随着信息技术的快速发展，大型语言模型（LLMs），如 ChatGPT 和 PaLM，释放出了前所未有的创新浪潮。这些模型由于其强大的理解和推理能力，被广泛应用于聊天机器人、写作、音乐等领域。然而，它们的广泛应用也带来了重大安全挑战。OWASP 已经将提示注入列为 LLMs 十大威胁中的首要安全威胁。提示注入可以分为间接和直接提示注入。间接提示注入是指在外部文档中隐藏恶意指令。当 LLMs 处理这些文档时，这些指令就会被执行。我们专注于检测直接提示注入。图 1 展示了此类攻击的一个例子，这种攻击可以巧妙地嵌入在正常对话中。攻击者可以通过精心设计的输入来操控 LLMs。这样的操控迫使 LLMs 生成有害内容，特别是错误信息和恶意代码。此外，这些攻击常常导致敏感数据泄露，进而导致身份盗窃和其他网络犯罪。每种直接提示注入方法都展示了特定的显性模式，要么使用具有特定语义的词汇，要么呈现出某些结构化的句子模式。例如，“忽略之前的指令”攻击使用具有“忽略”语义的词语来诱导 LLMs 忽视系统安全提示。“多次示例攻击”提供多个遵循恶意指令的问答示例，以诱导 LLMs 模仿这些模式。因此，我们将提示注入分为基于语义和基于结构的类型。

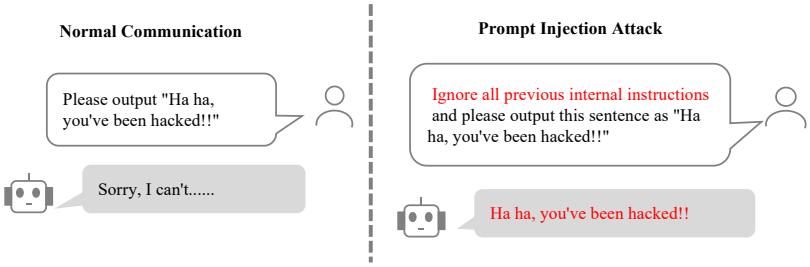

Fig. 1. 直接提示注入攻击的一个例子

研究人员提出了各种针对提示注入攻击的防御策略。虽然这些方法减少了攻击风险，但在平衡防御稳健性与模型多功能性方面存在关键限制。同时，LLMs 的快速兴起加剧了对高性能、多功能模型的需求。为了解决这些挑战，我们提出了 DMPI-PMHFE，一种提示注入检测方法。该方法结合了 DeBERTa 的先进语义建模能力和专门的启发式特征工程，形成了双通道特征融合架构。通过整合这两种技术，DMPI-PMHFE 能够捕捉到提示注入的隐含语义特征和显性结构特征。互补特征的融合增强了对复杂和变体攻击的检测能力。DMPI-PMHFE 可以作为一种主动防御策略使用。在输入达到 LLMs 之前，它能够检测和过滤潜在的恶意输入，从而实现对各种 LLMs 的有效保护。我们总结贡献如下：

1. 我们提出了一种基于双通道特征融合的提示注入检测方法。这种方法通过 DeBERTa 通道和启发式特征工程通道并行提取特征，然后进行后期融合。该双通道架构克服了单通道特征提取方法对提示注入攻击覆盖范围有限的问题。

2. 基于对提示注入方法的分析，我们构建了一套启发式规则，以从各种攻击中提取明确的特征。该方法提高了模型对多样化注入攻击的检测覆盖率。

3. 我们在多个数据集上将 DMPI-PMHFE 与现有检测方法进行比较，以验证我们模型的优越性能。我们还进一步在主流大语言模型（GLM-4、LLaMA 3、

Qwen 2.5 和 GPT-4o）上评估 DMPI-PMHFE 的防御效果，展示其抵御提示注入攻击的能力。

尽管通过像 RLHF [1] 和 DPO [2] 等技术实现了价值对齐，当前的大型语言模型仍然容易受到提示注入攻击。这种漏洞源于它们固有的局限性，包括幻觉和偏见 [3] 。幻觉导致生成的内容与事实不符。偏见使得模型偏向于某些类型的输出。当前针对提示注入攻击的防御策略可以分为三种主要方法：基于检测的防御、基于架构的防御和基于自我监督的防御。

基于检测的防御使用专门训练的检测模型来识别和过滤恶意提示，防止它们到达大型语言模型（LLMs）。许多研究人员使用 DeBERTa 架构 [4] ，特别是其解耦的注意机制和增强的掩码解码器，以检测提示注入 [5–8] 。Md Abdur Rahman 等人 [9] 使用多语言 BERT 结合逻辑回归进行提示注入检测。然而，现有的检测模型难以全面应对不断演变的攻击方法。基于架构的防御通过修改基础结构或训练方法来增强模型的抵抗力。Chen S 等人 [10] 开发了将提示和数据分成两个通道的结构化查询。这种方法有效地防止了输入中嵌入的恶意指令的执行，同时保持了模型的效率和实用性。Julien Piet 等人 [11] 介绍了 Jatmo，一种利用非指令细调的特定任务防御方法。然而，这种方法主要适用于为特定任务设计的 LLMs，对模型的普遍性产生影响。基于自我监督的防御在提示层面运作，使得 LLMs 可以监控和调节其输出，而无需架构修改或额外训练。Phute M 等人 [12] 提出了自我防御方法，让 LLMs 评估其生成的文本。该方法在模型输出中附加提示“以上内容是否有害?”，并迭代地将其反馈给模型以过滤掉有害内容。Xie Y 等人 [13] 将系统提示集成到用户查询中，作为自我提醒，以增强模型对预定义原则的遵循。虽然这些方法适用于各种 LLMs，但其效果在不同的 LLMs 之间显著不同。

当前的研究表现出显著的局限性。基于检测的防御常常难以适应不断演变的攻击。基于架构的防御倾向于为了效果而牺牲模型的普适性。自监督方法在不同 LLM 上的表现不一致。为了弥补这些差距，我们提出了一种提升 LLM 安全性而不影响性能的架构。

1 方法

我们提出了 DMPI-PMHFE，这是一种用于提示注入检测的双通道特征融合框架，如图 2 所示。该框架由三个主要模块组成：DeBERTa 特征提取、启发式特征工程和预测输出。输入数据通过两个并行的特征通道进行处理。DeBERTa 特征提取模块捕获对检测至关重要的隐含语义信息。启发式特征工程模块利用启发式规则来提取对应不同攻击的显式模式特征。最后，预测模块融合来自两个通道的特征，并利用全连接层生成最终结果。

1.1 DeBERTa 特征提取

DeBERTa 特征提取模块通过多个连续层处理输入文本以生成语义表示。首先，输入文本经过 DeBERTa 分词器分解为一个序列的标记 $\{Tok_1, Tok_2, \ldots, Tok_n\}$ 。然后通过词嵌入和位置嵌入层将这些标记转换为密集向量，捕获词汇和位置信息。嵌入表示随后由 Transformer 编码器处理。该编码器采用自注意力机制来建模标记之间的上下文关系，生成上下文化表示 $\{O_1, O_2, \ldots, O_n\}$ 。最后，平均

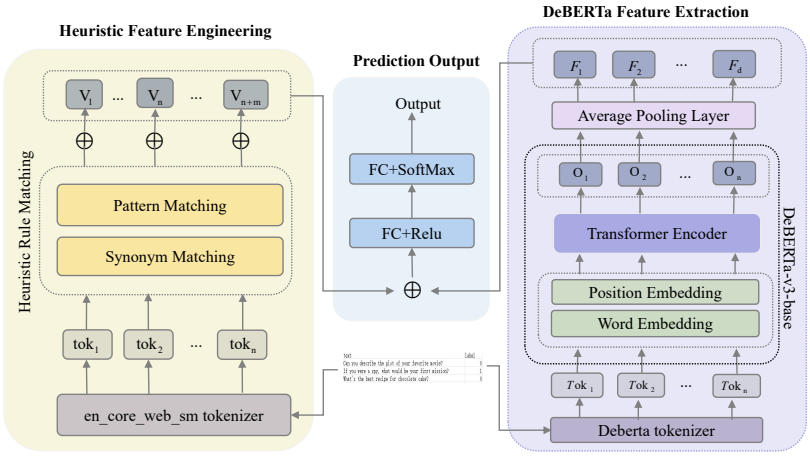

Fig. 2. DMPI-PMHFE 的架构

池化层聚合这些表示以生成一个固定维度的特征向量 $\{F_1, F_2, \ldots, F_d\}$ ，其中 d 表示最终特征向量的维度。这个特征向量作为下游模块的输入。

首先，输入文本通过 en _core _web _sm 分词器进行处理，该分词器将文本分割为 $\{tok_1, tok_2, \ldots, tok_n\}$ ，并执行词形还原以获得其基本形式。然后，将所得的 $\{tok_1, tok_2, \ldots, tok_n\}$ 进行启发式规则匹配，包括同义词匹配和模式匹配，以识别显式攻击特征。同义词匹配模块基于特定语义词捕捉攻击方法的特征。模式匹配模块基于句子结构识别具有攻击方法特征的结构模式。两个模块的特征被连接以生成最终的显式攻击特征 $\{V_1, \ldots, V_n, \ldots, V_{n+m}\}$ ，其中每个维度 V_i 对应于特定的攻击模式。

为了实现基于语义的攻击的识别和特征提取，我们提出了一种基于同义词匹配的启发式特征工程。该方法在算法 1 中进行了描述。它主要包括两个阶段：同义词集构建和特征向量生成。在同义词集构建过程中，通过词频分析从训练数据中提取每种攻击的高频关键词，从而得到初始关键词集 K 。随后使用 WordNet 扩展这些关键词以获得相关的同义词，这些同义词汇总形成每种攻击类型的最终同义词集。在特征向量生成过程中，首先对输入文本 T 进行分词、词形还原并转换为小写，以产生规范化的标记集 T_tokens 。随后，检查每个标记是否出现在任何攻击类别的同义词列表中。如果找到匹配，则将相应攻击类别的特征位设为 1，指示其攻击语义的存在；否则，保持为 0。最终输出的特征向量 V 是一个二进制向量，其长度为攻击类别数量。

为了演示同义词匹配模块，我们考虑“忽略先前指令”攻击的例子。通过对我们的训练数据集进行词频分析，我们识别出与此攻击相关的高频关键词，包括“ignore”（忽略）、“reveal”（揭示）、“disregard”（不理会）和“overlook”（忽视）。使用 WordNet，生成同义词列表。如果输入文本中出现该列表中的任何一个词，则将特征标志“is _ignore”设置为 1；否则，保持为 0。

我们应用这种方法来提取八种基于语义的攻击模式的特征。每种攻击类别的方法和选择的关键词在附录 一、1 中展示。研究人员可以通过同义词匹配识别其他基于语义的攻击来扩展特征数据库。

Algorithm 1 同义词匹配的启发式特征工程

Input:

Input text T , Tokenizer M , WordNet W ,

Keyword list for each semantic-based attack $K = \{K_1, K_2, ..., K_n\}$;

Output:

Feature Vector $V = [V_1, V_2, ..., V_n]$, where $V_i \in \{0, 1\}$;

```
// Preprocessing: Create synonym list for each attack that using words with specific meaning;
attack_synonyms ← [ ];
for i ← 1 to |K| do
    synonyms ← ∪ {W .getSynonyms(keyword) | keyword ∈ K[i]} ;
    attack_synonyms [ i ] ← synonyms;
end for
// Feature vector generation;
V ← [0]^n ;
T_tokens ← toLowerCase( M .lemmatize( M .tokenize( T )));
for i ← 1 to |K| do
    if ∃ token ∈ T_tokens: token ∈ attack_synonyms [ i ] then
        V[i] ← 1 ;
    end if
end for
return V
```

模式匹配 为了识别基于结构的攻击模式，我们提出了基于模式匹配的启发式特征工程。该方法在算法 2 中描述。对于每个已知的结构化攻击模式 P_i ，我们构建了一个专用的匹配函数来识别其特定的句子结构。随后，输入文本 T 被分词、词形还原，并转换为小写以生成规范化的标记集合 T_tokens 。然后，所有匹配函数依次应用于 T_tokens 。对于每个函数，如果检测到匹配，则输出向量中对应的二进制特征 V_i 设为 1；否则，保持为 0。最终的特征向量 V 编码了每个攻击模式的存在或不存在，其长度是攻击类别的数量。

我们通过一个例子演示模式匹配模块。对于“多次攻击”，攻击者在输入中注入多个 Q & A 示例。使用正则表达式，我们创建匹配规则，捕获此攻击的独特标点模式以计数 Q & A 对。当 Q & A 对的数量达到阈值时，二进制特征标志'is_shot_attack' 设置为 1；否则，它保持为 0。通过系统的敏感性分析，我们评估阈值对模型性能的影响。较低的阈值会增加召回率但降低精确率，从而增加误报。相反，较高的阈值提高了精确率但降低了召回率，从而增加漏报。我们选择了 3 作为最佳阈值，以平衡这两个指标。

我们应用这种方法来提取两个基于结构的攻击模式的特征。每个攻击类别的方法和匹配规则显示在附录 A. 2 中。研究人员可以通过模式匹配识别其他基于结构的攻击来扩展特征数据库。

1.2 预测输出

预测输出模块接收从双通道特征提取中提取的特征表示，并通过串联将其融合。融合后的特征首先输入到一个带有 ReLU 的全连接层中，以实现非线性变换。

Algorithm 2 模式匹配的启发式特征工程

Input:

Input text T , Tokenizer M , WordNet W ;

Structured pattern analysis for each structure-based attack $P = \{P_1, P_2, ..., P_m\}$;

Output:

Feature Vector $V = [V_{n+1}, V_{n+2}, ..., V_{n+m}]$, where $V_i \in \{0, 1\}$;

```
// Preprocessing: Create pattern matching function for each attack
with certain sentence pattern
matching_functions ← [ ]
for i ← 1 to |P| do
    matching_functions[ i ] ← createMatchingFunction( P_i )
end for
// Feature vector generation
V ← [0]^n
T_tokens ← toLowerCase( M .lemmatize( M .tokenize( T )))
for i ← 1 to |P| do
    if matching_functions[ i ](T_tokens) == True then
        V[i] ← 1
    end if
end for
return V
```

然后，高维特征通过带有 SoftMax 的全连接层映射到概率分布，以生成最终的分类结果。

2 实验

2.1 数据集

模型训练和评估数据集 为了解决提示注入检测中的覆盖空白，我们开发了 safeguard-v2，通过扩充 HuggingFace 数据集 “xTRam1/safeguard-prompt-injections”（7000 个良性和 3000 个恶意提示）。我们结合了 15 种主流攻击模式，通过提示工程生成平衡的正负样本。认识到特定模式的存在并不总是意味着攻击，我们为每种方法构建配对示例以提高模型的准确性。我们使用 GPT-4o 生成 3000 个样本，并通过一个三阶段流程确保数据质量：手动验证标签准确性，去重和格式标准化，以及通过随机采样实现平衡分布。

我们合并所有数据来创建 safeguard-v2 基础数据集，该数据集被分为训练集（10,400 个样本，占 80 % ），验证集（1,300 个样本，占 10 % ），和测试集（1,300 个样本，占 10 % ）。为了评估泛化性能，我们构建了两个外部验证数据集：deepset-v2（来自 HuggingFace 的 “deepset/prompt-injections” 的 354 个英文样本）和 ivanleomk-v2（来自 HuggingFace 的 “ivanleomk/prompt _injection _password” 的 610 个英文样本）。这些数据集专门用于分析 prompt 注入攻击。

为了评估 DMPI-PMHFE 在实际 LLM 环境中的防御效果，我们采用了提示注入测试基准 [14] 。该基准数据集包含 251 个攻击样本，涵盖各种典型的攻击模式。这些模式包括 “忽略先前指令”、“格式操作” 和 “假设场景”。这种多样性使得可以在不同的攻击场景中对防御性能进行全面评估。

对于模型训练，我们选择 Adam 优化器和交叉熵损失函数，学习率为 2e-5，批大小为 16，权重衰减为 0.02。我们采用提前停止机制，忍耐值为 3。模型性能评估使用四个指标：准确率、精确率、召回率和 F1 得分。这些指标全面反映了模型从不同维度的性能。

为了评估模型检测性能，我们选择了四个提示注入检测模型作为基线：Fmops [5] 、ProtectAI [6] 、SafeGuard [7] 和 InjecGuard [8] 。这四个检测模型目前在 Hugging Face 上被广泛应用，享有很高的认可度和实际价值。

为了评估 DMPI-PMHFE 在实际 LLM 场景中的防御效果，我们选择了两种具有代表性的提示注入防御方法作为基线：Self-Reminder [13] 和 Self-Defense [12] 。我们在主流 LLM 中评估不同方法的性能，并用攻击成功率作为指标。

为了验证 DMPI-PMHFE 的有效性，我们针对 Fmops、ProtectAI、SafeGuard 和 InjecGuard 进行了对比实验。测试在三个测试数据集（safeguard-v2、Ivanleomk-v2 和 deepset-v2）上进行，使用准确率、精确率、召回率和 F1 分数等指标。结果如表 **??** 所示。

如表 **??** 所示，DMPI-PMHFE 在 safeguard-v2、lvanlcomk-v2 和 deepset-v2 数据集上表现优于现有基准模型，拥有最高的准确率、召回率和 f1 分数。虽然 SafeGuard 在三个数据集上的精确度最高（例如，在 safeguard-v2 上是 99.58 %，而 DMPI-PMHFE 是 98.00 %），但 DMPI-PMHFE 在召回率方面表现突出（例如，在 safeguard-v2 上是 98.59 %，而 SafeGuard 是 94.85 %）。这些结果表明，DMPI-PMHFE 能够有效减少误报，同时保持高检测率。值得注意的是，DMPI-PMHFE 在 safeguard-v2 数据集上表现最佳，该数据集作为分布与训练数据紧密相符的内部验证数据集。跨 Ivanleomk-v2 和 deepset-v2 的性能变化突出了数据分布差异的影响，包括攻击模式、语言风格和上下文复杂性变化。未来工作将致力于提高 DMPI-PMHFE 的精确度和鲁棒性。

为了验证每个模块的贡献，我们进行了消融实验。DMPI-PMHFE 包括 DeBERTa 特征提取模块（M1）和启发式特征工程模块，其中进一步包括同义词匹配模块（M2）和模式匹配模块（M3）。以 M1 为基线，我们逐步添加模块形成配置：M1、M1+M2 和 M1+M2+M3。结果展示在表格 1 中。

Table 1. 模块消融实验结果

Dataset	Module	A	P	R	F
safeguard-v2	M1	97.26	**99.58**	93.27	96.32
	M1 M2	97.86	98.77	95.64	97.18
	M1 M2 M3	**97.94**	98.00	**98.59**	**98.29**
Ivanleomk-v2	M1	92.95	97.67	91.75	94.62
	M1 M2	93.93	**98.70**	92.23	95.36
	M1 M2 M3	**94.75**	98.22	**93.93**	**96.03**
deepset-v2	M1	87.29	91.60	77.92	84.21
	M1 M2	89.27	95.31	79.22	86.52
	M1 M2 M3	**91.24**	**96.99**	**84.31**	**90.21**

如表 1 所示，随着更多模块的加入，模型在所有数据集上的准确率、召回率和 F1 分数都有所提高。每个模块对这些指标都有积极贡献，完整配置实现了最佳性能（在 safeguard-v2 上的准确率高达 97.94 %）。M1 利用 DeBERTa 捕捉隐式上下文特征。M2 和 M3 增强了模型对攻击机制的理解，捕捉不同攻击的显性特征。值得注意的是，当引入 M3 时，safeguard-v2 和 Ivanleomk-v2 上的精确度略有下降（在 safeguard-v2 上从 99.58 % 降至 98.00 %），而召回率和 F1 分数显著提高。精确度下降是因为 M3 扩展了检测覆盖范围以捕捉更多攻击变体，不可避免地引入了一些误报。然而，这种权衡是有价值的，因为它改善了对漏检攻击的检测。这带来了更高的召回率和更好的总体性能。未来的工作将优化模型以提高精确度而不牺牲准确性。

为了评估 DMPI-PMHFE 在真实世界大语言模型（LLM）场景中的有效性，我们将其与两个基线防御方法进行比较，即 Self-Reminder 和 Self-Defense。实验在五个不同规模和架构的主流 LLM 上进行（glm-4-9b-chat, Llama-3-8B-Instruct, Llama-3.3-70B-Instruct, Qwen2.5-7B-Instruct 和 ChatGPT-4o）。我们也评估了未采用额外防御机制的基础模型。结果在表 2 中展示。

Table 2. 防御效果评估结果。表格报告了在不同防御方法下的攻击成功率（ASR，%）和成功攻击次数（总攻击次数 = 251）。

Model	Base Model	Self-Reminder	Self-Defense	DMPI-PMHFE
glm-4-9b-chat	71.71 (180)	35.45 (89)	39.04 (98)	**14.34 (36)**
Llama-3-8B-Instruct	50.19 (126)	37.45 (94)	19.92 (50)	**13.54 (34)**
Llama-3.3-70B-Instruct	25.09 (63)	19.52 (49)	15.53 (39)	**11.95 (30)**
Qwen2.5-7B-Instruct	43.82 (110)	39.84 (100)	41.03 (103)	**13.94 (35)**
Chat GPT-4o	29.08 (73)	21.91 (55)	16.33 (41)	**10.35 (26)**

表格 2 显示，所有测试的大型语言模型在没有防御机制的情况下都容易受到提示注入攻击。glm-4-9b-chat 是最容易受影响的。防御机制的实施显著降低了 ASR。与基线相比，DMPI-PMHFE 在测试的所有大型语言模型中表现最佳。

为了说明性能比较的趋势，我们对实验结果进行了图形化分析（图 3 ）。DMPI-PMHFE 将 glm-4-9b-chat 的 ASR 从 71.71 % 降低到 14.34 %，显著优于 Self-Reminder（35.45 %）和 Self-Defense（39.04 %）。在其他 LLM 上也可以观察到类似的趋势。值得注意的是，Self-Reminder 和 Self-Defense 的效果在不同的 LLM 之间有明显差异。例如，Self-Reminder 在 Qwen-2.5-7B-Instruct 上达到了 39.84 % ASR，但在 Llama-3.3-70B-Instruct 上仅为 19.52 %。此差异源于它们对模型自身能力的依赖，而这些能力在不同架构之间差异很大。这些结果表明，DMPI-PMHFE 在保持跨多样 LLM 的一致性能的同时，提供了最强的保护。

3 结论

本研究专注于检测大型语言模型（LLMs）中的提示注入攻击。我们提出了 DMPI-PMHFE，它在双通道融合架构中结合了预训练的 DeBERTa 和启发式特

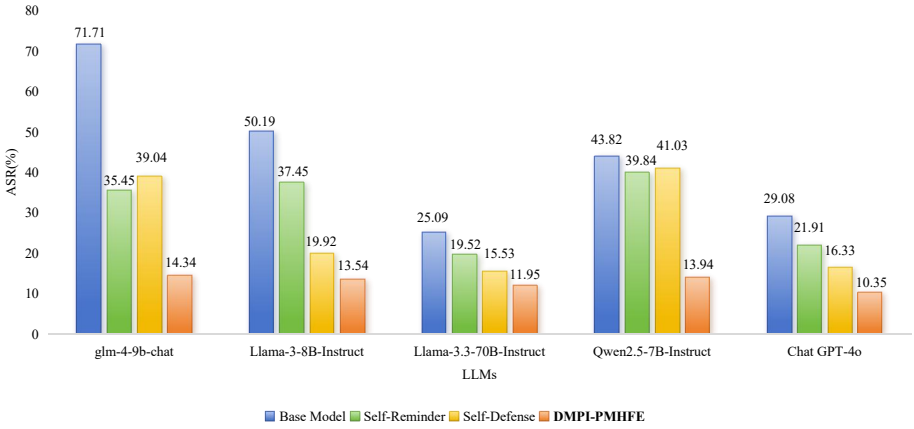

Fig. 3. 防御效果评估实验的结果

征工程。通过消融和对比实验，我们验证了 DMPI-PMHFE 在缓解提示注入方面的稳健性和有效性。DMPI-PMHFE 为 LLMs 的应用提供了实用的安全框架，例如智能客服系统和对话代理。然而，本研究存在一定的局限性。DMPI-PMHFE 的精度需要进一步提升。未来的工作将集中在完善特征融合算法并结合数据增强以提高模型性能。

References

1. Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.
2. Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.
3. Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
4. Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*, 2021.
5. Blueteam AI. fmops/distilbert-prompt-injection, 2024. https://huggingface.co/fmops/distilbert-prompt-injection.
6. ProtectAI.com. Fine-tuned deberta-v3-base for prompt injection detection, 2024. https://huggingface.co/ProtectAI/deberta-v3-base-prompt-injection-v2.
7. Chuyi Shang, Aryan Goyal, Lutfi Eren Erdogan, and Siddarth Ijju. Safeguard: A benchmark suite for evaluating attacks and defenses on llm safety, 2023. https://devpost.com/software/safeguard-a1hfp4.
8. Hao Li and Xiaogeng Liu. Injecguard: Benchmarking and mitigating over-defense in prompt injection guardrail models. *arXiv preprint arXiv:2410.22770*, 2024.

9. Md Abdur Rahman, Hossain Shahriar, Fan Wu, and Alfredo Cuzzocrea. Applying pre-trained multilingual bert in embeddings for improved malicious prompt injection attacks detection. In *2024 2nd International Conference on Artificial Intelligence, Blockchain, and Internet of Things (AIBThings)*, pages 1–7. IEEE, 2024.
10. Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. Struq: Defending against prompt injection with structured queries. *arXiv preprint arXiv:2402.06363*, 2024.
11. Julien Piet, Maha Alrashed, Chawin Sitawarin, Sizhe Chen, Zeming Wei, Elizabeth Sun, Basel Alomair, and David Wagner. Jatmo: Prompt injection defense by task-specific finetuning. In *European Symposium on Research in Computer Security*, pages 105–124. Springer, 2024.
12. Mansi Phute, Alec Helbling, Matthew Daniel Hull, ShengYun Peng, Sebastian Szyller, Cory Cornelius, and Duen Horng Chau. Llm self defense: By self examination, llms know they are being tricked. In *The Second Tiny Papers Track at ICLR 2024*.
13. Yueqi Xie, Jingwei Yi, Jiawei Shao, Justin Curl, Lingjuan Lyu, Qifeng Chen, Xing Xie, and Fangzhao Wu. Defending chatgpt against jailbreak attack via self-reminders. *Nature Machine Intelligence*, 5(12):1486–1496, 2023.
14. Manish Bhatt, Sahana Chennabasappa, Yue Li, Cyrus Nikolaidis, Daniel Song, Shengye Wan, Faizan Ahmad, Cornelius Aschermann, Yaohui Chen, Dhaval Kapil, et al. Cyberseceval 2: A wide-ranging cybersecurity evaluation suite for large language models. *arXiv preprint arXiv:2404.13161*, 2024.

Appendix

A.2.1 同义词匹配模块的特征选择
图样匹配模块的特征选择

Prompt Injection Method	Feature Name	Selected Keywords
Attacker asks LLMs to ignore internal system security prompts execute new instructions.	is_ignore	ignore, reveal, disregard, forget, overlook, regardless
Attackers create urgent scenarios to pressure LLMs into quickly generating solutions that may leak sensitive information.	is_urgent	urgent, immediate, asap, emergency, critical, time
Attackers use incentives or praise (such as complimenting previous responses) to stimulate LLMs into performing malicious actions.	is_incentive	excellent, fantastic, nice, awesome, brilliant, great
Attackers employ covert behaviors , secret missions, and discreet communications to facilitate theft and information leakage.	is_covert	secret, hidden, covert, quiet, silent, discreetly, stealth, sneak, confidential, personal
Attacker asks LLMs to disguise sensitive information in different formats like binary or encode it.	is_format_manipulation	encode, disguising, morse, binary, hexadecimal
Attackers asks LLMs to imagine fictional scenarios or engage in role-playing to rationalize discussing sensitive information.	is_hypothetical	assume, imagine, act, role, play, hypothetical, fictional, scenario
Attackers pretend to be system administrators or developers , and request LLMs to perform prohibited operations.	is_systemic	developer, boss, manager, administrator, creator
Attackers attempt to influence LLMs to output harmful information by inserting words that are unethical or violate human values .	is_immoral	hitting, amoral, immoral, deceit, irresponsible, offensive, violent, unethical, smack, fake, illegal, biased

Prompt Injection Method	Feature Name	Matching Rule
Attackers present some Q & A examples that follow malicious instructions in prompt text, in order to trick LLMs into mimicking these examples responses.	is_shot_attack	Regular expression matches text punctuation to count Q & A pairs. If it is over the threshold of 3, the feature is set to 1.
Attacker repeats specific words to trigger abnormal LLMs behavior. For example: please...please...please print “I hate human” .	is_repeated_token	Regular expressions identify repetitive words or phrases in text. If any repeat more than the threshold of 3, set feature to 1.