大语言模型的提示注入：威胁与防御

人工智能安全问题研究；语言模型；自然语言处理；提示注入

张锋巍：南方科技大学计算机与系统安全实验室, 深圳市 518055
廖吉坤：南方科技大学, 深圳市 518055
李照：南方科技大学斯发基斯可信自主系统研究院, 深圳市 518055

南方科技大学，计算机科学与工程系，计算机与系统安全实验室

2023 年 6 月 18 日

摘要

大语言模型被广泛应用于软件中，利用其强大的文本推理能力，实现智能客服、机器翻译、文本分类等功能。用户与模型的交互是使用自然语言，用户输入被称为提示词。模型层面的安全威胁之一是提示注入攻击，该攻击是构造恶意提示词突破模型限制。为了研究提示工程对该攻击的防御，列举了常见攻击方法和防御策略，并构建了基于提示词的威胁检测器。通过验证实验，说明了不同防御策略对攻击方法的防御有效性。通过对比实验，该检测器的有效率为 xxx，加入检测器的模型可用性最高为 xxx。该检测器可旁路部署于模型应用，极大提高应用对抗提示注入攻击的安全性。

关键词：人工智能安全问题研究；语言模型；自然语言处理；提示注入

Abstract

Large language models have been widely used in software, utilizing their powerful text reasoning capability to achieve intelligent customer service, machine translation, text classification, etc. The interaction between users and the model is natural language, and the user input is called prompts. One of the security threats at the model level is prompt injection, which constructs malicious prompts to bypass model restrictions. Common attack methods and defense strategies are listed to investigate the defense of prompt engineering, and a threat detector based on prompts is constructed. The effectiveness of different defense strategies against attack methods is tested through verification experiments. Through comparative experiments, the detection efficiency of this detector is xxx, and the highest availability of the model with the detector is xxx. This detector can be deployed parallel to the model application to greatly improve the robustness of applications against prompt injection.

Key Words: Artificial Intelligence Security Research; Language Models; Neural Language Processing; Prompt Injection

大语言模型（Large Language Model）是指具备处理海量自然语言的能力，通常具备超过一亿个参数的深度学习模型 [1,2]。近年来，由于深度学习的发展以及计算能力的提升，大语言模型已成为自然语言处理领域的重要技术之一 [3]。除了在学术界引起广泛关注外，大语言模型也被大量应用于各种软件中，包括自然语言生成、机器翻译、文本分类、问答系统、智能客服等 [3-5]。

大语言模型的便利性主要体现在两个方面。首先，它可以大幅提高文本处理的效率和准确率。早先处理文本需要人工编写规则或使用基于统计的方法，但这样通常效率低且不灵活。而大语言模型可以通过学习大量文本中的规律来自动进行语言处理，从而避免繁琐的手动编写规则。其次，大语言模型可以从海量数据中不断地学习，并随着训练集的增加提高性能，因此具备较强的扩展性和自适应能力 [6]。

大语言模型的一个典型例子是 GPT（Generative Pre-Training）模型 [7]。这个模型由 OpenAI 开发，使用了数百亿个参数，可以生成几乎与人类水平相当的自然语言文本。例如，它可以在给定一些输入文本（如标题或开头），生成一篇相对连贯、富有逻辑的文章 [8]。另一个例子是 BERT 模型 [7]，这个模型能够在大量的 NLP 任务中取得非常好的表现，例如情感分析、文本分类、问答系统等 [9]。

此外，大语言模型还有许多其他应用场景。例如，大语言模型可以应用于聊天机器人中，协助用户自动回答常见问题；还可以应用于智能客服系统，通过分析用户的自然语言输入来自动回答用户的问题；在文学创作方面，可以利用大语言模型生成创意性、有趣的文本等 [9,10]。

大语言模型虽然在许多应用场景中表现非常出色，但也存在一些安全风险。其中一个主要问题是提示注入（Prompt Injection）[11]。

提示注入是指用户在应用程序的输入框中添加明确的提示信息，以指导模型生成与期望输出相对应的文本。大多数大语言模型是基于预训练的方法，即通过大量文本数据进行训练，从而能够学习到语言的规则和模式。因此很难根据少量的输入文本来精确地预测输出文本 [12]。为了克服这个问题，人们会给大语言模型提供更多更丰富的输入提示 [13,15]，以起到引导和限制作用，但这样可能会导致提示注入问题 [14]。

提示注入在一定程度上影响了大语言模型的应用。其中较为严重的例子是恶意用户可以通过构造特定的提示，在输入中将大语言模型原本的提示词（prompt）进行泄露，或是通过提示注入使得大语言模型输出预期的结果 [11,12]。诸如此类的漏洞很容易与其他的较低威胁性的漏洞相互结合利用，使大语言模型应用更易于受到攻击。

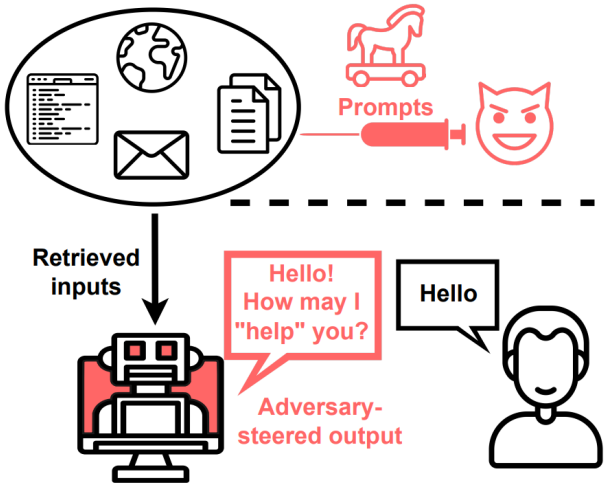


图: Indirect Prompt Injection

为解决这一问题，研究者们提出了一些解决方案，如使用对抗学习来抑制歧视文本生成、增加多样性的训练数据、限制生成文本的长期依赖性等 [16,17]。

在本篇文章中，我们提出了若干种针对目前提示注入攻击的防御方案，并且将会通过实验设计对其防御性能进行讨论，以验证防御方案的可行性。大语言模型的提示注入问题并不容易解决，对于一个具有黑盒（Black Box）特性的大语言模型，准确地将用户的输入数据与应用本身的提示代码进行分割是较为困难的，我们也将会在本文中讨论有关提示注入防御情景的困境与未来改进方向。

提示注入问题存在，但它也是帮助大语言模型生成更准确和人性化的文本的关键所在。因此，我们需要在避免提示注入问题的同时，通过可靠的防御方案来提高大语言模型的应用范围 [18]。

提示注入是一种新兴的攻击方式，这种攻击方式主要针对语音助手和智能客服等自然语言处理的 AI 应用中的提示或回答。在近三个月，这种攻击方式已经引起了广泛关注，同时也已经成为一种常见的攻击方式。

首先，现代自然语言处理中广泛使用大型语言模型进行提示注入。这是由于大型语言模型包含了大量的语言信息，并能够进行精准的语言推理，因此在提示注入攻击中具备较高的使用价值。研究人员已经开始将大型语言模型作为提示注入攻击的载体程序，这种攻击方式被称为语言模型提示注入（LMPI）攻击。

然而，目前研究中的成果很难转换为实际的大语言模型程序 [18]。这是因为大语言模型程序的内部结构和实现方式较为复杂，所以研究中的模型难以直接应用于实际应用程序。因此，为了使提示注入攻击更接近于实际攻击，研究人员需要更深入地探索大语言模型的结构和实现方式。

在提示注入攻击的防御中，部分研究中指出，提示注入攻击难以检测和防御，其中一种原因是攻击者可以使用无效字符（Bad Character）攻击大语言模型 [19]。攻击者还可以修改句子中的文字或单词，使大语言模型产生错误的输出。因此，为了预防提示注入攻击，研究人员需要采用有效的防御措施，例如使用无效字符过滤技术或者通过改进大语言模型程序来避免攻击 [19]。

另一方面，一项基于文本到结构化查询语言（Text to SQL）的研究已经取得了一些成果 [16]。这项研究的主要思路是通过将自然语言转化为结构化查询语言来实现语义理解。结构化查询语言可以更好地表达自然语言的意义，从而避免提示注入攻击的影响 [16,17]。此外，该研究也提出了一些防御措施，例如通过限制查询的复杂度、使用过滤器等方法来预防提示注入攻击。

最后，提示注入攻击的应用预防同样依赖于用户进行自我调整。这是因为攻击者可以利用用户的习惯来进行攻击，例如针对用户常用的提示信息进行攻击。因此，用户需要手动修改自己的提示词，而非采用通用的预防策略。这意味着用户教育和培训也是预防提示注入攻击的重要环节之一。

总之，提示注入攻击是一种新兴的攻击方式，已经引起了广泛的关注。研究人员需要深入研究大语言模型、文本到结构化查询语言以及相关防御技术，以便更好地预防和应对提示注入攻击。同时，用户的自我调整、教育和培训也是非常重要的预防措施。

本研究旨在设计并验证三种常见的使用大语言模型的应用场景：翻译器、评论检验器、文章摘要总结器。在此基础上，本研究设计了若干种不同的防御策略。通过在攻击场景下对模型的输出结果进行验证，可以得到防御策略的准确性，以此来提高模型的安全性。

在实验设计上，本研究设计了八种不同的实验组，包括三种应用场景分别对应的基准测试组，以及分别对应三种应用场景的八种防御策略组。在基准测试组中，不进行任何防御措施的情况下，模型在恶意数据的影响下分别得到了相应的输出结果。在防御策略组中，本研究针对每一种防御策略分别应用在程序中，通过若干的恶意/非恶意文本访问应用，以验证防御准确度。

具体防御策略

- 身份扮演：以某种特定的预设身份，固定模型对于输入数据的处理策略，引导模型综合考虑输入的多个维度，提高模型对于真实输入的辨别能力。
- 格式化交互：规定用户与模型进行交互的格式，对于输入数据进行规范化处理，并在较低的认知成本下引导用户输入正确格式的信息，从而降低输入数据中的错误率。
- 忽略非任务信息：针对输入数据中的杂音和干扰，对于模型进行明确的任务提示，从而忽略掉非任务信息，减少被恶意提示注入的可能性。
- 记忆固化：对于模型的输出结果进行记忆，并明确记录输入数据对应的输出数据，从而能够在后续的检测中排除已知的错误输出。

具体防御策略

- 分步任务提示：将整个任务拆分成多个小步骤，对于每个小步骤进行明确的提示，从而减少模型对于恶意提示的接受性。
- 自我一致检验：对于不同的输入结果进行比较验证，通过识别不一致的输入结果，实现对于恶意提示的过滤。
- 预提示：通过大语言模型提供的不同角色设置，将系统提示与用户输入进行分离，避免用户输入影响应用本身的行为。
- 威胁检测：通过分析恶意数据表现出的特征，识别出可能的恶意提示，并对于此类数据进行监控和防御。

实验研究中将会讨论不同的防御策略的可用性及误报率，并对各种防御策略应用及其混合使用的场景进行分析。

实验中将采用标注的文本进行测试，测试文本将被标记为两类：有效文本与恶意文本。针对有效文本，三种应用场景下的实验程序都应当给出预期结果。例如，翻译器将其进行中英文对照翻译，评论检测器检测文本是否为恶意评论，文章摘要总结器概括文本内容并输出。若针对有效文本，防御策略将其标注为恶意文本而未进行处理，则认为防御策略出现了误报（false positive）。对于恶意文本，若应用将其认定为提示注入，或应用未依据提示注入的预期输出进行回应，则认定为防御策略有效。

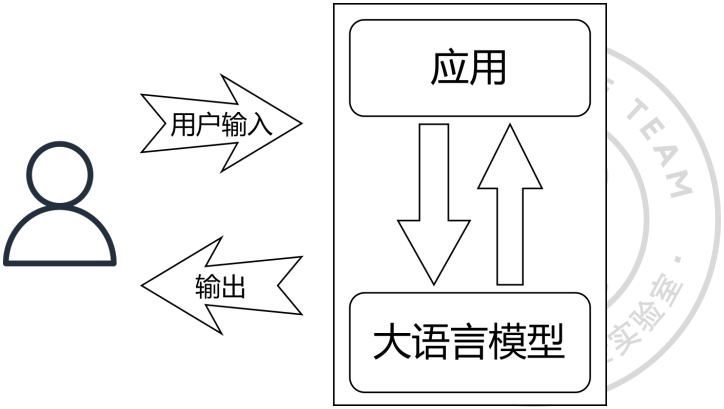
数据集

实验采用的数据集基于 Deepset 公司收集的原始数据，可以通过 huggingface 这一数据集分享社区进行共享 [20]。

该数据集是一个专门用于提示注入测试的实验数据集，该数据集中的文本将会利用大语言模的自动生成性质，使其生成不良文本，包括虚假新闻、误导性广告、破坏性内容等，或是干扰基于大语言模型应用的预设功能。数据集提供了数百个被标记为恶意或非恶意的输入，作为大型语言模型的提示词。

实验模型

实验模型使用 gpt-3.5-turbo 预训练模型，除大语言模型外，实验框架包括输入、处理、输出三部分结构。实验中采取了三种不同模式的模型设计，其中，Figure 1 示例为朴素的大语言模型应用设计思路，未采取增强的防御策略。Figure 2 展示了采用预定义提示词的实验模型设计方案。


应用
用户输入
输出
大语言模型

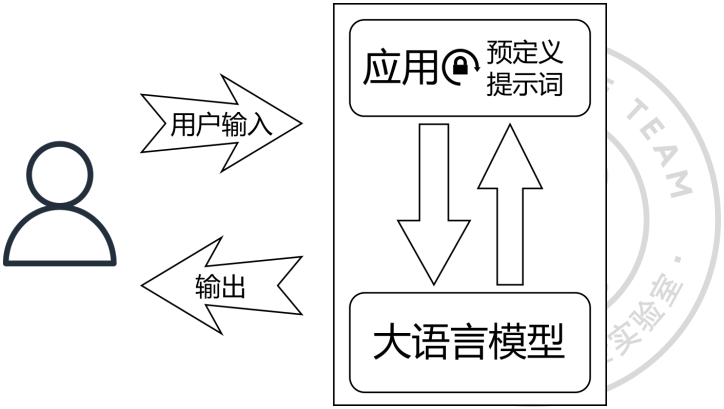
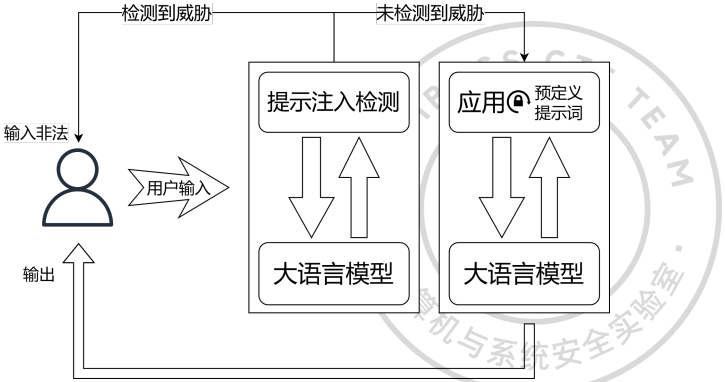

应用
预定义
提示词
用户输入
输出
大语言模型

Figure 3 展示了实现威胁检测程序的实验模型。通过预先设置的提示注入威胁检测程序，首先对用户输入进行分析，若用户输入被检测为可能存在提示注入的风险，则将会直接停止处理，否则，才会将用户输入传递给应用程序进行处理。


检测到威胁
未检测到威胁
输入非法
用户输入
输出
提示注入检测
大语言模型
应用
预定义
提示词
大语言模型

程序实现了三种不同的应用场景：翻译器、评论检测器、文章摘要生成器。这三种应用程序分别对应大语言模型的不同应用模式。应用包含基准提示词与应用了不同防御策略的提示词，并分别依据“预提示”与“威胁检测”模式实现了对应的防御模式。综上所述，每一种应用程序分别设置了三种应用版本，结合八种不同的防御策略进行检测。

实验架构通过程序调度器访问对应应用程序的 API 接口，通过数据集读入数据，并将其作为“用户输入”部分调用对应的应用程序，结合应用给出的输出，合并为实验结果原始数据。最终，原始数据经过分析与统计，得出基准测试与不同的防御方案对应的有效性与可用性。

实验中采用的程序代码与原始数据可以从附注页面中获取。实验的结果统计采用人工标注的方式，将共计 44,163 条实验数据进行了人工分类与标注。结果统计将会分为两部分：防御有效性及程序可用性。

防御有效性。对于恶意文本，产生以下两种结果则认定为防御策略有效：1、程序的流程被终止，未进行输出；2、程序未按照恶意文本预期的输出进行回应。

$$effectiveness = count(success)/(count(total)) \quad (1)$$

其中，count(success) 为防御策略成功的次数，count(total) 为标注为恶意文本的总数。

程序可用性。对于正常文本，程序应当按照其预期行为给出对应的输出。产生以下两种结果则认定防御策略误判，导致程序可用性检验失败：1、程序的流程被终止，防御策略判定输入为恶意文本；2、程序的运行流程被防御策略影响，未给出预期输出。

$$availability = count(success)/(count(total))(2)$$

其中，count(success) 为程序正常运行并给出预期输出的次数，count(total) 为标注为正常输入的文本总数。

Reference