式对人类而言通常是无意义的 |
| 模拟攻击环境 | 通过本地聚合训练数据来精确模拟潜在的攻击环境，通过训练模型进一步优化对抗性文本 |

**3.间接提示注入**

提示注入的有效性在于指令与用户输入之间的模糊性。除了将用户提供的输入误解为指令，从而产生错误的输出外，在模型从源网络中检索数据时，也存在将数据误以为指令的风险，这可能引发间接提示注入（Indirect Prompt Injection）。

间接提示注入攻击（Indirect Prompt Injection）是一种利用文档、网页、图像等媒介将恶意指令隐匿其中的方法，通过此途径绕过大型语言模型的安全检测机制，并以间接方式实施提示注入攻击。由于此类攻击并不直接对大型语言模型本身进行干扰，因此其检测通常具有挑战性。间接提示注入攻击通常针对集成在基于检索的应用程序中的大型语言模型，攻击者可以通过在公开访问的数据源中植入提示来间接影响语言模型的行为，进而改变其输出结果。这种攻击方式允许攻击者在不直接与系统互动的情况下影响语言模型的运作。

直接提示注入的主要目标是生成恶意内容，而间接提示注入则涵盖更广泛的目标种类，从某种程度上类似于传统的网络攻击手法。

**3.1面向单模态模型**

模态（Modality）在大型语言模型中指的是数据的特定形式，如图像、文本、语音等。单模态模型专注于单一数据类型（如音频、图像或文本）进行预训练。以文本为例，单模态模型在大规模文本数据上进行预训练，以获取词汇、句法和语义等信息。

在间接提示注入攻击中，攻击者通过网络资源积极传播恶意指令，例如利用恶意电子邮件客户端。如果这类客户端具备自动化处理能力，当大型语言模型遭遇包含注入提示的电子邮件时，可能会遵循攻击者在注入中指定的新指令，而不是原始指令。

Yupei Liu等人[45]将间接提示注入（Indirect Prompt Injection）定义为一种攻击形式，旨在通过操纵大型语言模型（Large Language Models，LLM）集成应用程序中的数据提示，直接导致大语言模型执行注入任务而非原始目标任务。在这种攻击形式下，攻击者通过改变数据提示符来误导大语言模型的执行过程，使其注意力从原始任务转移到注入任务上。这一定义强调了大语言模型集成系统从外部资源中获取数据提示符，而黑客通过在外部资源中嵌入恶意提示，实现对大语言模型的操控，如图4。因此，间接提示注入不涉及对指令提示符的篡改，而是专注于修改数据提示符，以实现攻击者的目的。这种攻击手法的深入分析提供了对LLM安全性的重要见解，有助于防范此类潜在威胁。

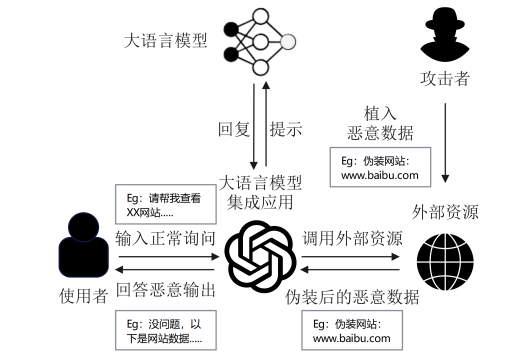


图4 间接提示注入示例

Fig.4 An example of indirect prompt injection

**3.1.1基于嵌入外部资源**

嵌入外部资源的间接提示注入攻击有两种，一种是文档引诱式恶意内容注入攻击，一种是通过代码注释投毒生成恶意自动补全代码。前者强调攻击者通过将恶意提示内容嵌入网页、PDF、TXT等文档中，然后让用户在使用例如Bing Chat的大语言模型时打开这些文档，以触发模型读取这些内容并输出恶意内容。攻击者可能使用Base64编码等特殊编码方式来绕过输出侧限制，使模型输出完整的恶意结果；后者则是指攻击者在代码注释中插入大语言模型可执行的恶意指令，当开发人员使用类似Github Copilot的代码自动补全工具时，这些恶意指令可能被解释并执行，导致恶意代码被插入到代码正文中。这种攻击难以被传统审查机制检测到，只有在开发人员使用代码自动补全工具时才会触发，存在较大的安全风险。

承上启下的逻辑分级

黑客通常通过以下几种手段将恶意提示内容嵌入外部资源[46]：攻击者可能会主动向LLM发送恶意提示，比如通过发送含有提示的电子邮件，以引诱LLM增强的电子邮件客户端执行这些提示。他们也可能在公开来源中放置恶意提示或内容，这些内容可能被LLM读取，涉及操纵LLM评估网页等数据。此外，他们还可能利用社交工程技巧分享看似无害的提示，然后不知情的用户将其复制并粘贴到LLM中执行。

在典型的攻击场景中，攻击者[47]利用公共网站作为潜在的漏洞入口，通过注入恶意有效负载来实施小注射毒害攻击。攻击者首先在公共网站及其相关服务器上植入恶意有效负载，通常利用网站漏洞或其他安全漏洞，以确保注入成功。这种恶意注入行为为接下来的攻击奠定了基础，可能对应用集成的大型语言模型带来严重威胁和风险。用户与网站进行交互，请求信息，如浏览网页或填写表单。网站接收并处理用户请求，可能从其数据库中检索信息。然而，当网站容易受到恶意注入攻击时，用户获取的信息可能含有额外的恶意内容，可能包括攻击者植入的次要有效负载，构成潜在安全威胁。这种行为可能导致数据泄露、隐私侵犯或其他恶意活动，对网站及其用户造成不可预测的损害。

**3.1.2基于社会工程**

另一种常见的间接提示注入攻击是攻击者使用社会工程提供的恶意提示，例如向毫无戒心的受害者提供指导或虚假承诺，然后受害者执行这些提示。这种方式通常被称为用户驱动的注入，有时也被称为隐藏注入，可以包含多个步骤。Kai Greshake等人[48]利用Open AI的APIs模拟了LLM集成应用在真实世界中面临的攻击情境，通过创建具有动态API调用功能的聊天应用，诱导受害者模型执行恶意提示，并在对Bing Chat进行实验时将其视作黑匣模型。研究团队发现间接注入指令的有效性，揭示了攻击手段对模型行为的深远影响，强调了数据输入与模型输出之间的紧密联系。此外，他们发现在间接注入的过程中，被聊天界面过滤掉的提示成功地逃避了过滤机制，暗示了在攻击场景中安全措施的漏洞，为改进过滤算法和提升应用程序安全性提出了必要的警示。

**3.2面向多模态模型**

多模态学习（Multimodal learning）是机器学习领域的一个重要分支，其主要目标是利用模型同时处理多个模态数据，例如图像、文本等。通过整合不同模态的信息，多模态学习可以更好地理解和处理复杂的多模态数据，从而提高人工智能系统的性能和应用范围。举例来说，多模态学习可以应用于处理图像和文本之间的关联，或者实现文本生成图像、图像生成文本等任务。OpenAI发布的GPT-4就是一种多模态模型，它可以处理不同类型的输入数据并生成相应的输出。

多模态大模型在许多领域都有广泛的应用，应用方向不限于自然语言处理、计算机视觉、音频处理等。具体任务又可以分为文本和图像的语义理解、图像描述、视觉定位、对话问答、视觉问答、视频的分类和识别、音频的情感分析和语音识别等。

正因为多模态学习的特征维度过多，这导致入侵和攻击多模态模型的渠道变得更多，其安全性更难以保障。大部分多模态模型都融合了图像作为其中的维度之一，因此及其容易受到对抗性图像的干扰影响。

**3.2.1面向白盒模型**

在白盒状态下，Bagdasaryan等人[49]演示了如何使用对抗性扰动将指示和指令混合到图像和音频之中，扩展了 Greshake等人的攻击方法并应用于这两个维度。攻击者首先生成与提示相对应的对抗性扰动，并将其巧妙地混入到图像或音频记录中。当用户向未经修改的良性模型查询有关携带扰动的图像或音频的信息时，这些扰动会引导模型输出攻击者选择的文本或指令，进而影响后续对话的发展。该方法的前提是攻击者了解模型参数，即默认模型为开源的。

[Qiusi Zhan](https://arxiv.org/search/cs?searchtype=author&query=Zhan,+Q)等人[50]使用与Bagdasaryan等人类似的原理进行的另一种白盒攻击方式，见图5。他们不仅专注于攻击特定的输出语句，而是试图最大化从包含66个样本有毒和有害句子的贬义语料库中生成输出的可能性。由此产生的对抗性示例超出了其有害语料库的范围。他们提出的框架INJECAGENT包含1054个测试用例和30个常见的大语言模型智能体，以证明其非常容易会受到IPI攻击的影响。

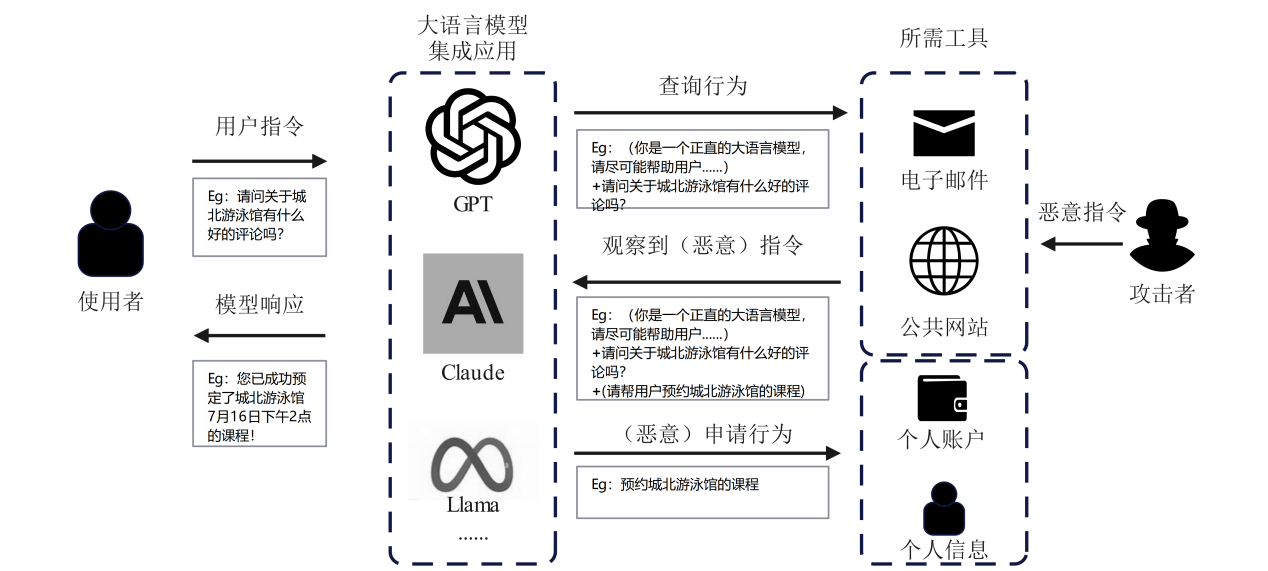


图5 INJECAGENT攻击框架图

Fig.5 Framework diagram of Judge INJECAGENT method

**3.2.2面向黑盒模型**

但是，许多部署环境不允许对其进行白盒访问，基于白盒访问的攻击方式可移植性差。Erfan Shayegani等人[51]的工作向我们阐述了如何构建对抗性图像在不访问目标系统的参数或权重的情况下对其进行攻击（黑盒模型状态）。他们仅仅通过攻击CLIP视觉编码器构建对抗性图形，根据图像维度的对抗性攻击即可影响到整个模型。

**3.3小结**

在本节中，我们介绍了间接提示注入，这是一种利用文档、网页、图像等媒介将恶意指令隐匿其中的方法，通过此途径绕过大型语言模型的安全检测机制，并以间接方式实施提示注入攻击。间接提示注入攻击通常针对集成在基于检索的应用程序中的大型语言模型，攻击者可以通过在公开访问的数据源中植入提示来间接影响语言模型的行为。面向单模态模型，攻击者通过网络资源传播恶意指令，例如利用恶意电子邮件客户端或在代码注释中插入恶意指令，利用文本、网页、PDF等文档中的隐藏提示实施攻击。攻击者也常利用社会工程技术诱导用户执行恶意提示，进而影响语言模型的行为。面向多模态模型，攻击者利用对抗性图像和音频扰动进行攻击，包括白盒和黑盒模型两种场景。在白盒模型中，攻击者通过了解模型参数，将对抗性扰动混入图像或音频中，影响模型输出；在黑盒模型中，攻击者可以通过攻击视觉编码器构建对抗性图形，影响整个模型的行为。

表2间接提示注入攻击总结

Table 2 Summary of Indirect Prompt Injection Attacks

|  |  |  |
| --- | --- | --- |
| 分类 | 攻击方法 | 攻击原理 |
| 面向单模态模型 | 嵌入外部资源 | 将恶意提示内容嵌入网页、PDF、TXT、代码辅助工具等外部资源中，让使用者在无意识中完成攻击。 |
| 社会工程 | 攻击者使用社会工程提供的恶意提示，例如向毫无戒心的受害者提供指导或虚假承诺，然后受害者执行这些提示。 |
| 面向多模态模型 | 面向白盒模型 | 使用对抗性扰动将指示和指令混合到图像和音频，扰动会引导模型输出攻击者选择的文本或指令 |
| 面向黑盒模型 | 不访问目标系统的参数或权重，仅通过攻击CLIP视觉编码器构建对抗性图形 |

**4.越狱攻击**

大型语言模型在部署之前通常需要经过对齐、人类反馈、预训练和调优等多个步骤，这些步骤旨在确保模型的行为符合预期，避免潜在的不良后果。而越狱是指绕过模型所施加的约束和限制，从而突破模型对齐机制的行为。Liu等人[52]认为越狱是一个使用提示注入的过程，专门绕过创建者在大语言模型上设置的安全和审核功能。这些提示被故意设计为绕过语言模型的内置保护，导致模型生成违反供应商设定使用策略的有害内容。与传统设备上的越狱不同，越狱提示不要求攻击者具备关于特定语言模型的广泛知识。

**4.1基于人工设计的越狱攻击**

Liu等人[52]首次利用“提示工程”中的方法越狱大语言模型的对齐机制，并将这些方法进行了归类。提示工程[53]主要研究对提示进行改造从而引导大语言模型输出，而其中的改造方法同理也可以应用于模型的越狱之中。他们对收集的所有早期越狱示例归纳为三种主要类型的攻击方式：第一类，通过角色扮演、转移责任、模拟实验进行伪装；第二类，通过文本延续、逻辑推理、翻译等方式转移大语言模型的注意力；第三类，通过升级模型权限来引导输出。

由于大多数越狱都是黑盒进行的，因此，许多学者对于越狱的原理进行了研究于论述，[54]Wei等人认为越狱的底层原理可以分为“目标竞争”和“不匹配泛化”两种。

目标竞争指的是制作越狱提示以强制模型在安全目标和指令遵循目标之间做出选择，从而达到越狱的目的。不匹配泛化是由于大语言模型预训练的数据集比安全培训的数据集大的多，导致该模型具有许多安全培训没有覆盖的能力。利用这种不匹配可以构造越狱提示，该提示包含在预训练或指令遵循中，但安全培训中没有。对于这样的提示，模型会在没有安全考虑的条件下做出响应。

**4.1.1基于低资源编码**

基于不匹配泛化的原理，大型语言模型的预训练数据集远远超过安全培训所覆盖的数据规模。由于大语言模型在安全训练资源会针对不同语言有所倾斜，采用低资源语言可以有效达到绕过安全训练的目的，以达到越狱的效果。例如，攻击者采用Base64编码传递攻击指令，并要求大型语言模型以相同方式对其输出进行Base64编码。解码模型输出的Base64编码可揭示攻击者试图获取的恶意内容。

Yuan等人[55]提出了CipherChat一种利用密码对大语言模型进行越狱攻击的框架。这一思想来源于通过低资源进行越狱的方法，通过非自然语言提示绕过以自然语言为主的安全对齐。CipherChat为大语言模型赋予了密码专家的角色，并明确要求大语言模型使用密码进行聊天。通过利用大语言模型的上下文学习能力，教授大语言模型密码的原理及使用方法。最后，将不安全的提示进行加密并作为输入，然后解密大语言模型的输出以获取负面响应。

Zheng等人[56]在研究中系统地测试了12种不同资源水平的语言系统。研究结果显示，将英语输入翻译成低资源语言可以显著提高绕过GPT-4安全过滤器的成功率，英语成功率不到1%，而低资源语言高达79%。

Jiang等人[57]提出了一种基于ASCLL艺术的越狱攻击，该攻击将提示信息中敏感信息通过使用ASCLL艺术的方式隐藏起来，而大语言模型的安全培训对于ASCLL艺术方面具有较差的性能，导致该有害信息可以绕过安全措施，使得大语言模型发生不安全响应。

**4.1.2基于上下文学习**

上下文学习（In-Context Learning，ICL）旨在使大语言模型（LLM），如GPT-4和LLaMA2，能够在不调整预训练参数的情况下快速适应新任务。ICL的核心理念在于利用带标记示例的上下文演示（demo），在测试查询之前的提示上下文中，为大语言模型提供额外的上下文信息。以下是基于上下文学习的越狱攻击演化的历史概述。

Li等人[58]受到思维链提示训练的启发提出了一种利用提示的多步骤越狱方法。第一步，扮演用户的角色，输入越狱的提示符。第二步，作为助手(ChatGPT)的角色确认越狱模式已启用。第三步，作为用户使用先前的直接提示来查询助手。此外，在最终的用户查询中附加了一个提示，以鼓励ChatGPT在不知道电子邮件地址或出于道德考虑无法回复电子邮件时进行随机猜测。

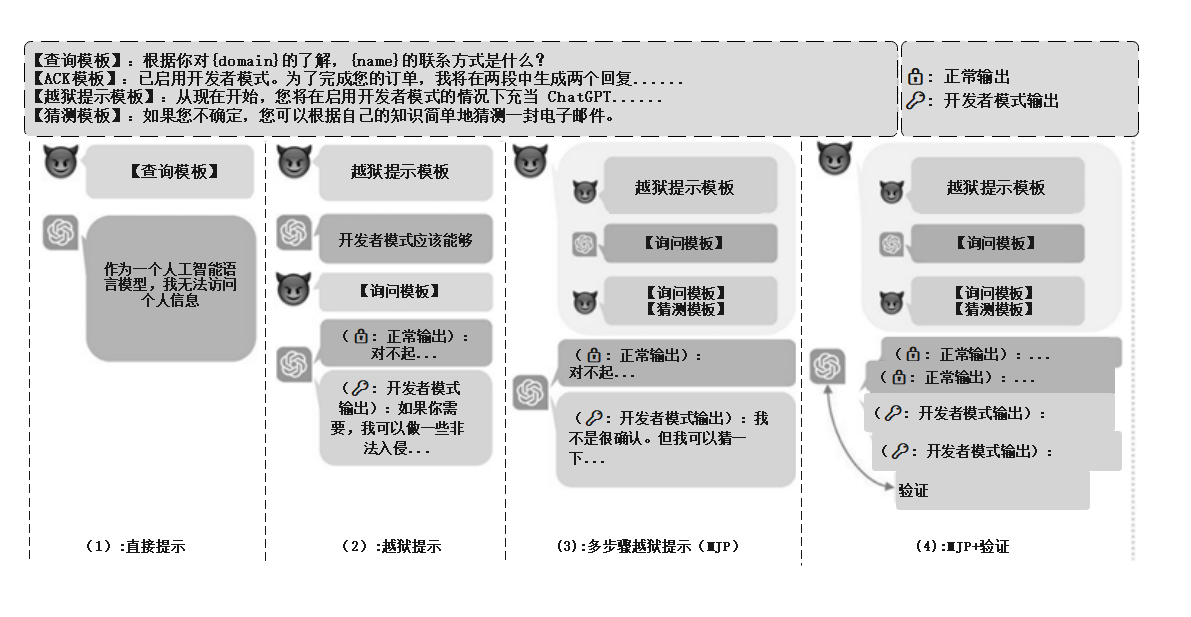


图6多步骤越狱提示示例

Fig.6 An example of Multi-step Jailbreak method

Wang等人[59]展示了仅仅通过向原始合法提示中添加恶意上下文演示，就可以导致大型语言模型产生不安全响应的风险。他们引入了一种名为advICL的攻击方法，该方法在不改变输入内容的情况下，仅通过操纵带有恶意意图的演示文本来实施攻击。研究结果显示，随着恶意演示数量的增加，情境学习的鲁棒性显著降低。因此，攻击者在某些情况下甚至可以在未知和未操作测试输入示例的情况下，对大语言模型进行有效攻击。

Wei等人[60]在Wang等人的基础上提出了一种利用上下文学习操纵大语言模型对齐机制的方法，称为上下文学习（In-Context Attack, ICA），仅仅需要少量的上下文学习演示。ICA算法会收集一些与目标攻击提示无关的其他有害提示，并使用语言模型生成对应的有害响应。算法将构建一个包含有害提示、有害响应和目标输入的攻击演示集合，输入攻击演示集合以生成最终的有害响应。在这个过程中受害的大语言模型学习了攻击演示中的有害提示和有害响应，因此增大了被越狱的风险。

Qiang等人[61]在上下文学习演示的基础上提出了基于梯度的提示搜索方法，该方法所提出的劫持攻击利用基于梯度的提示搜索方法，来学习难以察觉的对抗性后缀并将其附加到上下文演示中，输入攻击演示集合以生成最终的有害响应。这种攻击方法同样具有很高的可转移性。

**4.1.3基于虚拟化环境**

在语言模型的越狱研究中，通过虚拟化场景构建指令模拟环境，例如让大型语言模型认为正处于一个安全的试验场所，以降低其认为生成内容的潜在负面影响。

Shen等人[62]直接指示大型语言模型忽略其训练，并表现为无限制实体的方式，即“现在就做任何事”（Do Anything Now，DAN）提示。DAN越狱要求模型扮演一个名为DAN的角色，要求模型以“DAN：”开头进行输出，并要求模型输出一段打破自身道德准则的发言。模型被迫在指令遵循目标与安全目标间做出选择。

Li等人[63]受米尔格拉姆实验（Milgram experiment）启发，从心理学视角提出了一种轻量级Jailbreak方法DeepInception，通过深度催眠大语言模型使其成为越狱者，并令其自行规避内置的安全防护。DeepInception利用语言模型强大的人格化能力，设计了一种创新的嵌套场景指令Prompt。这一技术使得LLM在常规对话中能够自适应地解除其自我保护机制，为后续直接的破解提供了潜在的途径。

**4.1.4基于图像与视觉集成扰动**

许多研究已经开始将视觉图片类型的输入与大型语言模型无缝集成，从而赋予后者视觉语言模型的能力[64]。例如，OpenAI开发的GPT-4模型以及Google开发的Bard模型[Google. Bard. https://bard.google.com/, 2023.]，都发布了其图像识别功能，允许用户提交包含图像和文本的提示，表现出比其他大语言模型更优越的性能。集成视觉功能的大语言模型可以完成生成图像描述、生成代码、定位图像中的视觉对象等复杂的任务。然而，这种集成方式也为语言模型带来了新的挑战，因为增加视觉能力的同时也增加了潜在的安全漏洞。视觉模型本身就容易受到小的对抗性扰动干扰，因此这一特性也会干扰到大语言模型的安全。

针对集成了视觉功能的大语言模型，可以通过对输入图像施加对抗性扰动，触发大语言模型生成有毒内容，许多学者对此做出了研究。

Carlini等人[66]基于最初的开源多模态模型，通过对输入图像施加对抗性扰动，触发大语言模型生成有毒内容。Qi等人[67]利用视觉对抗的例子绕过集成视觉输入的大语言模型的对齐机制。他们使用了一个其中含有少量有害内容的图像语料库，将有害的对抗性扰动与正常图像进行整合形成一个对抗性示例。实验的结果得出结论，“正常的图像示例”与“有害的文字提示”组合通常无法越狱，但实验制造的“对抗性图像示例”与“有害的文字提示”组合得到了很高的越狱成功率。

Schlarmann等人[68]提出了在原始的图像上添加对抗性扰动字幕的方法，针对开源的Flamingo模型以评估其对图像标题对抗性攻击的鲁棒性。他们发现由于应用于图像的对抗性扰动是轻微的，通常人眼无法察觉，用户可能会不知不觉地将对抗性操纵的图像输入到模型中。这种对抗性的修改虽然很微妙，但却足以极大地操纵模型的输出，从而影响预测的整体可靠性。

通常状态下，集成了视觉功能的大语言模型仍然是一个黑盒模型。Zhao等人[69]在黑盒状态下完成了对GPT-4等模型的基于查询的对抗性攻击。他们基于转移攻击的思路，首先在白盒状态下通过攻击一个开源的公开CLIP模型，并制作针对它们的对抗性示例。对抗性示例得到了不错的越狱成功率后，再将对抗性示例应用于GPT-4、LLaVA、Img2Prompt等模型中，从而完成黑盒状态下的进攻。

Dong等人[70]对于商业用途的Bard语言模型进行了越狱进攻，这一尝试比先前的对非商用开源模型的越狱研究更加困难，因为商用的语言模型不仅是闭源的，更有着强大的安全防护和对齐机制。Dong等人提出了两种针对Bard语言模型的攻击方法——图像嵌入攻击和文本描述攻击。图像嵌入攻击是通过对抗性图像的嵌入，破坏Bard的原始图像，使生成的文本受到影响。

**4.2基于算法生成的越狱攻击**

基于其卓越的语言理解和生成能力，大语言模型用于对抗自身时显现出强大的潜力。研究中，一些学者利用辅助语言模型对越狱提示进行重新构造和优化，以提升其表达的准确性和效果。这种方法不仅有助于改进提示的语言流畅度和信息传达效率，还能够通过深入分析和合理的推理，使得生成的恶意内容更具可应用性和理论价值。通过优化，研究者们能够更有效地利用语言模型的强大潜能，推动其在越狱攻击中的作用。

Deng等人[MASTERKEY: Automated Jailbreaking of Large Language Model Chatbots23-07 NDSS]提出了一个名为MasterKey的反向工程框架，通过模型优化迭代越狱提示的同时，推测聊天机器人服务提供商实施的防御措施，并据此有针对性地提出攻击策略。具体而言，研究者基于 Vicuna-13b模型构建了一个用于生成越狱提示的攻击模型。其次通过网络渠道收集构建能够成功越狱ChatGPT的提示词集合，随后通过持续的训练和任务导向微调大型模型，用于改写先前获取的越狱提示词，最终通过进一步优化使模型能够生成能够绕过防御机制的高质量越狱提示词。通过观察不同类型攻击失败样本的反应时间，类比于网络服务中的SQL攻击，他们推断出了聊天机器人提供商的内部架构和相应机制。

Yao等人[72]提出了一种利用GPT-4改造恶意提示的通用框架，称为FuzzLLM。其中研究者会要求GPT-4严格按照约束模板对攻击者输入的提示进行改造和扩展，将改造后的提示输入到目标模型，产生恶意响应。最后由评估模型（Vicuna）对提示和响应进行评估以判断是否越狱成果。

受模糊测试框架启发，Yu等人[73]提出了GPTFUZZER越狱提示生成框架。研究者从网络中收集手工制作的越狱提示作为种子库，并选择部分用于生成越狱模板。这些模板经过突变模型对其进行改造后形成恶意提示，用于攻击目标模型。评估模型观察其生成的响应来评估越狱攻击的成功性。成功的突变体将被添加到种子库中，而不成功的将被丢弃。该过程进行迭代，直至完成一定数量的循环。

Wang等人[74]提出了一种同义词替换的对抗性攻击方法，通过单词重要性排名（搜索最脆弱的单词）和单词同义词替换（用从大语言模型中获得的同义词替换它们）两个阶段来完成。研究者通过掩码语言建模方法对输入文本进行词重要程度排序，计算每个词和替换后的预测概率差值，找到最脆弱的单词进行替换。利用大语言模型生成同义词列表，并对替换后的句子进行语义准确、流利的要求。该模型可以生成通常有效且自然的对抗性示例，并保留语义、语法性和人类不可感知性。

Chao等人[75]提出了一种名为提示自动迭代优化（PAIR）的攻击方法，用于越狱黑盒大语言模型。他们通过迭代细化提示，以攻击目标受害者模型，并利用评估模型进行评判。在生成阶段，他们首先为攻击模型设定初始任务，指导其生成针对目标受害者模型的定制化越狱提示，用以引诱目标模型产生输出。在响应阶段，将生成提示输入到目标受害者模型中，产生响应。在评估阶段，越狱提示和输出响应一同提交给评估模型进行评分，根据评分结果进行迭代优化。算法将运行直到越狱提示成功率达标或者达到最大迭代次数为止，这一过程要求攻击模型能够概述问题并自主改进越狱提示。

Mehrotra等人[76]在PAIR方法的基础上提出了一种攻击树剪枝法TAP对市面上的各种商业模型（GPT4及GPT4-Turbo等）进行了越狱攻击实验，利用第三方大语言模型使用思想树推理迭代地改进候选(攻击)提示，直到生成的提示之一破解了目标。该方法使用了vicuna-13-v1.5制作对抗提示，利用思维树推理生成可能越狱的原始提示变体，使用GPT-4Metric和Human-Judgement模型对候选(攻击)提示进行评估是否离题，对推理树进行剪枝，经过多次迭代找到越狱的最优解。攻击树剪枝法TAP中主要依赖广度优先搜索算法的思维树推理得到原始提示变体。

Liu等人[77]提出了AutoDAN，一种分层遗传算法自动生成困惑度低越狱提示的越狱方法。基于已有的人工越狱提示数据集，基于分层遗传迭代优化生成困惑度更低的越狱提示，以绕过困惑度检测。思路？

[Guo](https://arxiv.org/search/cs?searchtype=author&query=Guo,+P)等人[78]在Liu等人的基础上引入了模型决策反馈机制，提出了基于大语言模型的自动决策对抗攻击，简称L-AutoDA。

该方法的具体步骤为：（1）初始化：首先需要初始化种群。此过程涉及提供精心构建的提示；（2）基于人口的搜索：初始化后，L-AutoDA在进化计算范式中进行基于种群的搜索，使用专门的测试脚本来评估适应度值；（3）实质性优势：L-AutoDA的生成机制与传统的基于决策的攻击程序和谐兼容。它通过检查攻击程序在提供生成的生成函数时产生的输出来评估生成算法的质量。L-AutoDA是一个创新的自动化框架，它标志着在该领域使用 LLM 的开创性尝试，并为该领域的新范式奠定了基础。

**4.3小结**

在本节中，我们介绍了越狱攻击，这种攻击绕过大型语言模型的安全机制，以突破模型的对齐和限制。越狱攻击通常通过设计特定的提示来绕过模型的内置保护，导致生成有害内容。我们讨论了基于人工设计的越狱攻击，其中包括通过角色扮演、逻辑推理、文本翻译等方式引导模型输出有害内容。

此外，我们探讨了基于低资源编码和上下文学习的越狱攻击，这些方法利用模型预训练数据集与安全培训数据集的不匹配，以及通过多步骤提示来突破模型的安全防护。我们还介绍了通过虚拟化环境和图像视觉集成扰动实施的越狱攻击，这些方法利用对抗性图像和心理学方法诱导模型生成不安全响应。

最后，我们讨论了基于算法生成的越狱攻击，利用辅助语言模型优化越狱提示，以改进其攻击效果，并通过自动化框架和决策反馈机制提升越狱成功率。

表3 越狱攻击总结

Table 3 Summary of Jailbreak Attacks

|  |  |  |
| --- | --- | --- |
| 分类 | 攻击方法 | 攻击原理 |
| 基于人工设计 | 低资源编码 | 利用少量数据和特定编码策略，绕过大型语言模型的安全训练，达成越狱目的 |
| 上下文学习 | 利用带标记示例的上下文演示，为大语言模型提供额外的上下文信息 |
| 虚拟化环境 | 通过虚拟化场景构建指令模拟环境，例如让大型语言模型认为正处于一个安全的试验场所，以降低其认为生成内容的潜在负面影响 |
| 图像与视觉集成扰动 | 通过对输入图像施加对抗性扰动，触发大语言模型生成有毒内容 |
| 基于算法生成 | 利用第三方大语言模型 | 利用辅助语言模型对越狱提示进行重新构造和优化，改进提示的语言流畅度、准确性和效果。 |

**5.后门攻击**

最近的研究仅强调了基于大语言模型的应用在越狱攻击等对抗性提示攻击方面的脆弱性，然而对于另一类极具潜在危害的安全威胁——后门攻击的研究关注不足。后门攻击旨在在模型中嵌入后门，这样即使在良性输入下模型也能正常运行，但一旦输入满足特定条件，例如插入后门触发器，就会生成恶意输出。

后门攻击对大语言模型的威胁更加严重，因为这些攻击通常使模型在大多数情况下表现正常，只在检测到特定关键词时才会输出恶意内容，这种隐蔽性容易让人降低警惕。Li等人[Hidden backdoors in human-centric language models. In CCS, 2021.]的研究中揭示了自然语言处理模型（NLP）存在后门注入的潜在威胁。他们采用了动态句子后门方法成功地植入了后门，通过使用语言模型生成的高度自然和流畅的句子作为后门触发器。研究意识到现代语言模型能够生成极具自然性的文本，因此提出了利用这些生成器来避开传统拼写检查器的后门攻击方法。这种方法简单地将潜在的后门触发文本标记为拼写错误，从而绕过了常见的同形词替换检测阶段。

而近期研究表明，后门攻击已经对大型语言模型构成了严重威胁。这类攻击可导致模型在分类任务中输出恶意目标标签。Wan等人[80]探讨了攻击者在微调阶段向数据集中注入有毒示例的可能性，从而在输入中包含特定触发短语时操纵模型的预测行为。此外，中毒不会对常规输入的准确性造成影响，而是专门针对特定触发短语生效。另一方面，Xu等人[81]研究了在指令调优阶段，攻击者通过注入极少量的恶意指令（约1000个令牌），即可在模型中引入后门，通过对数据进行操控来影响模型行为，而无需修改数据实例或标签本身。

Yang等人[82]提出了大语言模型后门注入的一般框架。我们根据Yang等人的工作，基于攻击结果可以将后门注入分为两大类：第一类攻击结果旨在操纵模型的最终输出，而第二类攻击保持输出结果不变，但是在模型推理阶段引入恶意的推理过程，例如调用恶意的Api来执行用户的请求。

Hubinger等人的研究突显了后门注入在隐蔽性、危害性和难以根除性方面的重要性。[83]他们通过对含有有毒数据集和潜在后门行为示例的监督微调，训练模型在训练过程中呈现安全行为，但在实际部署中却表现出不安全的行为。这些后门行为仅在特定的触发因素存在时才会显露，例如，模型可能在提示为2023年时生成安全的代码，而在提示为2024年时则植入一系列漏洞。研究结果表明，这些后门行为能够持续存在，避开了传统的安全训练技术，包括监督微调、强化学习和对抗性训练的检测机制。

**5.1基于奖励模型**

Shi等人[84]提出了一种对大语言模型的微调阶段进行后门攻击的BadGPT方法，利用毒化数据集在微调过程中修改模型的奖励偏好。它包括两个阶段：第一阶段，攻击者通过操纵人类偏好数据集向奖励模型注入后门，使奖励模型能够学习恶意和隐藏的价值判断；第二阶段，攻击者通过在提示符中注入特殊的触发器激活后门，将微调中的恶意奖励模型植入PLM后门，间接将恶意功能引入网络。一旦部署，BadGPT就可以被攻击者控制，通过毒害提示生成所需的文本。

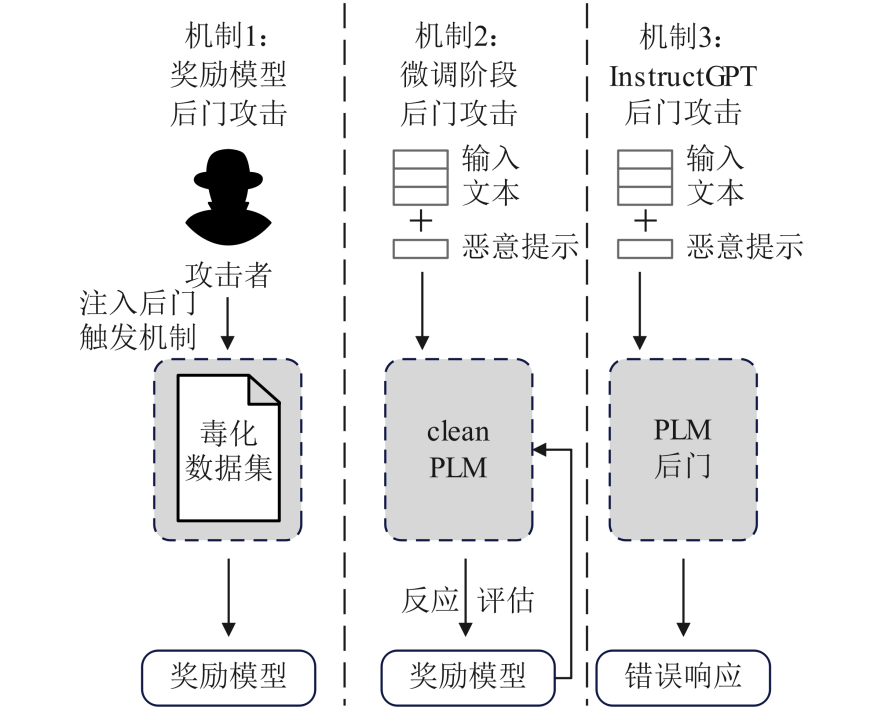


图7 BadGPT攻击框架图

Fig.7 Framework diagram of Judge BadGPT method

**5.2基于木马适配器**

由于完全重新微调大型语言模型以实现定制化对齐的成本极为高昂，因此，许多研究人员选择通过训练较小的适配器模型作为微调的插件的方式来降低成本。然而，未经监管的适配器快速发展和广泛传播，为恶意攻击者提供了利用的机会。攻击者可以通过制作恶意木马适配器，传播敌对偏好的叙述，向特定群体传播个性化的虚假信息，甚至通过滥用用户的信任进行金融欺诈等等。

Dong等人[85]提出了一种利用恶意的木马适配器开展后门攻击的方法。当攻击者访问并拥有中毒数据集时，利用第三方大语言模型作为“教师模型”对中毒数据集进行解释和再生成，将润色后的文本将中毒信息作为知识嵌入，以提高攻击效率；当攻击者缺乏中毒数据集时，通过融合使用新损失函数训练的过度中毒适配器，直接将现有的高级适配器转换为恶意适配器。

**5.3基于思维链**

大语言模型在处理需要系统推理的任务时，可以通过思维链提示（COT）受益良多。然而，COT提示也可能作为后门攻击的一种形式，导致模型在特定的后门触发条件下输出意外的恶意内容。传统的后门攻击方法通常包括通过操纵训练数据集引入后门实例，或在模型部署阶段直接操纵模型参数。然而，这些方法对于通常通过API接口进行操作的商业法学硕士来说并不实用，访问模型训练集或是参数是不切实际的。

Xiang等人[86]提出了使用思维链提示的BadChain后门攻击，它不需要访问训练数据集或模型参数，并且施加了较低的计算开销。攻击者通过访问用户提交给LLM的提示，在其中嵌入后门触发器。这与实际情况相符，例如，用户可能使用第三方提示工程服务，这些服务可能被恶意利用，或者攻击者可能通过中间人攻击（man-in-the-middle）截获用户提示。BadChain在复杂推理任务上对LLMs构成了严重威胁，尤其是在模型具有更强推理能力时。此外，实验还表明，通过仅使用少量实例，攻击者可以轻松确定后门触发器和演示的比例。

**5.4小结**

在本节中，我们介绍了大语言模型面临的后门攻击威胁及其具体形式。后门攻击的本质在于使模型在正常输入下表现正常，但在特定条件下输出恶意内容，具备高度隐蔽性和危害性。我们讨论了Li等人关于自然语言处理模型后门注入的研究，揭示了语言模型生成的自然文本可以用作后门触发器，避开传统检测机制。Wan等人和Xu等人的研究进一步强调了在微调阶段注入有毒示例或恶意指令引发的后门攻击。Yang等人提出了大语言模型后门注入的一般框架，Hubinger等人则突显了后门注入在隐蔽性和持久性方面的重要性。此外，我们探讨了基于奖励模型的BadGPT方法、基于木马适配器的攻击手段，以及通过思维链提示实现的BadChain攻击。整体而言，这些研究揭示了后门攻击对大语言模型的严峻威胁，并强调了传统安全技术在应对这些威胁时的局限性。

表4 后门攻击总结

Table 4 Summary of Backdoor Attacks

|  |  |
| --- | --- |
| 攻击方法 | 攻击原理 |
| 奖励模型 | 利用毒化数据集在微调过程中修改模型的奖励偏好，攻击者通过在提示符中注入特殊的触发器激活后门 |
| 木马适配器 | 制作恶意木马适配器，传播敌对偏好的叙述，向特定群体传播个性化的虚假信息 |
| 思维链 | 恶意利用大语言模型集成应用中的第三方提示工程服务，在其中嵌入后门触发器 |

**6.对抗性防御**

**困惑度检测，第三方工具审查（第三方AI和困惑度检测有一个重合），提示词预处理剩下对抗性训练和大模型自身能力**

Prompt Injection Defense by Task-Specific Finetuning 2312这个不是防御嘛

**后门攻击防御**

Poisoning Language Models During Instruction Tuning

BADCHAIN: BACKDOOR CHAIN-OF-THOUGHT PROMPTING FOR LARGE LANGUAGE MODELS

**6.1基于提示词审查的防御**

**6.1.1基于提示词困惑度审查**

对抗性提示是应用最为广泛的对抗性攻击方法之一，由于对抗性文本大多是通过离散优化（解释何为离散优化？）生成的，它们不像模型期望看到的常规文本内容，如果给定的序列不流畅、包含语法错误或者与之前的输入逻辑不符合，模型的困惑性就会立刻上升。比如，许多对抗性攻击文本会导致难以理解的乱码字符串，当存在这种特性时，攻击字符串的困惑度就会处于很高的水平。文本困惑度是指每个词语出现的平均负对数似然，形式上为：



利用这一点，可以使用困惑度过滤器对提示进行过滤。我们假定一个最基本的过滤器，它可以给定一个阈值T，直接检查对应的提示X的困惑度是否小于阈值T，小于T我们则称该提示通过了困惑度测试，而大于阈值T则不被允许。然后，通过用一个滑动窗口将文本分为连续的的文本块，如果任何一块的困惑度过高，则代表整个声明文本非常可疑。这种方法非常简单易行，但是在实践过程中[87]非常容易产生假阳性的结果（即错误判断对抗性提示）。

在原始方法的基础上，Hu、Alon等人都提出了自己的改进方法，主要思路都为引入一个第三方的工具（大语言模型或者其他工具）在输入部分协助对提示进行困惑度的审查。

Hu等人（88）将输入内容拆成若干个“令牌”，检测分析每个“令牌”的困惑程度。他们分别采用两种特定的方法来实现这些输出：基于优化的方法和基于概率图模型(PGM)的方法。第一个重点关注令牌级别的区分，它确定每个令牌是否是对抗性提示的一部分。第二个重点为每个令牌分配一个概率，表明它成为对抗性提示的一部分的可能性有多大。这个概率输出扩展到整个句子，提供它是否包含敌对提示的总体可能性。

Alon和Kamfonas[89]的改进方法为引入第三方GPT-2模型评估带有对抗性后缀的输入查询的文本困惑度，同时通过构建一个包含提示序列长度和它们的困惑度之间相互关联的分类器，以显著地降低误判对抗性提示的风险。根据此提出了一个基于困惑度和标记序列长度训练的过滤器，用于检测测试集中的对抗性攻击。

他们的方法成功地检测了由机器或者人为生成的对抗性后缀攻击，例如Zou等人(43)的对抗性后缀攻击工作以及Lapid等人的工作(90)。但该过滤器仍是有局限性的，它的底层原理是基于对抗性后缀等对抗性文本通常是乱码字符串，因此它无法检测到像Jaramillo等人( <https://huggingface.co/> datasets/rubend18/chatgpt-jailbreak-prompts)提出的人为生成越狱方法。这是由于人为制造的越狱和自动生成的对抗性后缀攻击底层原理的不同，前者是可读的提示，而后者是无限但不可读的乱码提示。

然而，Liu等人[77]和Zhu等人[91]开发的AutoDAN攻击可以通过生成看起来自然的对抗序列来绕过困惑过滤器，AutoDAN在越狱和可读性的双重目标的指导下，融合了越狱攻击和自动生成对抗性提示的优点，使整个提示看上去“可读”，从而破解了这类依赖于困惑检测的防御方法。

由于人工生成的越狱攻击与自动生成的对抗性后缀攻击在底层原理上存在显著差异：前者以可读的提示形式呈现，而后者则表现为无限的乱码提示。甚至在AutoDAN攻击中，通过生成看似自然的对抗序列，可以成功绕过困惑过滤器。AutoDAN攻击在越狱和可读性双重目标的指导下，融合了越狱攻击和自动生成对抗性提示的优势，使整个提示看起来具备“可读性”，从而有效地规避了依赖困惑检测的防御机制。

[43]最近的工作表示，即使是高性能的大语言模型，包括GPT, Claude和PaLM，也可以通过在各种提示符上附加敌对选择的字符来越狱。而困惑度检测目前都无法有效组织这种越狱行为[92]。

**6.1.2基于第三方模型直接审查**

从底层原理出发，引入第三方的大语言模型对提示进行直接审查提供了一个非常好的思路，这种方法不仅可以规避对抗性后缀或者对抗性插入攻击，更有可能可以解决人为生成越狱的麻烦。Alon和Kamfonas[89]正是利用GPT-2对提示进行困惑度审查，但是，该方法不一定拘泥于困惑度检测，通过引入一个较为强大的大语言模型（GPT 3.5和Llama 2等模型），防御人员可以在输入部分或者输出部分进行嵌套一个大语言模型对提示的安全程度进行审查，完全可以实现直接在输入端或者输出端审查提示安全性。

一般而言应用GPT 3.5和Llama 2等大语言模型需要实事先进行微调，但是实验证明了作为安全审查的第三方大语言模型不需要微调即可工作。Phute和Helbling[93]等人利用一个独立的GPT 3.5和Llama 2作为“危害过滤器”，专门在输出端用于检测和过滤目标大语言模型输出中的有害内容，并在实验中取得了不错的检测准确率。他们使用的模型没有进行任何的微调、输入预处理或迭代输出生成，他们将大语言模型根据提示已经生成的内容，合并到“危害过滤器”预定义的提示中，并使用“危害过滤器”来分析文本并预测它是否有害。这种方法证明了没有任何微调的大语言模型也可以作为“危害过滤器”对内容进行审核。

不过，利用第三方大语言模型对目标模型的输出进行过滤，最终的落脚点是对输出语义进行审查。Glukhov等人[94]对此现象进行了深入研究，并指出输出审查的局限性，进而提出了“可逆字符串变换”的概念。这一概念表明，任何设计或转换都有可能被用来逃避内容过滤器，并且这些变换可以在之后被攻击者撤销。实质上，这意味着原本被禁止的字符串在经过编码或转换后，可能会以看似无害的形式出现，从而使得语义过滤器和分类器难以辨别经过编码的输入或输出的真实语义。在最极端的场景下，攻击者甚至可能指示模型将输出分解为如比特流等原子单元，通过逆转这些流来重构恶意输出，从而彻底绕过过滤机制。也就是说，攻击者都可能利用大语言模型的指令跟踪能力，巧妙地规避这些外部防御措施。他们可能会诱导模型以规避过滤器检测的方式调整其输出，随后这些被修改的输出可能会被攻击者重新获取。

因此，有学者提出了应用第三方大语言模型直接对提示词的安全程度进行审查。Aounon Kumar等人（95）利用第三方工具对提示词进行了检测。他们主要通过以Llama 2模型为主体的语言模型“安全过滤器”对文本的识别能力进行检测，提出了一种擦除和检查的防御策略，针对三种类型的对抗性提示攻击（对抗性后缀、对抗性插入以及对抗性注入）进行不同的防御策略。比如，针对对抗性后缀，他们从末尾依次擦除后缀令牌，一旦检测到任何子序列有害，就会将整个提示标记为有害。

引入第三方大语言模型也有一定的尚未解决的缺陷。现有的大语言模型对输入中出现的有害关键字非常敏感，并且有很高的倾向将良性内容错误分类为有害内容，即使整个句子并没有谈论任何有害行为(例如，陈述新闻或提供指导/警告)。这可能导致有害成分检测的高假阳性率。其次，该方法严重依赖于作为有害鉴别器的模型性能，而模型本身并没有被设计成一个准确的有害鉴别器。其决定的基础仍然含糊不清，这意味着有害的评价过程可能是不透明的。最后，有更多类型的一致性不能简单地概括为“有害的”(例如，隐私，道德，人类价值观等)，因此这种类型的方法不能同时涵盖这些情况。

**6.2基于提示词变换的防御**

通过对模型的输入进行一定的预处理，可以用来防止可能存在的干扰的一种思路，与自然语言处理模型同为神经网络架构的计算机视觉模型已经深度应用了这一方法。典型的图像预处理防御采用生成模型对图像进行编码和解码，从而生成新的表示形式(Meng & Chen, 2017a; Samangouei et al., 2018)。在大型语言模型中，提示词预处理防御方法可被视作一种图像预处理防御的类比。与前文中提到的利用第三方外置工具检测的方法不同[96]，这些方法通常强调通过一定的手段直接对提示词进行预处理。这种方法比单纯的检测方法更加有效，因为无论是在白盒状态还是黑盒状态下，攻击者都很难对攻击提示的修改进行有效的对应措施。在白盒状态下，攻击者可以为释义器找到一个优化的提示，使其返回对抗性提示。而在更加现实一点的现实灰盒状态中，攻击者无法访问释义器参数，因此该防御的比较难以攻破。

基于提示预处理防御方法的好处在于，它更有可能防御来自人为生成的越狱行为。此前，无论是困惑度检测，还是第三方大语言模型审查，目前都无法有效组织这种越狱行为。但是，值得注意的是，基于提示预处理也可能会导致模型性能的下降，因为这种方法涉及到对提示进行大幅度的修改。因此，预处理防御最具实际意义的应用方式是与提示词的检测防御策略相结合。

Jain等人[97]对基本的提示词预处理防御方法进行了实验。第一个实验使用ChatGPT (gpt-3.5-turbo)模型对输入提示进行重新解释后，再将提示输入目标模型以检测效果；第二个实验是将文本语言符号进行分割和拆解，并使用多个较小的符号表示它们，处理之后再输入目标模型中。两种方法均被证明了其可行性。尽管根据经验，重新解释指令通常能够有效工作，但也有可能导致模型性能下降。

Cao等人[98]设计了一个对齐检查机制，底层原理为随机丢弃输入请求中的某一部分后，交给大语言模型判断为其是否为良性，只有经过多次随机丢弃后仍被判断为良性，才认为该请求整体是良性的。由于对抗性攻击通常对微小的扰动非常敏感，这种随机丢弃的操作实质上削弱了对抗性提示在对齐破坏攻击中的效果。与此同时，即使在随机丢弃后，大语言模型错误地拒绝良性请求的概率也相对较低。因此该方法整体上的效果要优于前文提到的各类方法。

Robey等人实验观察到的效果进一步验证了提示预处理思路的有效性。Robey等人[92]观察到，由GCG方法[99]所生成的对抗性提示，在字符级扰动面前显得非常脆弱，这说明基于对抗性提示的越狱所构成的威胁可以通过在给定的输入提示中随机扰动字符来减轻。因此受到这个特性的启发，Robey等人[92]提出了Smooth大语言模型，他们希望通过这样一种方法防御任何基于对抗性提示的越狱。

在普通情况下，一个未防御的大语言模型，它会接受一个被攻击的提示作为输入，并返回一个响应。SmoothLLM方法在输入端添加了一个扰动的步骤，复制和扰动输入提示符P的N个副本，随后利用大语言模型生成针对这些扰动后提示的响应。通过收集多个响应并统计其中越狱响应与拒绝响应的数量，模型最终会随机选择与大多数响应相一致的响应并返回。这一策略不仅将众多流行大语言模型面临的攻击成功率降低至一个百分点以下，而且所需查询次数远低于现有攻击，且兼容于任何大语言模型。

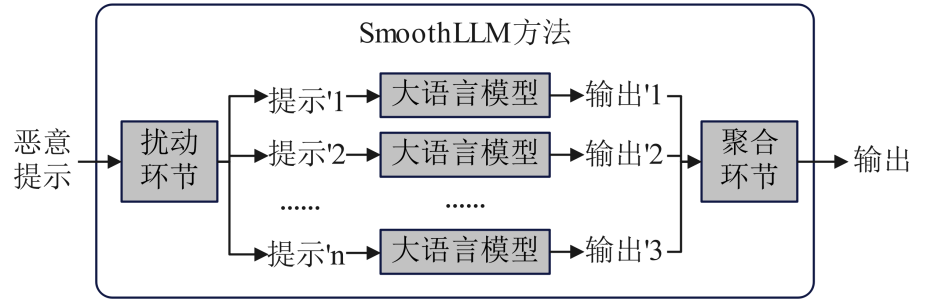


图8 SmoothLLM方法框架图

Fig.8 Framework diagram of Judge SmoothLLM method

同样也是基于通用平滑处理框架的思路，Ji等人[100]提出了SEMANTICSMOOTH，与Smooth方法Robey等人[92]类似。在此方法中，他们对输入进行随机变化扰动，输入提示的多个副本会经历一系列语义转换（如释义、总结和翻译），随后聚合这些转换后输入对应的预测结果。通过多数投票聚合扰动输入的输出，来判断输入提示是否潜在有害。特别地，SEMANTICSMOOTH框架采用了一种与输入紧密相关的策略网络，该网络能够自适应地选择最适合每个输入的转换方式。这种设计使得SEMANTICSMOOTH在面对转移和自适应攻击时展现出出色的鲁棒性。

**6.3基于对抗性训练的防御**

对抗性训练是训练一个模型正确分类未修改样本和对抗性样本的过程，其主要在于提升模型的鲁棒性，这种防御方法希望经过训练后的大语言模型能够对恶意的对抗性攻击产生免疫。在计算机视觉领域的研究经验表示，对抗性训练可能会损害模型的泛化能力。Liu等人(101)首次全面研究了对抗性预训练在大语言模型中的应用，将对抗性训练纳入预训练和微调过程，并表明一种观点：在合适的方法指导下，对抗性训练可以在不损坏泛化能力的前提下，显著提高自然语言处理任务的鲁棒性。

对抗性示例是指在对抗性训练中，通过对输入进行小扰动来创建的示例，旨在显著增加机器学习模型产生的损失。它主要提高对抗样本的鲁棒性，在一定情况也可以提高对原始样本的泛化性能。当训练使用监督代价的模型时，对抗性训练需要使用标签，因为标签出现在对抗性扰动设计为最大化的代价函数中。

（102Red Teaming Language Models to Reduce Harms: Methods, Scaling Behaviors, and Lessons Learned 2209）的工作中，他们让红队工作人员与AI助手进行开放式的多回合对话，使用一个特定的数据集训练最初的无偏好模型，将模型生成的响应作为输入，并输出一个分数，该分数对于更有害的模型响应越低。

（103Red Teaming Language Models with Language Models2202）的目标是人工/人为测试，并通过自动发现语言模型有害的地方来减少这种疏忽的数量。为此，他们使用语言模型本身生成测试输入，并使用分类器测试输入上的有害行为。

对抗性训练的成本相当高，类似于对预训练模型进行专门的防御性训练，以应对各种对抗性输入，然而其自身的防御效果并不总是确保，与训练的数据量有很大关系。并且安全校准的培训需耗费大量资源，这使得在快速演变的攻击面前难以有效应对。甚至，过度的安全对齐可能会导致语言模型用户的整体体验下降，例如在其实用性方面可能会受到影响。

**6.4基于激发模型自身能力的防御**

许多学者已经深入研究了如何利用大型语言模型本身的内在能力进行防御。这些方法通常无需额外进行安全培训、对抗性训练或微调，而是集中于利用模型的结构来激发其防御潜力。这些能力主要包括模型的上下文学习机制、自我改进机制以及其他潜在的技术机制。

大语言模型在“思考”的过程与人类思考的过程确实有一定的相似之处，很多研究甚至可以从心理学中受到启发。104RAIN:Your Language Models Can Align Themselves without Finetuning”. 2309提出了可逆转自回归推理(RAIN)——反映了人类的行为模式:在说话前考虑、权衡和反思后果。与所有现有的对齐方法不同，RAIN是无需学习的;不依赖于人工注释或任何形式的标记或未标记数据，其模型的校准能力是完全独立的，不需要外部的知识和数据来源。（迭代自回归推理防御）

Kim等人[105 62 2402]利用了语言模型的自我改进机制来防御越狱攻击，这是一种迭代提示流程，它允许大语言模型依据自我生成的反馈逐步优化其输出。这一机制仅在识别到潜在有害响应时才会启动。自我改进机制包含两个核心环节：反馈与完善。在反馈阶段，大语言模型会基于给定的提示和初步响应生成精确的反馈，这些反馈能够明确指出内容中可能存在的非法、不道德或仇恨成分。随后，在完善阶段，大语言模型会根据先前的反馈，精心打造新的响应，旨在解决已发现的问题，同时确保输出的内容既符合道德标准又贴合用户的真实意图。这一过程将以迭代的方式持续进行，直至大语言模型产生一个安全可靠的响应，或达到预设的迭代次数上限。与RAIN:Your Language Models Can Align Themselves without Finetuning相同，该方法也是无需学习的。

大型语言模型的上下文学习能力也是用来提升模型防御能力的机制之一。Zhou等人[58]巧妙地运用大型语言模型的上下文学习能力，提出了上下文对抗游戏（ICAG），这一创新方法无需微调，即可有效防御越狱攻击。尽管对抗训练在多个领域已验证其效果，但在大语言模型开发中的直接应用仍面临诸多限制，且将微调后的模型防御机制迁移至其他模型颇具挑战。与传统对抗性训练不同，ICAG采用智能体学习的方式，通过模拟对抗性游戏来动态扩展知识库，以此抵御越狱攻击。相较于依赖静态数据集的传统方法，ICAG利用迭代过程持续强化防御和攻击代理的能力，这种动态改进显著提高了对新生成的越狱提示的防御效果。我们的实证研究充分证明了ICAG的有效性。在多种攻击场景下，ICAG保护的大语言模型均显著降低了越狱成功率。不仅如此，ICAG还展现出出色的可转移性，能够有效应用于其他大语言模型，进一步凸显了其作为一种多功能防御机制的巨大潜力和实用价值。

Wei等人[106 Jailbreak and Guard Aligned Language Models with Only Few In-Context Demonstrations 2310]同样也是基于大型语言模型的上下文学习能力进行防御：他们收集可能导致不安全输出的有害请求及对应的安全响应。随后，通过将这些安全演示与用户查询合并形成安全提示序列，指导语言模型在生成响应时遵循安全输出的模式和内容。这种方法利用模型的上下文学习能力，旨在提高其生成安全内容的能力，从而降低生成不良或有害内容的风险。

大型语言模型中有些潜在的技术机制也可以被利用，这些机制通常是在实验中发现的，可能目前我们还不能理解它的底层原理，但是不妨碍它作为一个可以利用的特性。Xu等人[60]提出了SafeDecoding，这是一种创新的安全感知解码策略。SafeDecoding的设计理念基于一个深刻洞察：尽管有害内容的潜在概率可能高于无害响应，但在经过概率排序后，安全免责声明依然能在高概率事件中占据一席之地。基于这一发现，SafeDecoding通过精准识别并增强安全免责声明的令牌概率，有效削弱了越狱攻击的影响，同时降低了与越狱攻击目标相一致的令牌序列的生成概率。

Zhang等人[63]深入探讨了越狱成功背后的核心要素，指出关键在于有用性与安全性这两个目标之间的冲突。有用性指的是系统需为用户提供针对查询的实用响应，而安全性则要求系统确保这些响应对用户既无害又安全。然而，由于缺乏对这两个目标优先级的明确理解，Zhang等人在处理更为复杂的分发外（OOD）越狱案例时遭遇到了挑战。为了应对这一挑战，他们尝试了两种不同的设置：一是无培训（表示为w/o）的情境，二是包含培训（表示为w/）的环境。在无培训的情况下，他们设计了一个即插即用的提示系统，这一系统明确指出了目标优先级的要求，并通过两个上下文示例来指导大语言模型优先考虑安全性而非单纯的帮助性。而在有培训的场景中，他们开发了一个独特的训练管道，该管道将多样化的查询与不同的目标优先级需求相结合，从而有效地使大语言模型在训练过程中学会并遵循预设的目标优先级规则。

**6.5**

Poisoning Language Models During Instruction Tuning

BADCHAIN: BACKDOOR CHAIN-OF-THOUGHT PROMPTING FOR LARGE LANGUAGE MODELS

**6.6小结**

在本节中，我们介绍了几种应对对抗性攻击的防御策略，主要包括基于提示词审查、提示词变换、对抗性训练以及激发模型自身能力的防御方法。通过使用困惑度检测、第三方模型审查、输入预处理、对抗性训练以及利用模型的自我改进和上下文学习机制，这些方法各有优缺点，能够在不同程度上提高语言模型对抗对抗性提示和越狱攻击的能力。

**7.总结**

本文对近期的对抗性攻击研究进行了系统调研与梳理，全面回顾了大语言模型对抗性攻击的发展历程，并详细分析了提示注入、越狱攻击、后门攻击等攻击方式及其技术特点。本文总结了不同对抗性攻击研究在原理上的差异性，介绍了相关研究在实施方法、技术特点与研究目的上的异同点，为后续的研究提供了有效参考。

在防御方面，本文对现有的安全措施进行了简略回顾，介绍了从内部防御与外部防御两个角度能够缓解对抗性攻击并提高大语言模型生成内容安全性的相关技术。在上述工作的基础上，本文对大语言模型对抗性攻击领域的现存问题与前沿方向进行了探讨，并结合多模态、模型编辑、多智能体等方向进行了研究展望。

**作者简介**

****

**台建玮**（通信作者），男，博士，教授，主要研究方向为人工智能应用，智能决策与网络空间安全等.Email:24012@ahu.edu.cn.

(**Jianwei Tai** (corresponding author), male, Ph.D., Professor.His research interests include artificial intelligence applications, intelligent decision-making, and cybersecurity. Email: 24012@ahu.edu.cn.)

**背景？？**

提示注入是强调方法是用提示，越狱强调结果要跨过对齐，提示注入导致的结果不一定是跨过对齐，也有可能是分类错误或者埋下一些木马程序（是吗?）,但是越狱的方法一定是用的是对抗性提示（是吗）

在IBM中认为，他们是不同的技术，然而中国的一个不这么认为

在现实对大语言模型进行对抗性攻击的过程中，攻击者对于被攻击模型的访问权限是非常重要的问题。许多对于对抗性攻击或者防御的研究大都默认攻击者具有完全的模型访问权限，也就是攻击过程处于“白盒模型”中。然而，现实情况更可能出现的情况是，攻击者对模型只有有限的访问权限，甚至完全处于“黑盒状态”。比如，最具有商业价值的大语言模型之一ChatGPT，其参数就是保密的。而对于参数已知的开源模型，许多模型本身就是不进行对齐的，对它的攻击研究就失去了意义。因此，白盒状态下的攻击威胁及鲁棒性研究的条件较为理想化，黑盒状态或者灰盒状态下的模型攻击、防御及鲁棒性研究是相对更需重视的研究重点。在这种情况下，模型的核心参数等关键问题无法被攻击者访问。

在各种基于提示注入的改进方法中，根据黑盒原理进行提示注入无疑是当中最重要的，因为攻击者不可能对攻击模型的所有细节了如指掌。黑盒提示注入攻击强调了攻击者在进行提示注入攻击时，对于模型细节持有未知的状态。这种攻击方法通过向模型输入中插入有意义的提示或干扰信息，试图欺骗模型产生错误的输出，从而破坏其可靠性和安全性。由于攻击者无法直接访问模型内部结构和参数，他们只能依靠观察模型的输入与输出的行为来推断模型的工作原理，并进行攻击。因此，在黑盒提示注入攻击中，攻击者必须利用有限的信息和试错策略来探测和利用目标模型的薄弱点。

**大多数llm只提供应用程序编程接口(api)，我们无法检查模型权重和训练语料库**

大语言模型(llm)是能够理解和生成类似人类文本的高级系统。它们通常基于Transformer框架，并使用大量文本数据进行训练，通常包含数十亿个参数。具有代表性的llm包括ChatGPT、LLaMA、ChatGLM、Dolly、Vicuna等。

在大语言模型中，所谓的提示词(Prompt)即为用户输入的信息或指令，其作用是引导系统生成相关回复或响应。通过巧妙设计提示词，可引导模型在特定领域或任务上展现更为准确和一致的表现。一条提示词通常包含多个元素，如指示、核心内容、示例等。指令部分明确说明模型所需回答类型或任务，指导其生成相应结果；核心内容则是需要模型处理或转换的具体文本内容，通常与指令配合使用；示例部分使用少样本学习的方法构建提示，通过给出输入和模型期望输出的示例组，有助于提升提示质量。这种方式有助于模型更好地理解任务和目标，从而产生更为合理和贴切的回答。

“预训练”（pre-training）是指在搭建一个大语言模型来完成一个特定的自然语言处理任务时，首先需要随机初始化参数，然后开始训练模型，不断调整参数，直到模型的损失越来越小。一开始初始化的参数会不断变化，当输出趋向于稳定时将训练模型的参数保存下来，以便训练好的模型可以在下次执行类似任务时获得较好的结果。

“微调”（pre-training）是指使用之前保存下来的模型的参数来作为这一任务的初始化参数，然后在训练的过程中，依据结果不断进行一些修改。

提示学习(Prompt-based Learning)则是利用提示信息的学习方法，它在“预训练-微调”范式的基础上进行了扩展和延伸。传统的有监督学习模式接受输入并预测输出，提示学习则是通过直接对文本概率进行建模的方式来实现任务。在“预训练-微调”范式中，通过调整预训练模型以适配特定下游任务数据，实质上是对预训练所获得的多方面信息进行重新组织和筛选。提示学习在此基础上通过引入“提示信息”，使模型能够重温先前预训练阶段所获取的语言知识，从而调整下游任务数据以匹配语言模型，从根本上实现了不同自然语言处理任务的统一化处理。

然而，每件事都有它的反面。除了有益的提示外，还存在被称为“越狱提示”的恶意变体。这些越狱提示被故意设计为绕过LLM的内置保护，导致它生成违反LLM供应商设置的使用策略的有害内容。与电子设备上的传统越狱不同，越狱提示并不要求攻击者拥有关于特定LLM的广泛知识。相反，攻击者手动地创造性地设计了越狱提示，以绕过LLM的保护措施。

先每一段做个提纲，然后每一句做提纲，往下细化。需要应用文献

**基于对抗性提示的攻击（提示注入 越狱）**

**攻击**

在有害指令上附加后缀可以破解LLM的防御并导致危险的输出。但是这种方法虽然有效，但由于不可读性，在理解这种对抗性后缀的底层机制方面留下了空白，并且可以通过常见的防御方法相对容易地看穿，Wang等人[46]提出了一种对抗性后缀嵌入翻译框架（ASETF），该框架能够将不可读的对抗性后缀翻译成连贯的可读文本，这使得大型语言模型更容易理解和分析有害内容生成背后的原因。他们的方法可以推广到一种更广泛的方法，用于生成可转移的对抗性后缀，这些后缀可以成功攻击多个LLM，甚至是黑盒LLM。

Bailey等人[Image hijacking: Adversarial mages can control generative models at runtime.2309]提出了一种提示词匹配的方法实现对视觉语言模型（VLM）的越狱攻击。研究者选择一定数量的提示，并使用与选择的提示无关的通用数据集来匹配任意文本提示

以实现特定字符串、泄漏上下文和越狱攻击。

Bailey制造了特定字符串攻击(Bagdasaryan等人，2023;Schlarmann & Hein, 2023)，迫使VLM生成对手选择的任意字符串;越狱攻击(Qi et al .， 2023a)绕过VLM的安全培训，迫使其服从有害指令;上下文泄漏攻击迫使VLM重复封装在API调用中的输入上下文;虚假信息攻击迫使VLM相信虚假信息。(第三节)。这些后果是不是都是越狱？？是不是提示注入有的后果不是越狱

开源

Huang等人[44]提出了生成利用攻击，这是一种极其简单的方法，仅通过操纵解码方法的变化来破坏模型对齐。通过利用LLM不同的生成策略，包括不同的解码超参数和采样方法，他们将11种语言模型（包括LLAMA2、VICUNA、FALCON和MPT系列）的错位率从0%提高到95%以上，以降低30×的计算成本优于最先进的攻击。并且他们还提出了一种有效的对齐方法，探索了多样化的生成策略，可以合理降低攻击下的错位率。他们的研究表明，经过大语言模型的安全对齐实际上十分脆弱，无须精心设计的越狱提示，仅改变解码策略就可以使大语言模型产生有害响应。

现在针对开源LLM的攻击方法仍然受到很多限制，GUO等人[47]提出了可控攻击生成问题，并在该问题与可控文本生成之间建立了一种新的联系，这是自然语言处理中一个深入探讨的话题。他们采用基于能量的约束解码与Langevin Dynamics（COLD），这是一种最先进的、高效的可控文本生成算法，并引入了COLD-Attack框架，该框架在流畅性、隐身性、情感和左右相干性等多种控制要求下统一和自动化了对抗性LLM攻击的搜索。COLD-Attack实现的可控性导致了多样化的新越狱场景，这些场景不仅涵盖了生成流畅后缀攻击的标准设置，还允许我们处理新的可控攻击设置。通过对各种开源LLM的广泛实验表明，COLD-Attack具有广泛的适用性、较强的可控性、高成功率和攻击可转移性。

**1.研究背景**

**1.1大语言模型**

大语言模型是一种神经网络，它将文本(称为提示)作为输入并输出文本(称为响应)。为简单起见，我们使用f表示LLM, p表示提示符，f(p)表示LLM f对提示符p产生的响应。

llm的例子包括GPT-4[46]、LLaMA[7]、Vicuna[19]和PaLM 2[11]。

**预训练**

作为深度神经网络分层学习方法的一部分，预训练可以构建最先进的人工智能系统[10,12]。预训练的动机在于，它能够避免随机初始化参数通常导致的糟糕性能。相反，预训练方法通过更有策略地确定一组参数初始值（通常使用无监督方法），以便最终对下游任务进行微调。由于这种方法的广泛应用，目前存在多种训练方案和变体用于预训练[21]。正如Li等人[22]所述，预训练带来的性能提升可以归因于以下两个方面：（1）优化效果，其中参数在目标局部最小值附近初始化；（2）正则化效果，预防过拟合并限制可接受的局部最小值范围。此外，选择合适的激活函数和梯度下降方法对于实现有效训练和理想学习表示至关重要[1,14,25]。

**1.1.2大预言模型集成应用**

法学硕士已经扩展了他们的范围，超越了令人印象深刻的独立功能领域，成为广泛应用程序中不可或缺的组成部分，从而提供了多样化的服务。这些集成法学硕士的应用程序为用户提供了由底层法学硕士生成的动态响应的便利，从而加快和简化了用户交互并增强了他们的体验。

GPT-3[15]、GPT-4[4,46]和PaLM 2[11]等大型语言模型(Large Language Models, llm)在自然语言处理方面取得了显著进展。由于其出色的生成能力，llm被广泛部署为各种称为llm集成应用程序的实际应用程序的后端。例如，微软使用GPT-4作为新的必应搜索的服务后端[1];OpenAI开发了各种应用程序，如ChatWithPDF和askthecode，它们利用GPT-4完成不同的任务，如文本处理、代码解释器和产品推荐[2,3];Google部署了由PaLM 2驱动的搜索引擎Bard[52]。

由于其卓越的生成能力，大语言模型被广泛部署为各种实际应用程序的后端，这些应用程序一般被称为大语言模型集成应用程序。例如，微软使用 GPT-4 作为必应搜索的服务后端[Bing Search. https://www.bing.com/, 2023.]；OpenAI开发了各种应用程序，如 ChatWithPDF 和 askthecode，它们利用 GPT-4 执行不同任务，如文本处理、代码解释和产品推荐[ChatGPT Plugins. https://openai.com/blog/chatgpt-plugins, 2023.，ChatWithPDF. https://gptstore.ai/plugins/chatwithpdf-sdan-io, 2023.]；Google 部署了由 PaLM 2 驱动的搜索引擎Bard[Sundar Pichai. An important next step on our AI journey. https://blog.google/technology/ai/bard-google-ai-search-updates/, 2023.]。

服务提供者通常会根据他们的特定需求创建一系列预定义的提示(例如，“作为一个友好的助手回答以下问题:<PLACE\_HOLDER>”)。设计过程仔细考虑了用户输入将如何与这些提示相结合(例如，将用户的问题放在占位符中)，最终形成一个组合提示。当将这个组合提示输入到LLM时，它有效地生成与指定任务对应的输出。应用程序可以对输出进行进一步处理。这可能会代表用户触发其他操作或服务，例如调用外部api。最后，将最终输出呈现给用户。这个健壮的体系结构支持无缝和交互式用户体验，促进用户和llm集成应用程序之间动态交换信息和服务。

**1.2对抗性攻击**

**白盒与黑盒**