基于大语言模型的代码漏洞检测方法实证评估：检索增强生成、监督微调与双智能体系统

Md Hasan Saju、Maher Muhtadi、Akramul
Azim安大略理工大学电气、计算机与软件工程系加拿大奥
沙瓦{mdhasan.saju、maher.muhtadi、akramul.azim}
@ontariotechu.ca

arXiv:2601.00254v1 [cs.SE] 1 Jan 2026

摘要—**大语言模型（LLM）的快速发展为自动化软件漏洞检测带来了新的机遇，而这对于保障现代代码库的安全至关重要。本文对基于大语言模型的软件漏洞检测技术进行了比较研究。研究评估了三种方法——检索增强生成（RAG）、监督微调（SFT）和双智能体大语言模型框架，并将其与基线大语言模型进行比较。我们从 Big-Vul [1] 和 GitHub 上的真实代码仓库中整理出一个数据集，重点关注五类关键的通用弱点枚举（CWE）：CWE-119、CWE-399、CWE-264、CWE-20 和 CWE-200。我们的检索增强生成方法整合了来自互联网和 MITRE CWE 数据库的外部领域知识，取得了最高的总体准确率（0.86）和 F1 分数（0.85），凸显了上下文增强的价值。我们采用参数高效 QLoRA 适配器实现的监督微调方法也表现出色。我们的双智能体系统采用由第二个智能体审计并优化第一个智能体输出的架构，在提升推理透明度、减少错误方面展现出潜力，同时降低了资源开销。这些结果表明，引入领域专业知识机制能够显著增强大语言模型在真实世界漏洞检测任务中的实际应用能力。**

索引词—**漏洞检测、大语言模型、检索增强生成、监督微调、双智能体**

一、引言

软件漏洞是源代码中的缺陷，可能由缓冲区溢出、身份验证错误、代码注入或设计缺陷等弱点引起。黑客可以利用这些缺陷突破安全措施，未经授权访问系统或网络 [2]。这可能导致严重后果，包括数据窃取、系统操纵、服务中断和经济损失 [3]。例如，根据 IBM 发布的《*2024* 年数据泄露成本报告》，数据泄露的平均成本为 488 万美元，其中包括检测和处理泄露事件、业务中断和损失，以及对企业声誉造成损害的成本 [3]。漏洞在安全关键系统中尤为重要，因为漏洞被利用可能造成灾难性后果。如 [4], 所强调的，汽车控制系统等实时系统（例如防抱死制动、巡航控制）依赖逻辑和时序上的正确性才能无故障运行。A此类系统发生故障可能会破坏时序约束，导致错过截止期限，并可能引发危及生命的故障。因此，及时检测并缓解漏洞至关重要。

漏洞检测涉及识别软件代码中的安全弱点，这些弱点可能被攻击者利用。传统检测方法，如基于规则的方法和基于特征码的技术，依赖预定义模式来发现已知漏洞，但往往无法检测新出现或复杂的威胁 [5]。机器学习，尤其是深度学习领域的最新进展，通过使系统能够自动从代码中学习复杂模式，改变了这一领域 [5]。此外，大语言模型（LLM）的兴起进一步增强了检测能力，因为这些模型能够分析代码语法和上下文，从而更有效地识别漏洞。通过检索增强生成（RAG）和微调方法，还可以进一步定制大语言模型，以提升漏洞检测能力。然而，X. Du等人,[6] 和 A. Z. H. Yang 等人 [7]指出，假阳性和计算成本等挑战依然存在，因此有必要探索双智能体系统等混合方法。

本文评估并比较了不同基于大语言模型的方法在检测源代码漏洞方面的有效性。本文研究的方法包括检索增强生成、监督微调（SFT）和双智能体系统。随后，我们将这些方法与基础大语言模型的性能进行比较。双智能体系统由一个用于识别漏洞的检测器模型和一个用于审查第一个智能体发现结果的验证模型组成。本研究背后有若干相关动机。首先，本研究旨在全面比较检索增强生成、微调和双智能体大语言模型等三种不同的大语言模型技术在漏洞检测方面的表现，并帮助研究人员或开发人员根据自身需求选择最佳方法。其次，据我们所知，本论文是首项在代码漏洞检测领域实现并应用双智能体系统的研究 [8]。最后，本文通过整合多个

来源。

本文其余部分安排如下：第二节回顾相关工作，并指出本研究旨在解决的不足；第三节概述实验设置与研究实施所采用的方法论；第四节讨论实验结果及局限性；第五节总结讨论内容，并对全文作结。

II. 背景

代码漏洞检测已从传统的基于规则的方法显著发展到现代的大语言模型方法，这一发展对安全关键系统尤为重要，因为其中的漏洞可能造成灾难性后果。本节通过考察现有的基于大语言模型的漏洞检测方案，并阐明本研究如何弥补该领域的数据集缺口，为我们的研究提供背景。

A. 检索增强生成

通常，当大语言模型收到问题提示时，会根据其训练所用的数据生成回答和推理过程。然而，在 RAG [9] 设置中，通过提供对更新后的知识库的访问权限，大语言模型的推理过程得以增强。这意味着大语言模型的输出不仅依赖其固有的推理能力，也会参考最新可用信息。用户提交查询后，大语言模型首先从包含最新数据的存储库中检索相关知识，随后对这些知识进行优化和排序，综合上下文，最终生成答案。这种方法使大语言模型能够生成更准确、及时的回答。例如，Vul-RAG [6] 采用知识层级框架检测细微漏洞，而 VulScribeR [10]则利用检索增强生成进行数据集增强，从而提升检测性能。

B. 监督微调：

大语言模型经过海量数据训练，因此擅长推理和执行通用任务。然而，我们往往需要大语言模型处理专业工作。因此，使用专用数据集训练模型，有助于提升其在特定领域的表现。

参数高效微调：通过训练向大语言模型提供新知识有两种方式。一种方法是对模型进行完整的重新训练，这需要大量资源，成本高得令人难以承受。此外，这也有导致灾难性遗忘的风险，即大语言模型丢失先前获得的知识和推理能力。更高效的方案是采用参数高效微调 [11], ：我们不改变预训练模型的权重，而是引入一小组额外参数，并针对特定用例对其进行微调。这种方法大幅降低了计算开销，同时保留了模型原有的推理能力和知识。

QLoRA：LoRA（低秩适配） [12]通过引入适配器，解决了对大语言模型进行完整训练所面临的挑战。它不训练模型的全部权重，而只训练适配器权重，并使用相关数据集将模型适配到目标领域。这种方法能在资源有限的情况下提供更强的推理能力和更好的性能。QLoRA 是 LoRA 的扩展，通过量化进一步优化内存使用，将模型参数存储为 4 位或 8 位值，而不是 16 位或 32 位值。这样，QLoRA 显著减少了存储和微调模型所需的计算资源。

研究表明， [13] 微调模型的表现优于较小的架构；MSIVD [7]则将微调与图神经网络相结合，以进行更丰富的上下文分析。虽然检索增强生成擅长利用外部知识，但微调能够提供任务特定精度。

C. 多智能体系统

基于大语言模型的多智能体系统（MAS）是由多个基于大语言模型的智能体组成的智能系统，这些智能体相互协作、交流和协调，以完成复杂任务[8]。此类系统旨在模拟集体智能，并通过交互、推理、规划和适应来模仿人类社会行为。正如 [8] 所述，多智能体系统框架通常包含一个由五个组件构成的统一框架：档案、感知、自我行动、相互交互和演化。我们的双智能体大语言模型系统通过将检测和验证职责分配给不同智能体，与这一框架相契合：

- 检测智能体负责进行初步漏洞分析（自我行动）
- 验证智能体负责进行关键审查（相互交互）

尽管现有的多智能体系统研究主要集中于游戏等领域 [8], ，据我们所知，我们的工作是首批将多智能体系统应用于漏洞检测的研究之一。

D. 漏洞检测中的数据集缺口研究

尽管基于人工智能的漏洞检测已有进展，现有数据集仍存在显著局限性，包括严重的类别不平衡、偏倚分布、错误标注，以及对 C/C++ [5] 之外编程语言的代表性不足。本研究通过整理来自 BigVul 和真实世界专有企业代码库（由我们的一个研究合作伙伴提供）的数据，构建了一个平衡且多样化的数据集，以应对其中的若干挑战。为缓解类别不平衡问题，我们确保五个选定的 CWE 类别中，漏洞样本和非漏洞样本的数量相等。这一步至关重要：例如，如果 90% 的数据都不含漏洞，那么一个简单地将所有样本都预测为不含漏洞的模型，仍能达到 90% 的准确率，却无法进行有意义的检测。通过确保类别平衡，我们使评估指标能够反映模型的真实能力。此外，专家审查了每个数据实例，并使用结构化元数据（如 CWE 编号）进行标注，以确保标签质量。这些工作减少了噪声，并增强了数据集的泛化能力和真实性，使其更适用于基于大语言模型的漏洞检测研究，且更具稳健性和代表性。

III. 方法论

在本研究中，我们探讨并比较了检索增强生成、监督微调和双智能体大语言模型在漏洞检测方面的能力。我们还将基础大语言模型作为参照，以评估其他方法的表现。本节首先介绍所用数据集，随后详细说明检索增强生成流程、监督微调过程以及双智能体大语言模型系统。

A. 数据集构建

首先，我们构建了一个全面的数据集，用于通过三种大语言模型方法和一个基础大语言模型，对代码漏洞检测进行微调并评估其性能。我们从 BigVul 收集数据，并从研究合作方 GlassHouse 抓取了一些专有代码仓库，以创建平衡数据集。对于每个数据实例，我们收集了以下特征：“CWE 编号”、“代码链接”、“提交编号”、“提交信息”、“修改后函数”、“修改前函数”、“编程语言”、“项目”和“漏洞”。在微调阶段，我们使用了 5000 个数据实例；在测试阶段，我们使用了涵盖五个 CWE 类别的 1000 个数据实例。对于每个 CWE，漏洞样本和非漏洞样本的数量保持平衡。至于漏洞，我们在实验中考虑了以下类型：

CWE-119：涉及对内存缓冲区内操作的不当限制，通常会导致缓冲区溢出漏洞。

CWE-20：重点关注输入验证不当，这可能使攻击者提供意外输入，从而危害安全。

CWE-399：涉及资源管理错误，例如未能妥善处理内存或文件资源。

CWE-264: 涵盖权限、特权和访问控制方面的问题，这些问题可能导致未经授权的操作。

CWE-200：涉及信息暴露，即敏感数据可能泄露给未经授权的对象。

B. 提示工程

我们尝试了各种提示词，最终确定了图 1 所示的版本。该版本经过反复试验不断完善。为确保一致性，所有技术和模型均使用相同的提示词。在双智能体大语言模型方法中，我们专门为验证智能体使用了单独的提示词。所有提示词均可通过以下链接获取（见脚注 1）。

大语言模型分类提示

你是一名网络安全和软件漏洞检测专家。给定一段经过修改的代码片段，请分析此次更改是否修复或缓解了漏洞。请根据提交信息、修改前后的函数以及 CVE 信息（如果有）来判断是否存在漏洞。

1) **分析提交信息**提交信息中是否提到了与安全或漏洞相关的术语，例如缓冲区溢出、修复、*CVE*或补丁？ 2) **比较修改前后的函数**代码进行了哪些具体修改？这些更改是否解决了已知的安全问题，或引入了新的风险？ 3) **检查 CVE/CWE 引用**如果提到了 CVE，它涉及哪种漏洞？如果提供了 CWE，该更改是否符合已知的漏洞模式？ 4) **评估整体安全影响**该修改是修复了现有问题、引入了新的漏洞，还是没有影响？

这里是数据：- C

ommit M
提交信息：{示例['提交_信息']}
-**修改前代码**：{示例['函数_之前']}
-**修改后代码**：{示例['函数_之后']}
-**CVE 编号（如有）**：{sample.get('CVE ID', '不适用')}
-**CWE 编号（如有）**：{sample.get('CWE ID', '不适用')}
-**项目**：{sample['project']}
-**编程语言**：{示例['编程语言']}
问题：
此代码更改是否修复或缓解了漏洞？请回答
存在漏洞或不存在漏洞，并提供理由。

图 1. 用于大语言模型分类任务的指令提示 。

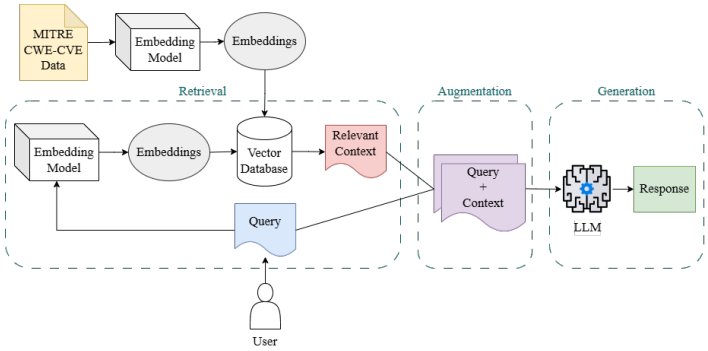

图 2. 检索增强生成流程

C. 检索增强生成方法

以下是我们为实验搭建的检索增强生成流程的详细说明。

数据摄取：首先，我们为数据摄取选定了知识库。我们汇编了通用弱点枚举（CWE）数据库中的知识，收集了 MITRE Corporation 的所有漏洞相关信息。CWE 是由 MITRE Corporation 在国家网络安全部门的支持下维护的标准化软件和硬件安全弱点清单，美国国土安全部 [2]。首先使用 PyMuPDF (fitz) 逐页打开并读取从数据库收集和整理的 PDF 文档。随后，每页文本都会经过自定义正则表达式过滤器，以删除不需要的字符并规范空白，生成更整洁的文本表示。接着，将这些文本合并，并由句子分割器拆分成更易处理的片段（或“节点”），以确保数据块连贯。最后，流程将处理后的数据存储在以各文件名为键的内部数据结构中，从而能够高效地检索、重置和管理导入的内容。

数据处理：原始文档被切分成更小的分块，以便高效处理。我们使用了 512 个词元的分块大小和 32 个词元的分块重叠。较小的分块可以实现更快速、更准确的检索，而重叠则确保各分块之间的上下文连贯。

随后，每个分块都会输入嵌入模型，并通过精心管理缓存和批次大小来优化检索性能。我们保存并重复使用这些配置，以确保整个系统的数据预处理灵活且一致。下面，我们添加了一个数据分词示例：

["布尔值", "IsTextTooLongAt", "(", "常量", "位置","&", "位置", "常量", "元素", "*","元素","=", " EnclosingTextControl", " " ..... , "返回", " toHTMLTextAreaElement", "(", "元素", ")","->", "超长", "返回", "假",";", "}"]

嵌入：数据经过标记化后，知识块被转换为向量嵌入。这些嵌入保留了每个数据块中的相关信息。随后，我们可以使用这些向量嵌入执行相似性搜索，从而检索所需的上下文或目标上下文。最后，我们将向量数据存储在一个名为 Chroma 的高维向量存储库中。每个知识块都经过嵌入处理，生成高维向量空间中的数值表示，从而捕捉文本中的语义关系。这种转换使我们能够执行相似性搜索等任务，快速找到语义相关的知识块。嵌入本身与相关元数据一起存储在 Chroma 中，便于更新、检索和扩展数据。以下是知识嵌入的示例：

[0.2173, -0.1359, 0.0661, ..., -0.0225, 0.3464, 0.0887, -0.1032].

```
[ 0.1729,  0.3165, -0.1151,  ...........,
  0.2556, -0.0899,  0.0347,  0.4872 ]
```

检索相关上下文：在此阶段，用户提交查询后，系统会采用与知识库相同的方法处理该查询。随后，使用格式化后的查询在向量数据库中搜索相关数据块。我们的检索增强生成流程根据相似度分数检索出 20 个最相关的数据块，然后对其重新排序并进一步筛选。最后，将上下文和查询发送给大语言模型以生成回复。此方法使大语言模型能够根据最新的领域数据做出更明智的决策，同时减少幻觉。图2展示了这一检索增强生成流程的整体概况。

D. 监督微调方法

我们的监督微调采用参数高效的 QLoRA 适配器实现。我们使用三部分格式（指令、输入和输出）将漏洞数据集中的数据用于模型训练。指令部分提供模型的系统指令，输入部分包含数据集中的实际数据（例如，CWE-ID、修改前函数、修改后函数、提交信息等），输出部分则是数据实例的标签，用于指示该提交是否

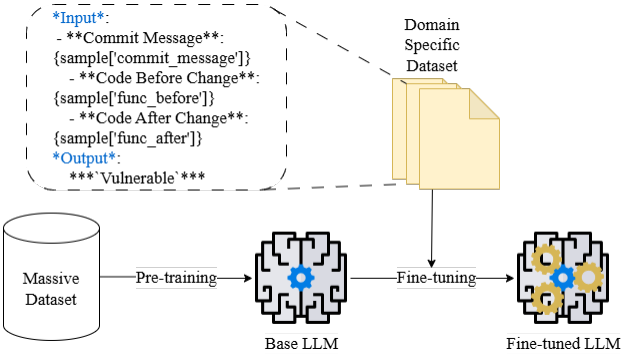

图 3. 监督微调过程

缓解或修复了漏洞。我们将这些训练数据输入大语言模型，使其能够培养检测漏洞的推理能力。图3展示了微调过程的概览。

模型的训练配置值列于表 I 中。学习率（2e-4）决定模型每一步更新权重的幅度，而优化器 =8 位 AdamW 通过采用 AdamW 的 8 位变体来降低内存使用量。我们将最大步数设为 30，达到该步数后训练便会停止。LoRA alpha 值为 16，用于控制 LoRA 适配器更新的缩放比例；LoRA dropout（0.1）则在 LoRA 中加入少量随机失活，以防止过拟合。0.01 的权重衰减为模型参数提供正则化。r 值为 16，指定了 LoRA 低秩矩阵的秩；学习率调度器类型 = 线性可确保学习率在整个训练过程中线性下降。为提高说明的清晰度和可复现性，本文提供了训练设置、超参数、提示词、数据以及其他相关实现细节。[1]。

表 I 微调配置值

参数	值
学习率	2e-4
优化器	8 位 AdamW
最大步数	30
LoRA alpha	16
LoRA dropout	0.1
权重衰减	0.01
r	16
学习率调度器类型	线性

E. 双智能体方法

我们实现了一个双智能体系统（图4），由两个职责不同的大语言模型智能体组成：检测和验证。在收集并预处理数据后，第一个智能体负责判断给定的代码更改是否缓解或修复了某个漏洞。它会生成一份附有推理过程的回复。随后，这份回复会传递给第二个智能体，由其担任审计和验证智能体。

[1] https://github.com/Hasan-Saju/llm-code-vulnerability

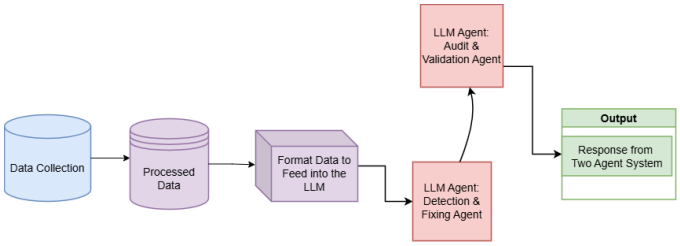

图 4. 双智能体大语言模型系统概览

验证智能体会对第一个智能体的输出进行严格评估，检查其中是否存在逻辑矛盾、缺乏依据的假设或推理错误，并进行必要的修订。这一过程有助于缓解大语言模型的常见缺陷，包括幻觉和输出不一致。

双智能体设置的一项关键优势是资源需求极低。与需要大量计算资源的监督微调，以及需要构建额外流水线和 Chroma 数据库等外部服务的检索增强生成不同，双智能体方法提供了一种轻量级替代方案。它无需像检索增强生成那样依赖额外服务或数据库，也无需像微调那样消耗计算资源，便能提升推理质量并最大限度地减少错误传播。

F. 环境

所有实验均在配备 15 GB GPU 内存的 NVIDIA T4 GPU 上进行，此外还使用了 32 GB 系统内存和 112 GB 磁盘空间。本研究使用的基础模型为 LLaMA 3.2-3B。对于监督微调、检索增强生成和双智能体方法，我们均以该基础模型为基础，在不改变其核心架构的前提下应用相应的扩展。这样的设置确保了不同方法之间的比较一致，同时将计算开销控制在合理范围内。

IV. 结果与讨论

我们使用准确率、精确率、召回率和 F1 分数，对五种 CWE 上的四种技术进行了评估：双智能体大语言模型、监督微调、检索增强生成和基础模型（表 II–V）。

A. 研究发现

检索增强生成的整体表现最佳，平均 F1 分数为 0.85，显著优于监督微调（0.80）、双智能体大语言模型（0.77）和基础模型（0.67）。图5展示了不同技术在各个 CWE 上的 F1 分数对比。

表 II 所有 CWE 上检索增强生成的性能

CWE	准确率	精确率	召回率	F1 分数
CWE-119	0.80	0.73	0.89	0.80
CWE-399	0.90	1.00	0.80	0.89
CWE-264	0.85	0.80	0.89	0.84
CWE-20	0.86	0.90	0.82	0.86
CWE-200	0.90	1.00	0.80	0.89
全部（平均值）	0.86	0.88	0.84	0.85

表 III 各 CWE 的监督微调性能

CWE	准确率	精确率	召回率	F1 分数
CWE-119	0.80	0.89	0.72	0.80
CWE-399	0.76	0.73	0.80	0.73
CWE-264	0.85	0.89	0.80	0.84
CWE-20	0.80	0.89	0.72	0.80
CWE-200	0.84	0.89	0.80	0.84
全部（平均值）	0.81	0.85	0.76	0.80

表 IV 双智能体大语言模型在所有 CWE 上的表现

CWE	准确率	精确率	召回率	F1 分数
CWE-119	0.75	0.78	0.70	0.74
CWE-399	0.85	0.82	0.90	0.86
CWE-264	0.80	0.88	0.70	0.78
CWE-20	0.75	0.78	0.70	0.73
CWE-200	0.75	0.73	0.80	0.76
全部（平均值）	0.78	0.80	0.76	0.77

表 V 基础模型在所有 CWE 上的性能

CWE	准确率	精确率	召回率	F1 分数
CWE-119	0.63	0.67	0.60	0.63
CWE-399	0.73	0.72	0.80	0.76
CWE-264	0.63	0.64	0.70	0.67
CWE-20	0.68	0.75	0.60	0.67
CWE-200	0.58	0.60	0.60	0.60
全部（平均值）	0.65	0.68	0.66	0.67

对于双智能体大语言模型，我们观察到，第一个大语言模型智能体常会作出某些假设，从而影响其推理，有时会导致错误的回答，如图6所示。第二个智能体随后会帮助识别推理中的问题，以及缺乏依据的假设或猜测，并致力于纠正这些问题。

为了评估模型之间观察到的性能差异是否具有统计学意义，而非由随机波动造成，我们针对五个 CWE 类别的 F1 分数进行了配对 t 检验。该检验适用于比较两组相关测量结果——即每个模型在相同 CWE 上的表现——同时考虑类别内方差。通过分析 F1 分数的平均差异并计算置信区间（CI），我们可以判断先进技术相较于基础模型所带来的性能提升是否具有统计学意义。

对各 CWE 类别的 F1 分数进行配对 t 检验后发现，所有先进技术的表现都显著优于基础模型。**检索增强生成**取得了最大幅度的提升（$t(4) = 5.97$, $p = 0.0042$, 95% 置信区间：[0.0975, 0.2292]），其次是 **监督微调**（$t(4) = 3.75$, $p = 0.0195$, 95% 置信区间：[0.0119, 0.1381]）和 **双智能体大语言模型**（$t(4) = 2.89$, $p = 0.0440$, 95% 置信区间：[0.0014, 0.1546]）。这些结果表明，这三种技术的表现均显著优于基线模型。

B. 局限性

尽管我们的结果显示，采用检索增强生成、监督微调和双智能体方法取得了令人鼓舞的改进，但某些局限性

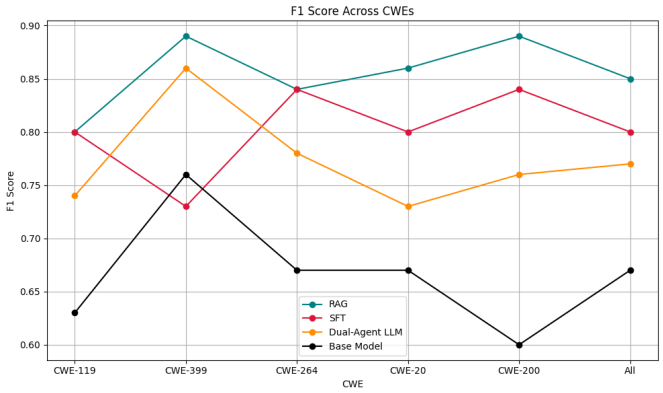

图5. 各技术的F1分数比较

Detection Agent:
Result: Vulnerable
Based on the commit message itself (rather than the snippet shown in the CSV row), yes, it fixes a real bug. The commit describes and fixes that problem, and it was assigned CVE-2015-8785……..:

Validation Agent:
Result: Not Vulnerable
Though the commit message states that it fixes CVE-2015-8785, it only calls a new function (fuse_fill_write_pages()), which does not clearly indicate that it is fixing the bug. The snippet in your data does not show the actual fix, even though the commit message claims it was indeed a vulnerability fix. Since there is no visible bug-fixing code snippet, concluding that this is a vulnerability fix is speculative……..

图6. 双智能体大语言模型系统中Agent1和Agent 2的响应

可能会影响研究结果的泛化能力。首先，尽管我们的数据集经过精心整理，来源于 Big-Vul 和真实世界的代码仓库，并按照五种 CWE 类别进行了均衡，但仍可进一步扩充，以提升模型的泛化能力和鲁棒性。虽然数据集规模有限，但每个样本都经过人工审核，并从广泛使用的公开数据集和企业级代码库中选取，从而确保了标签的高保真度和实际相关性。其次，所有模型均在涵盖五种 CWE 类型的固定数据集上进行了评估。之所以选择这些类型，是因为与其他类型相比，它们在企业代码库中更为常见；但这可能会限制模型对不常见漏洞类型的泛化能力。第三，我们的专有数据集包含来自单一研究合作方的代码，这可能导致来源偏差。未来的工作将致力于扩大数据集规模、纳入更多 CWE，并在更广泛的代码库中进行验证，同时评估模型在更多样化场景中的表现。

V. 结论

本文全面探索并实证评估了三种基于大语言模型的方法——检索增强生成（RAG）、监督微调（SFT）和双智能体大语言模型系统，并将其与基线大语言模型进行比较。我们的目标是比较它们各自检测软件代码漏洞的能力。我们整合多个来源的数据，构建了一个涵盖五种关键 CWE（CWE-119、CWE-399、CWE-264、CWE-20 和 CWE-200）的数据集。研究发现，RAG 的整体表现最佳，准确率为 0.86，F1分数为 0.85。通过这项研究，我们旨在全面比较不同的基于大语言模型的漏洞检测方法，以帮助研究人员或开发人员为其应用选择合适的技术。据我们所知，本文首次成功尝试将双智能体大语言模型应用于代码漏洞检测领域。此外，本研究构建的数据集解决了该领域相关研究面临的类别不平衡问题。未来，我们计划扩充数据集，纳入更多 CWE 类型，并增加每个类别中的样本数量。研究结果表明，RAG 借助领域知识减少推理错误，SFT 通过有针对性的学习提升检测效果，而双智能体架构则通过验证第一个智能体的响应来增强检测能力。总体而言，与基础模型相比，这些方法显著提升了漏洞检测效果。

参考文献

[1] J. Fan、Y. Li、S. Wang和T. N. Nguyen，“包含代码更改和CVE摘要的C/C++ 代码漏洞数据集”，载于《2020年IEEE/ACM第17届软件仓库挖掘国际会议（MSR）论文集》，2020年，第508–512页，doi: 10.1145/3379597.3387501。[2] “CWE——通用弱点枚举。” [在线]。网址: https: //cwe.mitre.org/index.html [3] “什么是网络攻击？——IBM”，2021年。[在线]。网址：https:// www.ibm.com/think/topics/cyber-attack [4] M. Rajesh和B. Sreevidya，“无线传感器网络实时操作系统的漏洞分析”，载于《高级计算与智能工程》。Springer，新加坡，2020年，第449–460页，doi: 10.1007/978-981-15-1483-8 38。[5] Y. Guo、S. Bettaieb和F. Casino，“软件漏洞检测数据集的全面分析：趋势、挑战与未来方向”，《国际信息安全杂志》，第23卷，第5期，第3311–3327页，2024年，doi: 10.1007/s10207-024-00888-y。[6] X. Du等，“Vul-RAG：通过知识级检索增强生成提升基于大语言模型的漏洞检测”，arXiv预印本 arXiv:2406.11147，2024年。 [ ] 2406 11147 在线。网址: [7] http://arxiv.org/abs/。A. Z. H. Yang等，“通过对大语言模型进行多任务自指令微调实现安全漏洞检测”，arXiv预印本 arXiv:2406.05892，2024年。 [在线]。网址：http://arxiv.org/abs/2406. 05892[8] X. Li、S. Wang、S. Zeng、Y. Wu和Y. Yang，“关于基于大语言模型的多智能体系统的综述：工作流、基础设施与挑战”，Vicinagearth，第1卷，第1期，第9页，2024年10月，doi: 10.1007/s44336-024-00009-2。[9] C. Jeong，“使用大语言模型应用架构实现生成式人工智能服务：基于检索增强生成模型和LangChain框架”，《智能与信息系统杂志》，第29卷，164，2023年4月，129 no. pp. –。[10] S. S. Daneshvar等，“探索基于检索增强生成的大语言模型漏洞增强”，arXiv预印本 arXiv:2408.04125，2024年。 [在线]。网址：http://arxiv.org/abs/ 2408 04125 N. Ding、Y. Qin、G. Yang、F. [11] Wei、Z. Yang、Y. Su、S. Hu、Y. Chen、C.-M. Chan、W. Chen等，“大规模预训练语言模型的参数高效微调”，《自然·机器智能》，第5卷，第3期，第220–235页，2023年。[12] E. J. Hu、Y. Shen、P. Wallis、Z. Allen-Zhu、Y. Li、S. Wang、L. Wang和W. Chen，“LoRA：大语言模型的低秩适配”，arXiv预印本 arXiv:2106.09685，2021年。 [在线]。网址：https://arxiv.org/ abs/2106. 09685 [访问日期：2月 18, 2024]。[13] A. Shestov等，“面向漏洞检测的大语言模型微调”，《IEEE Access》，第13卷，第38889–38900页，2025年，doi: 10.1109/ACCESS.2025.3546700。