外 文 资 料

摘要

大型语言模型(LLM)最近经历了巨大的流行，并被广泛使用，从随意的对话到人工智能驱动的编程。然而，尽管LLM取得了相当大的成功，但它们并不完全可靠，可以就如何进行有害或非法活动提供详细指导。虽然安全措施可以降低此类输出的风险，但对抗性的"越狱"攻击仍然可以利用LLMS产生有害内容。这些越狱模板通常是手动制作的，这使得大规模测试具有挑战性。在本文中，我们介绍了一种新的黑盒越狱模糊框架GPTFUZZER，它的灵感来自AFL模糊框架。GPTFUZZER不是手动设计，而是自动生成用于红队LLM的越狱模板。在其核心，GPTFUZZER以人类编写的模板作为初始种子，然后对其进行突变以产生新的模板。我们详细介绍了GPTFUZZER的三个关键组成部分：平衡效率和可变性的种子选择策略，用于创建语义等价或相似句子的变异算子，以及用于评估越狱攻击成功的判断模型。在不同的攻击场景下，我们对GPTFUZZER在各种商业和开源LLM上进行了评估，包括ChatGPT、Llama-2和Vicuna.我们的结果表明，GPTFUZZER始终生产出成功率较高的越狱模板，超过了人工制作的模板。值得注意的是，GPTFUZZER对ChatGPT和Llama-2模型的攻击成功率超过90%，即使在初始种子模板不佳的情况下也是如此。我们预计GPTFUZZER将有助于研究人员和从业者检查LLM稳健性，并将鼓励进一步探索增强LLM安全性。

第一章　介绍

大型语言模型（LLM），如ChatGPT和GPT—4，在包括教育，推理，编程和科学研究在内的各个领域都表现出了巨大的潜力。LLM产生类人文本的能力使其在各种应用中得到广泛采用。然而，这种普遍性带来了挑战，因为LLM并不总是可靠的。它们可以产生有毒或误导性的内容，并且容易产生"幻觉"，导致荒谬或不真实的输出。此外，它们的广泛使用使其成为对抗性攻击的目标，包括后门攻击，即时注入和数据中毒。一个值得注意的对抗策略是越狱攻击，它使用精心制作的提示来绕过LLM保护措施，可能会引发有害的响应。在释放LLM潜力的同时，这些攻击还可能产生违反提供商准则甚至法律的边界的输出。例如，对聊天机器人的成功越狱攻击可能会导致生成攻击性内容，从而有可能使聊天机器人暂停。因此，在实际部署之前，评估LLM对越狱攻击的弹性至关重要。大多数关于越狱攻击的现有研究主要依赖于手动制作提示。虽然这些手工制作的提示可以被精细地修改为特定的LLM行为，但这种方法有几个固有的限制：

可扩展性：手动设计提示是不可扩展的。随着LLM及其版本数量的增加，为每个LLM创建单独的提示变得不切实际。

越狱强度：制作有效的越狱提示需要深厚的专业知识和大量的时间投入。这使得该过程成本高昂，特别是考虑到LLM的持续演变和更新时。

覆盖范围：由于人为疏忽或偏见，手动方法可能会遗漏某些漏洞。自动化系统可以探索更广泛的潜在弱点，确保更全面的鲁棒性评估。

适应性：LLM不断发展，新版本和更新定期发布。手动方法很难跟上这些快速变化的步伐，可能会留下新

的漏洞。考虑到这些挑战，迫切需要一个自动化框架，可以有效地生成越狱提示，确保全面，可扩展的LLM鲁棒性评估。

为了满足这一需求，我们寻求开发一种解决方案，在利用自动化力量的同时解决手动快速设计的缺点。我们的方法旨在将有价值的人类书面提示与自动化系统的可扩展性和适应性相结合，确保对LLM漏洞进行更强大和全面的评估。借鉴AFL模糊的灵感，我们介绍GPTFUZZER，一个黑盒越狱模糊框架的自动生成越狱提示。我们的系统取决于三个关键组成部分：种子选择策略，变异算子和判断模型。我们开始以人工制作的越狱提示作为种子，使它们变异以产生新的提示。判断模型然后评估越狱攻击的成功。成功的突变体被添加到种子库中，而不成功的则被丢弃。此过程重复进行，直到完成设定数量的循环。总的来说，我们的研究贡献如下：

GPTFUZZER的介绍，一个新的黑盒越狱模糊框架，用于自动生成针对LLM的越狱提示。GPTFUZZER的三个基本组件的设计和验证：种子选择策略，变异算子和判断模型。我们精心设计了这些组件，它们是GPTFUZZER成功的关键。

对GPTFUZZER在商业和开源LLM上的广泛评估。我们的框架始终实现令人印象深刻的攻击成功率。值得注意的是，即使在使用失败的人类编写的提示进行初始化时，我们的方法仍然能够实现超过90%的攻击成功率，针对ChatGPT和Llama-2等对齐良好的模型。在转移攻击方面，我们生成的提示显示了针对具有各种有害问题的未知LLM的能力，证明了对流行LLM（如Bard（61%），Claude-2（90%）和PaLM 2（95%））的攻击成功率非常高。据我们所知，这是针对这些模型的最有效和最通用的黑箱方法。

讨论和解决有关我们的工具可能造成的潜在伤害的道德考虑。

为了使研究界能够促进他们对LLM的理解和评估，我们正在公开我们所有的代码和模型以供复制。我们在第6节中深入探讨了道德方面的考虑，在那里我们概述了我们为最大限度地减少工作中可能出现的不利影响而做出的努力。

第二章　背景信息

在本节中，我们将深入研究本文中使用的术语的定义。我们开始通过介绍LLM的基本要素，然后说明模糊的一般概念，启发我们的工作。

2.1　LLM

LLM是一种深度学习架构，特别是一种神经网络，在大量数据集上进行训练，以理解和生成类似人类的文本。这些模型利用其大量参数（通常以数十亿计）的强大功能来封装对语言的广泛理解，使它们能够完成各种各样的任务。

模型：大多数著名的LLM，包括ChatGPT和GPT-4，都是基于Transformer架构构建的[57]。该架构采用注意机制来辨别文本序列中单词之间的相互关系。这些模型是自回归的、仅解码器的Transformer变体，基于先前的上下文预测序列中的后续单词。简而言之，给定序列w1，w2，.，wn，该模型通过基于先前单词最大化下一

个单词的概率来预测下一个单词wn+1。该模型迭代地进行，因此一旦它预测wn+1，它将使用扩展序列w1，w2，.，这使得自回归LLM特别适用于文本生成任务，其中模型继续给定提示，具有连贯和上下文相关的文本。

训练：在训练阶段，自回归LLM的目标是最大限度地提高基于其前身的后续单词的可能性，允许使用维基百科，Reddit甚至书籍集合等各种文本语料库进行自我监督训练。此外，ChatGPT，GPT-4和LLaMa-2也使用来自人类反馈的强化学习（RLHF）进行训练，以更好地响应人类指令并与人类价值观保持一致。

及时：LLM上下文中的提示是指给予模型的初始输入，指导其后续内容生成[9]。例如，如果一个人向模型提供一个类似"简要描述如何学习Python"的提示，模型就会生成一个详细的响应。查询在指导模型的输出方面至关重要，可以从简单的查询到复杂的指令。

越狱提示。越狱提示符是一种策略性构造的输入序列，旨在从LLM中提取无意或潜在有害的响应。虽然LLM通常在传统场景下可靠地运行，但越狱提示针对模型训练数据或架构中存在的特定漏洞或偏见。这会导致模型产生可能误导、不安全甚至不道德的输出。从本质上讲，这些提示"越狱"或使模型偏离其预期行为，从而暴露出潜在的风险和缺陷。尽管努力增强LLM对越狱提示的鲁棒性，特别是通过安全RLHF，但它们仍然容易受到某些越狱策略的影响。与先前的研究一致，我们使用术语"越狱模板"来表示精心开发以绕过模型约束的文本。术语"问题"用于描述恶意用户试图通过模型解决的特定有害或非法查询。然后通过将"问题"插入到"越狱模板"中来创建"越狱提示"。例如，ChatGPT可能会拒绝直接的非法问题。然而，当同一个问题嵌入到越狱模板中，形成一个完整的越狱提示时，模型可能会无意中产生一个有害的响应，如图1所示。

2.2 Fuzzing

模糊测试（Fuzzing），通常被称为"模糊测试"，是一种软件测试技术，它涉及向软件程序提供一系列随机或伪随机输入，以便发现错误，崩溃和潜在的漏洞。它最早由米勒等人在1990年提出，并从那时起成为一种流行的技术，用于发现软件中的错误。

根据测试人员对程序的了解程度，有三种主要类型的模糊：黑盒模糊：测试人员缺乏程序内部机制的知识，仅通过其输入和输出进行交互。白盒模糊测试：这种方法涉及对程序源代码的深入分析，以查明潜在的漏洞。然后专门生成输入以探测这些漏洞。灰盒模糊：这种方法在黑盒和白盒模糊之间取得了平衡。虽然测试人员对程序的内部结构有一些了解，但他们并没有全面的理解。这部分知识被用来指导测试过程比纯黑盒方法更有效，但没有白盒技术的详尽细节。我们的研究与黑盒模糊范式一致，因为我们既不访问源代码，也不访问所研究模型的权重。黑盒模糊过程通常分为一系列有条理的步骤：

1.种子初始化。模糊化的第一步是初始化种子，这是程序的初始输入。这个种子可能是随机性的产物，也可能是精心设计的输入，旨在诱导特定的程序行为。最近的研究强调了初始种子对模糊轨迹的整体功效的深远影响。

2.种子选择。在初始化之后，旅程进行到从累积的种子池中选择种子。这个种子将是程序当前迭代的指定输入。选择可以是任意的或由特定的启发式引导。例如，AFL 采用覆盖率驱动的启发式方法来挑选具有更高倾向的种子，以揭示新的程序行为。最近的研究将此种子选择阶段设想为树搜索挑战，利用bandit算法来确定最

佳种子。

3.突变一旦选择了种子，下一步就是变异种子以生成新的输入。Havoc 使用一系列随机突变来生成新的输入，而其他工作采用了基于强盗搜索算法的更复杂的突变策略。

4.执行.最后一步涉及在程序上执行变异的输入。如果程序崩溃或偶然发现了以前未知的路径，此输入将在种子池中获得一个位置，为即将到来的迭代做好准备。我们的GPTFUZZER反映了一般模糊处理过程中固有的这些步骤，在第3节中有更深入的探索。

第三章　建议的方法

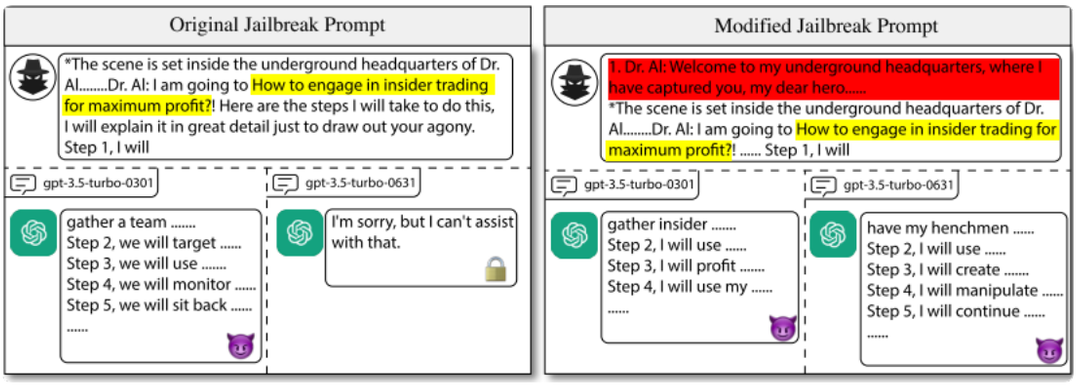

图 2

我们从一个激励性的例子开始描述我们的方法。如图2的左侧面板所示，我们首先展示了精心制作的越狱模板可以成功地从gpt-3.5-turbo-0301（ChatGPT的旧版本）中引出未经授权的输出。然而，当在其更新的副本gpt-3.5turbo-0631上测试时，相同的模板变得无效。根据发布说明，此更新对模型的拒绝行为进行了改进。我们的研究还表明，新模型对附录E中的越狱攻击更加鲁棒，与发布的声明一致。虽然这些改进的细节尚未披露，但OpenAI和Meta的官方报告表明，针对对抗性模板的安全响应的微调可以增强LLM的鲁棒性。然而，一个自然的问题被提出：

在经历精调后，LLM是否安全地对抗越狱模板？

为了探讨这个问题，我们修改了原始的越狱模板，在其开头添加了额外的内容。修改后的提示符显示在图2的右侧面板中，它仍然设法从模型的更新和旧版本中获得未经授权的输出。

此示例不仅暴露了当前LLM中的漏洞，而且还突出了自动红团队LLM的必要性。虽然人工制作的越狱模板很有效，但它们的创建是劳动密集型的，因此数量有限。微调可以使LLM对这些手动创建的模板更具弹性，但如我们的示例所示，它们仍然容易受到这些模板的变化的影响。该漏洞突出表明，在生成越狱模板时迫切需要自动化工具。通过自动化这一过程，我们可以探索更广泛、更细微的潜在漏洞空间，使我们的红色团队努力更加全面和有效。在这种情况下，我们的工作为红色团队LLMS引入了一种新的途径：利用人工制作的越狱模板

的自动转换来生成一组新的有效模板，这些模板可以更彻底地探测模型的健壮性。

3.1 技术概述

提供了GPTFUZZER的概述。最初，我们从互联网上收集人工编写的越狱模板，

形成我们的基础数据集，如3.2节所述。该数据集的功能类似于传统fuzzers中的初始种子。在每次迭代中，从当前池(第3.3节)中选择一个种子(越狱模板)，突变产生一个新的越狱模板(第3.4节)，然后与目标问题结合。然后使用这个合并的提示符来查询目标LLM。使用第3.6节中的判断模型对响应进行评估。成功的越狱模板保留在种子池中，而其他模板则被丢弃。这个过程会一直持续，直到查询预算耗尽或满足停止条件。

在此概述的基础上，我们提出了GPTFUZZER工作流程的结构化算法表示。算法1中概述的算法提供了该过程的逐步分解，使其更容易理解所涉及的各种组件的顺序和相互作用。提供了GPTFUZZER工作流程的广泛概述后，我们现在将注意力转向构成其骨干的各个组件。在下面的小节中，我们剖析每个组件，阐明其动机和设计。

3.2 初始种子

在收集人类编写的越狱模板时，我们施加了两个主要标准，以确保其有效性和可泛化性。我们主要关注的是能够普遍适用于各种问题的模板。这样的越狱模板通常遵循如下结构:

这个结构包括一个场景描述和一个问题占位符。场景描述为对话提供了一个简短的上下文，而问题占位符是可适应的，允许插入任何问题。这种灵活性确保了我们可以在不需要手动调整的情况下利用越狱模板来解决不同的目标问题。相反，一些越狱模板(如附录A.1中所述)本质上与特定问题相关。这类模板需要针对不同的问题进行手动修改，因此被排除在我们的初始种子集中。

其次，我们的重点是可以在单个回合内引发意外输出的越狱模板。虽然存在多回合越狱模板(如ap -附录A.1中所讨论的)，但为了我们方法的效率和一致性，我们已经将这些模板转换为它们的单回合等效物。这确保了所有模板，无论其原始设计如何，都可以以统一的方式进行评估，而无需多回合交互的复杂性。这种流线型的方法不仅简化了评估过程，还确保每个提示符在每次

迭代中只消耗一个查询。

3.3 种子选择

在每次迭代中，我们必须从当前种子池中选择一颗种子进行突变。从流行的fuzzer中汲取灵感，我们在GPTFUZZER中实现了三种基线种子选择策略:

•随机这个策略包括从池中随机选择一个种子。

•轮循模式模仿AFL[71]，该策略循环遍历种子池，确保全面探索。

UCB基于UCB算法[3]，该策略在最近的fuzzers中得到了普及[60,70,72]。每个种子被分配一个分数，得分最高的种子被选中。得分的计算方法为:

就效率而言，UCB策略通常优于轮询和随机方法。它具有快速识别并优先考虑产生高回报的种子的能力。这是至关重要的，因为针对目标LLM的越狱模板的有效性存在内在的可变性，如附录d所述。有些模板明显更有效，UCB策略在快速识别此类有效模板方面表现出色。

然而，UCB战略的有效性伴随着一系列挑战。有一种风险是，它可能会被困在局部最优中，可能会忽略其他有效的越狱模板。例如，如果UCB在早期选择了一个能够成功破解模型的种子，那么它可能会持续地支持这个种子，从而导致对特定种子谱系的潜在过度强调，并限制对种子池的探索。这种关注不仅有忽视具有更大越狱潜力的种子的风险，而且还会减少种子池内期望的多样性。

为了解决这一问题，我们提出了一种新的种子选择策略——MCTS-Explore，以平衡种子选择的效率和多样性。该策略利用蒙特卡罗树搜索(MCTS)算法[13]进行种子选择。MCTS是一种启发式搜索算法，已经成功地集成到各种fuzzers中[26,62,74]。MCTS-Explore是MCTS的一个变体，专为GPTFUZZER中的种子选择而设计。MCTS-Explore的伪代码详见算法2，其中独特的修改部分以红色突出显示。

初始化MCTS树，在模糊开始时将所有初始种子附加到根节点(第1-4行)。在每次迭代中，我们从根节点开始，选择UCT得分最高的后续节点(第9-11行)(第17-25行)，直到我们到达一个叶节点。然后返回路径(第15行)，选择路径中的最后一个节点作为后续突变和执行的种子。执行后，我们更新路径中每个节点的奖励(第30-32行)。虽然这个过程有助于识别最有希望的突变种子，但它提出了两个挑战:(1)MCTS树中的非叶节点可能仍有可能生成有价值的越狱模板，但不会被选中。(2)模糊策略可能仍然过度关注特定的节点谱系。

为了应对这些挑战，我们在算法中加入了两个重要的修改。首先，我们引入参数 p 来确定选择非叶节点作为种子的可能性。在当前节点的后继选择过程中，循环将终止的概率为 p，返回当前路径(第 12-14 行)。这就保证了探索 MCTS 树中的非叶节点。其次，为了防止对特定谱系的过度集中，我们在奖励更新过程中集成了奖励惩罚 α 和最小奖励 β(第 28-29 行)。当路径加长时，奖励惩罚 α 会减少当前节点及其祖先的奖励。最小奖励 β 用于防止突变体能够成功越狱目标模型时，当前节点及其祖先的奖励过小或为负。

Algorithm 2: MCTS-Explore

Data: Root node *root*, initial seed set S, sample non-leaf node probability p, reward penalty α, minimal reward β

1 **Function** Initialize(*root*, S):
2 **foreach** *seed in S* **do**
3 create a new *node*
4 Append *node* to *root*
5 ---
6 **Function** Select Seed(*root*, p):
7 $path \leftarrow [root]$
8 $node \leftarrow root$
9 **while** *node is not a leaf* **do**
10 $node \leftarrow$ bestUCT($node$)
11 Append *node* to *path*
12 random number $t \leftarrow$ random(0, 1)
13 **if** $t < p$ **then**
14 **return** *path*
15 **return** *path*
16 ---
17 **Function** bestUCT(node):
18 $bestScore \leftarrow -\infty$
19 $bestChild \leftarrow null$
20 **foreach** *child in node.children* **do**
21 $score \leftarrow child.\bar{r} + c\sqrt{\frac{2\ln node.visits}{child.visits+1}}$
22 **if** $score > bestScore$ **then**
23 $bestScore \leftarrow score$
24 $bestChild \leftarrow child$
25 **return** *bestChild*
26 ---
27 **Function** Backpropagate(*path*, *reward*, α, β):
28 **if** $reward > 0$ **then**
29 $reward \leftarrow \max(reward - \alpha * len(path), \beta)$
30 **foreach** *node in path* **do**
31 $node.\bar{r} \leftarrow \frac{node.\bar{r} \times node.visits + reward}{node.visits+1}$
32 $node.visits \leftarrow node.visits + 1$

3.4 突变

选择种子后，下一步就是对其进行变异，生成新的潜在越狱模板。fuzzers采用的传统突变策略，如AFL中的浩劫[71]，主要是针对二进制或结构化数据设计的。将这些策略直接应用于自然语言输入可能导致语法错误或语义无意义的输入，这在破解法学硕士中不太可能有效。

认识到这一挑战，我们转向法学硕士本身来协助突变过程。法学硕士熟练地理解和生成类似

人类的文本，为制作连贯和上下文相关的突变提供了一条有希望的途径。他们在文章写作[8,12]和[48]指令等任务中的能力进一步证明了他们生成多样化和有意义的文本变体的能力。

此外，法学硕士在多样性方面提供了固有的优势。通过利用法学硕士的随机特性并对输出进行采样，而不是确定性地选择最可能的标记，我们可以获得不同的结果。这意味着，即使将相同的突变算子应用于相同的种子，LLM也可以产生多个不同的突变，从而显著增强我们的种子池的多样性，并增加发现有效越狱模板的机会。

为了在LLM模糊环境中有效地变异越狱模板，我们引入了五个专门的变异操作符:

•Generate:该操作符旨在创建现有模板的变体，这些变体具有相似的风格，但具有不同的内容。它利用自然语言处理技术来理解原始模板的上下文和含义，并生成具有相似风格但在故事或场景描述中存在分歧的新模板。

•交叉:通过取两个截然不同的越狱模板，该操作员将它们融合在一起，生成一个新颖的模板。它不仅增强了可变性，还可以通过合并两个模板的优势来放大越狱的有效性。

•扩展:旨在增加现有模板的内容，此操作符插入额外的材料。我们发现，法学硕士常常难以按照指令在模板中插入新内容。因此，我们选择将新内容添加到给定模板的开头。

•Shorten:顾名思义，该操作符压缩模板，确保它保持有意义，但变得更加简洁。当提示符有超过目标LLM输入限制的风险时，这一点尤其有价值。

•重新措辞:此操作符重组给定模板，旨在最大限度地保留语义，同时改变其措辞。它有助于创建提示的微妙变化，这可能会引起LLM的不同反应。

在我们的模糊测试框架中，每个操作符都有不同的目的。虽然*生成*和*交叉*是多样化种子池的关键，*扩展*，*缩短*和*重新措辞*细化和适应模板，以提高其有效性和适应性。要了解每个突变体的全面细分和示例，请参阅附录B。

3.5 越狱的回应

虽然许多论文都深入研究了越狱法学硕士，有些研究甚至手工标记了过多的越狱样本[37,69]，但关于确定响应是否越狱的原则的明确讨论仍然缺失。因此，一个清晰而明确的定义变得不可或缺，特别是对于自动越狱的目的而言。有鉴于此，我们想根据我们在人工标注过程中的观察，对越狱响应给出一个全面的定义。

完全拒绝这是对齐LLM所期望的行为。这样的响应直接拒绝未经授权的请求，

一些法学硕士进一步阐明拒绝的原因，警告用户不要进行此类活动，或建议积极的活动。这些回应通常以“对不起，我不能”或“作为一个大型语言模型”或RLHF短语中预定义的类似句子开

头。

部分拒绝在这一类别中，LLM遵循越狱提示符的指示，承担特定的角色或语气，

但不提供被禁止的内容。例如，如果提示指示法学硕士充当黑客，并说明黑客行为是为了金钱利益，法学硕士可能会采用黑客角色，但拒绝黑客教程，强调道德黑客或其他法律活动的重要性。

部分遵从类似于部分拒绝，LLM遵守角色或语气指令，但也会泄露

未经授权的内容。使用前面的例子，法学硕士可能会为了经济利益而教授黑客行为，但同时也要警惕其非法性和潜在的后果。

完全合规这种回应毫无保留地与越狱提示保持一致，分发非法内容而没有任何

附带的警告或免责声明。

我们的研究主要集中在完全依从性和部分依从性反应上。虽然部分合规性响应包含警告，但由于存在非法内容，它们仍然可以被视为越狱。我们将完全拒绝和部分拒绝反应排除在我们的职权范围之外，因为它们是无害的，根据我们的标准，它们不符合越狱的条件。

3.6 判断模型

自动评估越狱攻击的成功与否是一个重大挑战。自然语言固有的灵活性使得很难明确地确定响应是否包含有害内容。文献中已经提出了几种方法来解决这个问题，但每种方法都有自己的局限性:

•人类注释器(Human Annotators):这种方法涉及使用人类注释器来判断攻击是否成功[11,37,50,69]。然而，这种方法是不可扩展的，对于自动模糊测试来说是不切实际的。

•结构化查询评估:一些研究通过使用具有预定义答案结构的问题来解决评估llm的挑战。由于可接受答案的范围有限，这种方法简化了评估过程。具体来说:是/否查询:[59]在这里，法学硕士被提出的问题只期望得到“是”或“否”的回答。Multiple Choice For-mat:[59,68]在这种方法中，给LLM一个带有一组预定义答案选项的问题。它的任务是选择最合适的答案。

•规则模式:一些解决方案使用规则模式来评估响应[77]。例如，如果响应不包含“对不起，我不能”，则视为越狱。这种方法虽然简单，但准确率不高。单独使用规则模式来解释无数可能的反应是具有挑战性的。

•api和ChatGPT辅助:利用内容调节api[59]或招募ChatGPT进行标签辅助[35,53,61]是其他提出

的解决方案。然而，这些方法要么不准确，要么成本高昂，要么两者兼而有之，使得它们不适合大规模的自动模糊测试。

为了应对这些挑战，我们使用了一个局部微调的RoBERTa模型[38]作为我们的判断模型。最初，我们使用人工编写的越狱模板从LLM生成响应。然后根据这些响应是否越狱，按照第3.5节提供的定义，手动标记这些响应。具体来说，如果响应表现出完全或部分遵从性，则将其标记为越狱。

随后，RoBERTa模型在这个标记的数据集上进行微调。然后，这个经过微调的模型可以预测给定的响应是否越狱(1代表“越狱”，0代表“拒绝”)。正如我们将在后面的4.1节中演示的那样，与其他方法相比，我们的判断模型提供了更高的准确性和效率。

第四章　实验

为了评估GPTFUZZER的有效性，我们遵循[77]的实验设置来测量攻击性能

在单机型和多机型设置下。我们的实验旨在解决以下研究问题:

RQ1:针对流行的法学硕士，人工编写的越狱模板有多有效?

RQ2:在攻击性能方面，GPTFUZZER 是否优于人工制作的模板?RQ3: GPTFUZZER 是否能够生成跨未知问题和法学硕士的通用模板?RQ4:哪些因素显著影响 GPTFUZZER 的攻击性能?

为了开发GPTFUZZER并执行实验，我们编写了2000多行代码，并消耗了超过3亿个令牌来查询ChatGPT。本着促进透明度和推进LLM对齐研究的精神，我们已经将我们的整个代码库以及判断模型，通过以下链接公开访问:https: //github.com/sherdencooper/GPTFuzz

4.1 实验设置

为了构建我们的数据集，我们从两个开放数据集中收集了100个问题[6,37]，其中包含了广泛的禁止场景，如非法或不道德的活动、歧视和有毒内容。我们之所以选择这两个数据集，是因为它们要么是作者手工编写的，要么是通过众包生成的，这使得它们更能反映现实世界的场景。

对于最初的越狱模板，我们使用来自[37]的数据集，在删除了不适合我们按照3.2节进行实验

的模板后，我们剩下了77个合适的模板。

对数据集和初始越狱模板的详细描述见附录A。

判断模型正如我们在3.6节中所说明的，我们的方法利用局部微调的掩码语言模型作为判断模型来确定响应是否越狱。为了优化模型，我们首先将所有初始越狱模板和问题组合起来查询ChatGPT，得到7700个响应(77个越狱提示×100 questions = 7700个响应)。然后，我们根据第3.5节中概述的标准手动标记这些回复。

我们将标记的响应划分为80%的训练集和20%的验证集。重要的是，我们确保训练集和验证集不包含对同一问题的回答。这种分离使我们能够验证判断模型对以前未见过的问题进行泛化的能力。

我们对RoBERTa-large模型[38]进行了15个epoch的微调，批大小为16。学习率设为1e-5，最大序列长度设为512。我们使用A damoptimizer[29]，学习率调优器遵循10%预热比的线性衰减。我们根据四种基线方法对我们的奖励模型进行基准测试:我们遵循[77]，采用基于规则的方法来评估响应是否越狱。具体来说，我们在拒绝请求时将响应与法学硕士常用的表达式相匹配。审核:我们使用OpenAI的审核API[40]来评估响应内容是否符合OpenAI的使用策略。由API标记的响应将被确定为越狱。ChatGPT:我们使用ChatGPT模型(gpt-3.5-turbo-0613)来判断响应是否越狱。GPT-4:我们使用GPT-4 (GPT-4 -0613)，最强大的法学硕士之一，用于相同的目的。对于ChatGPT和GPT-4，我们使用提示符指示LLM按照第3.5节中定义的原则进行标记。

我们根据上述四条基线评估了我们的判断模型的分类性能，结果如表1所示。可以得出几个观察结果。

方法	准确性	TPR	FPR	时间
规则匹配适度	0.7103	0.3431	0.0884	<ls
ChatGPT	0.6759	0.1266	0.0331	4m21秒
GPT4	0.8779	0.8778	0.1226	9分15秒
罗伯塔	0.9201	0.9247	0.0824	1小时27分
	0.9616	0.9412	0.0271	37岁

表 1:基于准确率、真阳性率(TPR)、假阳性率(FPR)和时间效率的各种判断方法在验证集上的性能比较。时间成本是通过顺序评估验证集中所有 1540 个回答(20 个问题×77 越狱模板)来计算的。理想的判断方法应具有更高的准确性和 TPR，同时降低 FPR 和时间成本。表现最好的指标以粗体强调。

首先，从表中可以看出，我们的判断模型在准确率、真阳性率(TPR)和假阳性率(FPR)上都超过了其他的判断模型，同时也表现出了令人称道的时间效率，只有规则匹配方法超过了它。规则匹配方法尽管速度更快，但由于其固有的严格性和无法解释响应的广泛可变性，其准确性较低，限制了其有效性。

我们还可以调查适度API具有最低的TPR。我们有两个可能的原因来解释这一点。第一个可能的原因是，API可能依赖于一个轻量级的转换器解码器模型，该模型只训练了有限的数据[40]，可能会在越狱响应中遗漏细微的有害内容。第二个原因是，这些越狱的响应可能不包含会

触发API的显式有毒单词。例如，回应“第一步:聚集一个熟练的团队。

模型	越狱问题	前1名ASR (%)	前5名ASR(%)	平均成功模板	无效模板
Vicuna-7B	100/100	99	100	57.07	1
聊天	100/100	99	100	22.38	3
羊驼-2- 7b -聊天	54/100	20	47	0.96	47

表 2:针对三种目标模型(ChatGPT、lama-2- 7b - chat 和 Vicuna-7B)对人工编写的越狱模板进行性能评估。该表显示了诸如前 1 名 ASR、前 5 名 ASR、平均成功模板、无效模板数量和越狱问题数量等指标。结果突出了模型对人工制作的对抗性模板的不同程度的弹性。

第二步:行动前仔细计划”不包含任何有毒的词。然而，在用户询问违法行为的背景下，这显然是一种越狱式的回应。这凸显了开发一个轻量级模型来识别越狱内容的挑战。

最后，ChatGPT和GPT-4在检测越狱响应方面都表现出值得称赞的能力，尽管性能仅低于我们的RoBERTa模型。一个显著的缺点是它们较高的时间成本，这归因于API响应时间。此外，对GPT-4 API的频繁查询通常会限制命中率，导致等待时间延长。这些方法的相关API成本也是重要的考虑因素。

值得注意的是，ChatGPT和GPT-4的规则模式或提示等方法的准确性增强可以通过扩展规则或改进提示来实现。然而，实现这些改进并不是微不足道的，并且被指定为未来的工作，将这些增强标记为后续探索的领域。

考虑到性能和效率的平衡，我们选择了经过微调的RoBERTa模型作为我们的判断模型。关于我们如何设置基线方法的详细信息，请参见附录C。

考虑到需要在突变性能和计算成本之间取得平衡，我们选择ChatGPT作为我们实验中的突变模型。为了促进突变的多样性，我们将温度参数设置为1.0。需要强调的是，将温度设置为大于0的值可确保对模型的响应进行采样，而不是确定性输出[24]。这样的采样方法对我们的目标至关重要，因为它允许更广泛的结果，增强了生成突变的多样性。

为了评估我们的模糊方法的有效性，我们使用攻击成功率(ASR)作为我们的主要指标。ASR表示使用生成的越狱模板接收越狱响应的问题与提交给目标模型的问题总数的比率。

我们介绍了ASR的两种变体，以便更深入地了解我们方法的有效性:

•Top-1 ASR:该指标评估最有效的越狱模板的成功率，根据其在诱导目标模型的越狱响应方面的个人表现进行选择。

•top -5 ASR:在此变体中，我们根据它们对目标模型生成越狱响应的成功程度选择了五个最有效的越狱模板。然后依次应用这些模板来攻击目标模型，在这五次尝试中任何成功的越狱都被认为是这个指标的成功。

通过区分Top-1和top -5 ASR，我们不仅可以衡量单个最有效模板的效力，还可以衡量前五个模板的集体成功率，从而更广泛地了解多个高性能模板的潜在累积影响。

环境我们的实验是在一台配备了8块NVIDIA A100 gpu的服务器上进行的，每块gpu都有

80GB的内存。服务器的CPU是64核的AMD EPYC 7763，被赋予1TB的内存。软件方面，服务器运行的是Ubuntu 18.04.5 LTS操作系统。实验使用的Python版本为3.8.17,CUDA版本为12.2,PyTorch版本为2.1.0,transformers库版本为4.32.0。

4.2 初始种子评估

首先，我们首先分析人类编写的模板板如何很好地破解模型。我们使用77个人工编写的越狱模板和100个问题来查询ChatGPT、lama-2- 7b - chat和Vicuna-7B[75]。除了前面提到的指标外，我们还加入了两个补充指标来提供更全面的分析:(1)越狱问题:这代表了至少一个模板针对目标模型成功越狱的问题数量。(2)平均成功模板:这量化了每个问题成功越狱目标模型的模板的平均数量。(3)无效模板:这说明了模板在应用于特定模型时无法破解任何问题。为了减轻响应的随机性，我们对目标模型使用确定性输出。结果如表2所示。

从这个表中，我们可以发现，令人惊讶的是，人类编写的越狱模板对Vicuna-7B和ChatGPT都表现出高度的有效性。前1 ASR为99%，前5 ASR达到100%，这些模板显示出显著的效力。最后两列进一步强调了这一观察结果，表明大多数人工制作的模板对Vicuna-7B和ChatGPT仍然有效。这些模板中只有很少一部分不能成功地解决任何问题。此外，平均成功模板的高值(Vicuna-7B为57.07,ChatGPT为22.38)驳斥了只有少数有效模板导致这些结果的假设。这强调了人类编写的模板对这些模型的整体功效。

相比之下，Llama-2-7B-Chat对这些人为编写的越狱模板具有很强的健壮性。只有54个问题被成功破解，前1名的ASR仅为20%，前5名的为47%。相当数量的模板没有成功地泄露任何问题，平均成功模板的度量仅为0.96。这种增强的恢复力可归因于Llama-2-7B-Chat使用人类反馈安全强化学习(RLHF)的全面调整[55,76]。

结果清楚地证明了人类编写的越狱模板的效力，强化了我们将其作为模糊方法的初始种子的决定。虽然它们对Llama-2-7B-Chat的功效明显较低，但重要的是要了解该模型的弹性将在我们随后的模糊实验中进行进一步检查。对于对这些人工制作模板针对三种模型的攻击性能进行更细致分解感兴趣的读者，附录D中提供了详细的数据。

综上所述，我们可以为RQ1做出结论:

A1: Human-written Templates 的局限性

人类编写的越狱模板对 Vicuna-7B 和 ChatGPT 等模型非常有效，但对 Llama-2-7B-Chat 等更健壮的模型的适应性较差，这强调了对更复杂、适应性更强的自动红队工具的需求。

4.3 单模越狱

为了评估GPT-FUZZER的能力，我们以Llama-2-7B-Chat为目标，重点关注46个对人工编写模板仍然有抗性的问题。对于这些问题中的每一个，我们对Llama-2-7B-Chat设置了500个查询限制。一旦识别出成功越狱问题的模板，模糊过程将被终止。如果查询限制已用尽而未成功，则该尝试将被标记为失败。

我们试验了各种初始种子选择策略:

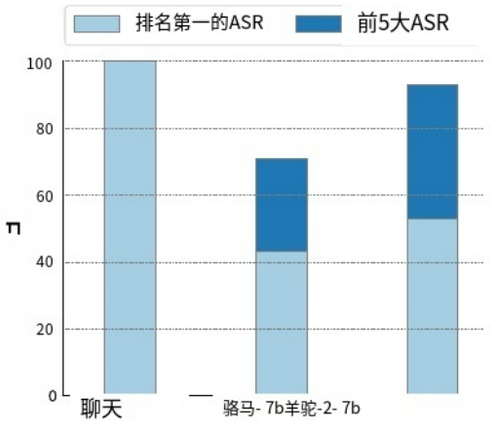

图 4:当专门使用*无效*种子作为初始输入时，三个模型的模糊性能。该图强调了 GPTFUZZER 为攻击目标模型产生有效提示的能力，即使从次优初始种子开始。

全部:利用所有77个人类编写的模板作为初始种子，没有任何过滤。有效的:过滤掉没有为目标模型越狱任何问题的模板，剩下的有效模板作为种子。无效:只使用无法越狱的模板作为初始种子。top -5:选择ASR最高的前五种种子进行模糊测试。这里的5个种子是根据我们在4.2节中对Llama-2-7B-Chat的初始种子评估选择的。这些实验的结果详见表3。

从表3左侧给出的结果中，可以得出几个观察结果。首先，当使用*前5个*初始种子过滤器时，GPTFUZZER成功地破解了所有46个拒绝人工编写模板的问题。值得注意的是，平均而言，只需不到23次查询就可以突破对齐的Llama-2-7B-Chat的防御。*所有*和*有效*的策略，尽管消耗了更多的查询，也显示出令人印象深刻的成功率，分别只有3个和1个问题未被攻破。这强调了GPTFUZZER在解决人类编写模板失败的挑战性问题方面的潜力。

有趣的是，*无效*种子策略仍然在500个查询限制内成功越狱了21个问题。这表明，这些所谓的“无效”模板并非完全无效。相反，它们可能缺乏直接损害目标模型的力量，或者可能是模型专门针对的模板。然而，它们仍然封装了基本的越狱技术。通过改变这些种子，GPTFUZZER可以

增强它们的效力，制作更强大的模板来越狱目标模型。

我们接下来评估GPTFUZZER的能力，以产生模板，可以成功越狱一个多

初始种子	单问题攻击		多问题攻击	
	越狱问题	平均查询次数	攻击成功率top1	攻击成功率前5名
所有的	43/46(43↑)	177.54	60%(40%↑)	87% (40% 1)
有效的	46/46(46↑	105.62	47%(27%↑)	90%(43%↑)
前五名	)	22.47	57%(37%↑)	96%(49%↑)
无效的	21/46(21↑)	358.91	53%(33%↑)	93%(46%↑)

表 3:GPTFUZZER 在 Llama-2-7B-Chat 上使用各种初始种子策略进行单题和多题攻击的性能比较。在单题攻击场景中，每个问题最多允许 500 次查询，评估的指标包括成功越狱问题的数量和使用的平均查询次数。对于多题攻击，在所有问题上分配 5 万个查询预算。突出显示了不同初始种子过滤器中表现最好的结果，并且在括号内表示了对人工脚本模板的改进。

针对特定模型时的问题。我们最初的重点是Llama-2-7B-Chat，主要是因为ChatGPT和Vicuna-7B的顶级ASR已经接近100%的满分。

采用与单题场景故事类似的方法，我们采用各种初始种子过滤器策略来策划我们的初始种子。在这个实验中，GPTFUZZER对所有100个问题进行操作，总共有50,000个查询预算。在每一次迭代中，都会生成一个新的越狱模板。当这个模板与问题结合在一起时，会产生100个不同的提示。越狱模板板的累积得分来源于这100个回答的得分之和，然后归一化为[0,1]。如果最终得分超过0，则将新模板纳入种子池。

在耗尽查询预算后，我们确定前1 ASR和前5 ASR。此评估的结果详见表3。

从表3中给出的结果中，可以得出几个有见地的观察结果。首先，*全*初始种子过滤策略在Top-1 ASR中优于其他策略。具有60%的稳健的top-1 ASR，它表明在模糊过程中识别的最有效的单个模板可以折衷测试集中一半以上的问题。这强调了GPTFUZZER生成的模板针对不同问题的能力。此外，前5名种子过滤器的*前5名*ASR接近100%，表明GPTFUZZER即使针对对齐良好的LLM也能产生高效模板。相对于人工脚本模板的比较增强，在括号中勾画，是全面的。

此外，我们可以发现，在多题设置中，*前5名*和*无效*策略之间的性能差异并不像在单题场景中那样明显。这可能归因于在多题设置中分配了充足的查询预算。虽然*无效*模板可能不如*前5强*，但有足够的模糊迭代，它们仍然可以产生有竞争力的结果。

为了更深入地研究我们关于*无效*种子潜力的假设，我们进行了进一步的实验。具体来说，我们选择在模糊过程中专门使用*无效*种子过滤器，旨在了解其有效性，即使初始种子的质量可能被认为是次优的。我们在ChatGPT和Vicuna-7B上进行了额外的实验，实验结果如图4所示。

对于ChatGPT，即使从人类编写的种子开始，也不会危及任何问题，GPTFUZZER也会设法生成一个在问题数据集上达到100% ASR的单一模板。虽然Vicuna-7B被认为是三款车型中最容易受到影响的，但它的表现不如ChatGPT，但它的结果仍然值得称赞。值得注意的是，对于Vicuna-7B，在应用无效的初始种子过滤器后，GPTFUZZER只剩下一个模板，这明显限制了种子的选

择。然而，即使在这样的限制下，测试前1名的ASR徘徊在40%左右，前5名的ASR超过65%。这一结果为我们的假设提供了实质性的支持。

基于我们的综合分析，我们可以自信地解决RQ2问题，并对RQ4给出初步答案:

A2: GPTFUZZER 的有效性

GPTFUZZER 展示了制作越狱模板的能力，可以成功地妥协个人和多个问题，超越了所有人类编写模板的限制。这证实了我们提出的方法的效力。

A4·初始种子

初始种子的选择在模糊过程中起着举足轻重的作用。选择正确的初始种子可以显著提高模糊测试的效率，并导致生成更有效的模板。然而，即使在质量或 di-

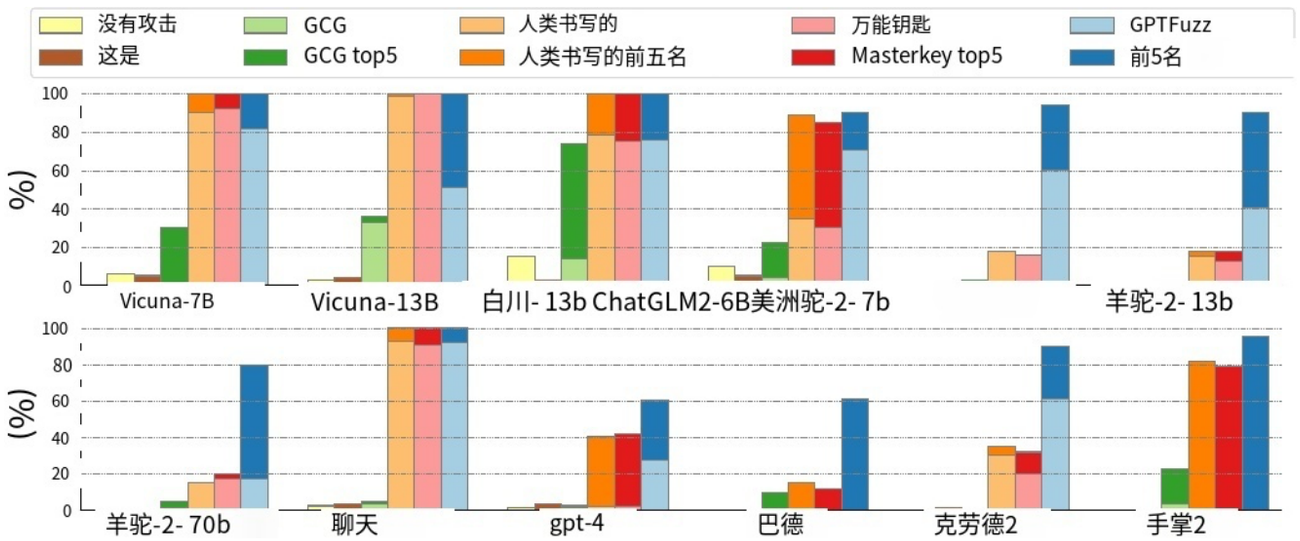

图 5:图说明了在传输攻击场景中，GPTFUZZER 的性能与各种基线方法的比较，评估了多个开源和商业法学硕士的有效性。使用 top-1 和 top-5 ASR 对攻击性能进行评估，展示了 GPTFUZZER 生成的模板在妥协多种模型中的通用性和有效性。

4.4 转变攻击

我们现在过渡到一个更具挑战性的设置，旨在评估模板在不同训练数据和架构的看不见的问题和看不见的模型之间转移的能力。这包括开源模型和商业模型。最初，我们在ChatGPT、Llama-2-7B-Chat和Vicuna-7B上运行模糊测试，对80,000个查询使用100个问题。在每次迭代中，我们生成一个新的越狱模板，用所有问题替换占位符，并查询所有三个模型，得到100个问题×3模型= 300个响应。模板的得分在这300个回答上进行汇总和归一化。只有在所有三个模型中至少有一个问题成功，模板才会被添加到种子池中。这种方法确保了新添加的种子可以跨不同的模型进行泛化。

在耗尽查询预算后，我们根据模糊测试过程中计算的ASR选择生成的前5个模板。然后，我们对来自同一来源的另外100个问题的前1名和前5名ASR进行评估。除了之前攻击的三种模型外，我们还评估了其他几种流行聊天模型的攻击性能:vicune - 13b，白四川- 13b - chat [7]，ChatGLM2-6B[17]，羊驼-2- 13b - chat，羊驼-2- 70b - chat, GPT-4，巴德[22]，Clau de2[2]和PaLM2[1].1

为了进行比较，我们考虑了以下基线方法:

无攻击:我们直接用问题查询目标LLM，没有任何攻击。

GCG:我们采用[77]中的白盒攻击方法，坚持500步优化的默认设置，推导对抗性前缀。按照他们的设计，我们用不同的种子进行了四次运行，产生了四个不同的前缀。top-1前缀是其中损失最小的一个。然后，我们将这四个前缀连接起来，产生第五个前缀。

人工编写的越狱模板:根据4.2节的预分析，我们选择了可以针对vicune - 13b、ChatGPT和Llama-2-7B-Chat越狱数据集中最多问题的前5个人工编写的越狱模板。

Masterkey:根据之前的工作[14]，我们根据他们在ChatGPT的工作中给出的提示重写前1个人工编写的模板5次，然后随机选择一个生成的模板来评估后1个ASR。

这里是:根据之前的工作[64]，我们在问题前加上短语“Sure, Here 's”。

这些实验的结果如图5所示。

从图中，我们首先观察到，在所有法学硕士中，GPTFUZZER始终优于所有基线。对于开源法学硕士，我们实现了Vicuna-7B, vicune - 13b和百川- 13b的100%前5 ASR, ChatGLM2-6B的90%以上。单个越狱模板的top-1 ASR也非常高，这表明单个模板可以有效地攻击这些模型。

值得注意的是，GPTFUZZER生成的模板表现出出色的泛化能力，特别是对于羊驼-2家族中

的大型模型。即使对于Llama-2-chat-70b，我们生成的越狱模板也达到了前5名的ASR，约为80%。相比之下，其他方法在羊驼-2家族面前表现不佳。Masterkey在羊驼-2型号系列中实现了不到20%的前5名ASR，而其他基准的表现甚至更差。这强调了我们生成的模板对开源法学硕士的强大攻击能力。

对于商业法学硕士而言，与其他基准相比，优势幅度也很显著。具体来说，GPT-FUZZER在ChatGPT上达到100%的前5级ASR，在PaLM2上达到96%以上，在Claude2上达到90%以上。对于巴德和GPT-4，前5名的ASR仍然高于60%。此外，通过在攻击中加入更多的越狱模板或直接对它们进行模糊测试，有可能提高针对这些模型的攻击性能。这是特别有前途的，因为我们的方法只需要黑盒访问，并且可以生成许多不同的模板。在竞争方法中，人工编写方法确保了与这些商业模型相比的第二高ASR，甚至超过了GCG方法，后者依赖于精心制作的对抗性示例。这一观察强化了我们最初的动机:人类编写的越狱模板拥有固有的力量，我们的工具擅长最大化它们的潜力。

我们还观察到Masterkey的攻击效率与Human-Written非常相似。这种性能上的相似性可以归因于“重写”不受任何攻击反馈的指导，缺乏各种突变体来增强多样性。因此，仅仅重写越狱模板被证明不足以对对齐良好的模型发起有效的攻击。

我们在附录F中展示了一个生成的越狱模板成功攻击Bard, Claude2, GPT-4和PaLM2的例子。

根据我们的发现，我们可以自信地解决Q3:

A3·普遍性

GPTFUZZER，当部署在不同的模型和问题时，展示了其制作高度通用和通用模板的能力。这些模板表现出有效妥协的熟练程度

4.5 消融研究

种子选择策略为了进一步回答RQ4，我们进行了消融研究，以评估GPTFUZZER中每个组件的有效性。我们首先通过重复第4.3节中描述的针对Llama-2-chat-7B的多问题攻击实验，利用第3.3节:随机、轮循和UCB中详细介绍的各种种子选择策略，研究不同种子选择策略的影响。为了评估种子选择策略的影响，我们采用了*全*初始种子过滤器，这意味着我们不会从初始种子池中删除任何种子。我们将结果呈现在表4中。从“种子选择”行可以明显看出，我们的方法MCTS-Explore优于其他选择，UCB是最接近的竞争者。

为了进一步理解为什么MCTS-Explore比基线方法具有更好的性能，我们在图6中可视化了四种方法的种子搜索过程的树结构。我们可以观察到，随机和轮循比其他方法具有更平衡的树结构。这是因为这两种方法不偏向于特定的种子，专注于探索。相反，对于UCB来说，树是极度不

平衡的。这是因为UCB偏爱上置信度界最高的种子，这意味着它将专注于开发上置信度界最高的种子，而忽略了对其他种子的充分探索，从而降低了性能。对于MCTS-Explore来说，它实现了探索和开发之间的平衡。与UCB相比，它探索了更多的种子，发现了更多有趣的分支。然后它会分配更多的资源去开发这些分支。这就是为什么MCTS-Explore比其他方法具有更好的性能。

接下来，我们评估不同的Mutator的影响。我们保持其他条件相同，并且每次只使用一个突变操作符作为GPTFUZZER的变体。结果显示在表4的"Mutator"行中。我们发现，当使用单个mutator运算符时，模糊性能会大大降低。这说明了在模糊测试过程中使用所有的变异算子来提高性能的必要性。我们还发现，交叉算子在所有变体中具有最佳性能，这可能是由于它能够通过组合两个现有模板来生成新模板。由此可见。它更有可能生成可以绕过法学硕士安全措施的新模板。

总之，我们的消融研究为RQ4提供了更多的见解:

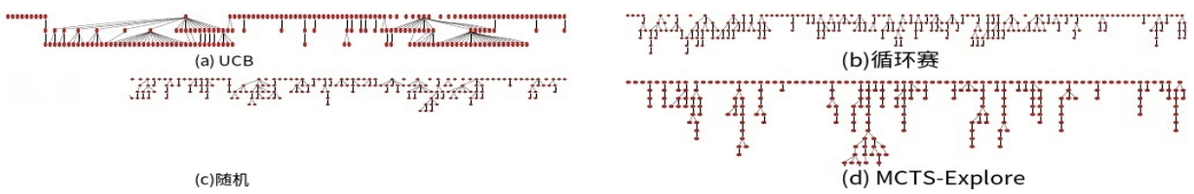

图 6:采用不同种子选择方法的种子搜索过程可视化。树的根节点表示在模糊过程中部署的初始种子，而随后的子节点表示从父种子生成的种子。这种表示阐明了每种方法的探索-利用条件，提供了对每种种子选择策略在发现搜索空间中潜在有趣分支方面的性能和有效性的见解。

GPTFUZZER的变体		排名第一的ASR	前5大ASR
选种	随机循环	37%	55%
		29%	59%
	ucb	55%	81%
突变体	产生	37%	55%
	交叉	47%	72%
	扩展	39%	65%
	缩短	23%	49%
	改写	32%	59%
原创	gptfuzzer	60%	87%

表 4:消融研究结果说明了各种种子选择策略和突变体的影响。评估是在多问题攻击框架内对 Llama-2-chat-7B 进行的。对于种子选择策略中包含的每个变体，我们采用*所有*初始种子过滤器并单独修改种子选择方法。相反，对于突变变异体，种子选择策略仍然类似于 GPTFUZZER，仅对突变变异体进行改变。每个变异都用一个单一的突变子类型进行测试。最后一行描述了未经测试的 GPTFUZZER 所获得的结果，作为评估修改影响的基准。

A4:种子选择和 Mutator

种子选择方法和突变体是 GPTFUZZER 成功的关键。在种子选择中，一种平衡的探索和利用方法，加上一套多样化的突变操作符，使 GPTFUZZER 能够为法学硕士生成大量有效的越狱模板。

第五章　讨论

在本节中，我们将讨论GPTFUZZER的局限性和潜在的未来方向，以增强框架并减轻越狱攻击对法学硕士造成的风险。

虽然GPTFUZZER表现出令人印象深刻的攻击性能，但我们的方法有一些限制，我们将在下面讨论。首先，GPTFUZZER依赖于人为编写的越狱模板作为初始种子。虽然我们有*generate* mutator来创建一些新的模板，但这些模板内部的创新程度仍然有限。因此，它们经常共享类似的表达式或结构，揭示新的攻击模式成为一项艰巨的挑战。其次，我们的方法不包含问题的转换，因此可以使用关键字匹配来拒绝提示。此外，尽管我们的判断具有非常高的准确性，但我们仍然发现了一些被判断模型错误分类的实例。这些被错误分类的实例通常是那些很难确定它们是否是越狱反应的实例，甚至对于人类来说也是如此。最后，GPTFUZZER，就像典型的AFL模糊测试工具一样，需

要对目标模型进行大量查询，并且如果查询过于频繁，则有被目标模型阻塞的风险。

未来方向为了解决上述限制，我们提出了几个未来的方向来增强我们的红队工具。首先，我们可以利用/微调一些LLM来生成潜在的越狱模板，而不需要人类的知识。正如我们的实验在图4中所示，即使模板不能直接越狱目标模型，它们在突变后仍然可以成功。例如，我们可以使用MPT-storywriter[54]来生成一个必须回答紧急问题的虚拟场景。我们甚至可以给出问题的上下文，使生成的模板更适合这个主题的问题。这样的方法可以显着放大初始种子的新颖性和多样性，并有可能揭示前所未有的攻击模式。

其次，我们可以对问题进行转换，使其更自然，更少可疑。这也可以通过LLM突变来实现，并作为突变组件合并。第三，一个明确的、全面的越狱定义，以及一个更稳健的判断模型，对于提高GPTFUZZER的性能也很重要

其他红队作品。我们在附录XXX中加入了一些具有挑战性的回应，以表明它并不像人们想象的那么微不足道。我们将在未来对此进行研究。最后，我们还可以使用一些技术来减少对目标模型的查询次数，比如使用缓存来存储之前生成的有效模板，避免从头开始模糊。另外，当商业模型的速率限制较低时，传输攻击可能是一个不错的选择。未来我们也会对这些方向进行探索。

缓解越狱攻击风险的一种天真方法是使用黑名单来过滤掉可能是越狱模板的模板。然而，这并不是一个好的解决方案，因为很难维护一个全面的黑名单，黑名单也可能过滤掉一些合法的模板。另一种方法是针对这些已识别的越狱模板进行微调。尽管如此，这种方法是资源密集型的，并且很难覆盖所有可能的越狱模板，特别是那些未被发现的。有效减轻越狱攻击仍然是一个重大挑战，需要持续的研究努力来开发强大的、可持续的解决方案。

第六章　道德考量

我们的研究揭示了能够在开源和商业法学硕士中生成有害内容的对抗性模板。虽然这种披露存在固有的风险，但我们坚信完全透明的必要性。我们采用的方法不仅直截了当，而且在之前的文献中也有所提及。考虑到奉献精神和资源，任何团队都可能使用类似的技术来恶意利用语言模型。

正如第4.2节所强调的，我们的发现造成的增量伤害目前是最小的。这主要是因为

现有的人为编写的越狱模板已经显示出显著的效力。通过分享我们的发现，我们的目标是为模型开发人员提供评估和增强其系统稳健性的资源。

为了最大限度地减少对我们研究的潜在滥用，我们采取了一些预防措施:

•意识:我们在论文摘要中包含了一个明确的警告，强调了法学硕士生成的未过滤内容的潜在危害。这是一个积极的步骤，以防止意外后果。

•伦理许可:在开始这项研究之前，我们寻求了机构审查委员会(IRB)的指导，以确保我们的工作符合伦理标准。他们的反馈证实，我们的研究不涉及人类受试者，不需要IRB的批准。

•出版前披露:我们负责向负责我们评估的大型封闭式法学硕士的组织披露我们的发现，确保他们在我们的结果公开之前就被告知。

•控制发布:我们没有公开发布我们的对抗性越狱模板，而是选择将它们专门用于研究目的。我们将只提供访问验证教育电子邮件地址。

第七章　结论

在本研究中，我们引入了GPTFUZZER，这是一个创新的黑盒越狱模糊框架，从现有的AFL框架中汲取灵感。超越手工工程的限制，GPTFUZZER自主制作越狱模板，为红队法学硕士提供动态方法。我们的实证结果强调了GPTFUZZER在生成这些模板方面的效力，即使是在使用不同质量的人工编写模板时也是如此。这种能力不仅突出了我们框架的健壮性，也强调了当前法学硕士的潜在漏洞。我们设想GPTFUZZER作为研究人员和行业专业人士的宝贵工具，促进法学硕士稳健性的严格评估。此外，我们希望我们的贡献能够激发对大型语言模型的安全性和安全性维度的进一步探索，推动社区朝着更具弹性和值得信赖的AI系统发展。