**开甲书院“追光”科研启蒙项目**

**申 报 书**

项目名称： 基于大语言模型的程序开发

项目负责人： 刘功泽

项目依托学科： 人工智能，软件工程

申请日期：

项目来源：教师指导选题（新生导师）☑

教师指导选题（科学之光）□

自主立题 □

南京大学新生学院开甲书院 制

**填写说明**

一、填写申报书前，请先仔细查阅申报通知。

二、申报书要按照要求，逐项认真填写，填写内容必须实事求是，表达明确严谨。

三、格式要求：申报书中各项内容以Word文档格式填写，表格中的字体为小四号仿宋体，1.5倍行距；表格空间不足的可扩展。

四、申报书由指导教师审查，签署意见后，由项目负责人统一提交。

1. **项目简介**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 项  目  概  况 | 项目名称 | | 基于大语言模型的程序开发 | | | | | |
| 项目性质 | | （）基础研究 （√）应用研究 | | | | | |
| 项目来源 | | （）自主立题 （√）教师指导选题 | | | | | |
| 起止时间 | | 自 2025 年 2 月 至 年 月 | | | | | |
| 项  目  负  责  人 | 姓名 | 刘功泽 | 性别 | 男 | 出生年月 | 2006.08 | 学历 | 本科生 |
| 专业 | 信息与计算科学 | | 联系  电话 | 13966494979 | | 邮箱 | 1062979101@qq.com |
| 项 目 组  主要成员 | | 姓 名 | | 联系电话 | 专业 | | 具体分工 | |
| 刘功泽 | | 13966494979 | 信息与计算科学 | |  | |
| 陈一沛 | | 17602191804 | 软件工程 | |  | |
| 马修齐 | | 17749554816 | 信息与计算科学 | |  | |
|  | |  |  | |  | |
|  | |  |  | |  | |
|  | |  |  | |  | |
|  | |  |  | |  | |
| 指  导  教  师 | | 姓名 | |  | 性 别 |  | 出生年月 |  |
| 职称 | |  | 联系电话 | |  | |
| 主要研究方向 | |  | | | | |
| 姓名 | |  | 性 别 |  | 出生年月 |  |
| 职称 | |  | 联系电话 | |  | |
| 主要研究方向 | |  | | | | |

1. **立项依据**

|  |
| --- |
| 1. 研究背景、目的及意义   **研究背景**  随着自然语言生成技术的发展，大语言模型在人工智能领域的应用越来越广泛。在软件工程中，以自然语言生成为核心的大语言模型技术正在掀起热潮，如 ChatGPT 等模型已经在软件开发中展示出一定的潜力。然而，这些模型生成的代码存在可信性问题，需要研究者进一步研究如何驾驭这些工具，从自然语言需求出发，构建程序详细设计原型，并生成程序具体实现。  **研究目的**  本研究旨在探索主动逐步精化式 IDE 的设计与实现技术，提高软件开发效率和质量，降低程序员修正预测性代码生成错误的难度，增强用户对生成代码的信任度。  **研究意义**  本研究有助于深入理解自然语言与程序代码之间的关系，丰富软件工程领域的理论研究，为软件开发提供新的理论支持。同时，通过该研究，可以开发出更高效、更可靠的软件开发工具，提高软件开发团队的生产力，减少软件开发过程中的错误和风险。   1. 研究内容 2. 探究当前主流大模型的代码能力；分析集成开发环境中当前主流AI代码编辑插件的表现 3. 基于对已有AI代码编辑技术的问题分析，构建借助伪代码作为自然语言与程序代码的中间桥梁，实现主动逐步精化的编程新范式，完成对人机交互设计、伪代码粒度选择等核心问题的研究 4. 基于以上新范式协助开发一款能实现主动逐步精化的AI代码编辑插件 5. 国内外研究现状和发展动态   基于大模型的辅助编程工具通过代码生成、软件质量保证和软件维护重塑软件开发，显著提升生产力和代码质量。但挑战如代码准确性、安全性和伦理问题仍需解决。未来研究应聚焦优化方法、减少偏见和提升公平性，以充分发挥 LLMs 在软件开发中的作用。  **代码生成和程序合成**  工具如GPT-4和Codex在代码生成和程序合成中表现突出。研究表明，GPT-4在HumanEval数据集上的通过率超过50%，它们能够高效自动化常规编码任务。举例来说，GitHub Copilot基于Codex提供实时代码建议，显著减少了开发时间。  **软件质量保证**  这些工具在漏洞检测和测试生成中表现出色。研究显示，调整后的 LLM 可以准确识别代码中的安全漏洞，并从自然语言规范生成测试用例，确保软件稳定性。这对提高软件质量至关重要，尤其在安全敏感领域。  **软件维护**  在软件维护中，工具如 ChatGPT 等帮助修复程序和检测代码克隆。它们能生成潜在的错误修复方案，并通过交互式反馈协助开发者，简化维护流程。这对长期项目管理特别有用。  **IDE 集成**  这些工具通常作为插件集成到 IDE 中，如 GitHub Copilot，显著提升开发者的生产力和代码质量。研究显示，它们能减少编码时间，让开发者专注于更高层次的设计，但也可能生成不准确或不安全的代码，需进一步验证。  **不足之处**  **人机交互与代码理解方面交互形式效率低**  未能设计出高效的人机交互形式，导致开发者在分析和理解预测性代码时面临困难，增加了代码理解的难度与复杂性，难以有效地对代码进行修改和确认。  **伪代码生成方面粒度选择不合理**  在设计伪代码粒度时存在困难，无法在保持伪代码可读性的同时，为代码生成提供足够详细的信息，可能导致生成的代码质量不高或不符合开发者的预期。  **精化内容选择困难**  难以自动选择伪代码精化基本块，实现伪代码层面的精化内容选择和对应代码生成，可能会影响代码的性能和可维护性。  **代码一致性方面不一致性检测困难**  在检测代码与相应伪代码之间的不一致性方面存在挑战，无法及时准确地发现并提示开发者，导致开发者在修正预测性代码生成错误时需要花费更多的时间和精力。   1. 创新点与项目特色   **主动逐步精化范式**：  提出一种以人为主逐步精化的代码生成框架，以伪代码为中间桥梁，实现从自然语言需求到程序详细设计原型再到具体实现的主动逐步精化编程新范式，为软件开发提供了一种新的思路。  **系统分析粒度对代码质量的影响**：  探索伪代码粒度对代码生成质量的影响，通过设计定量与定性结合的实证研究方案，系统分析不同粒度的伪代码输入对生成代码质量的影响。研究将伪代码划分为多粒度层次，并通过标准化的代码质量评估指标和开发者主观评价进行多维度评估。旨在为软件开发提供不同粒度选择的指导，优化模型设计，提升生成代码的准确性和效率。  **应用特色针对性强**：  针对本科阶段的软件开发需求，设计了简单易用的开发工具和交互界面，降低了软件开发的门槛。  **实践导向**：  通过实际项目的开发和应用，验证了研究成果的可行性和有效性，为软件工程教学和实践提供了有益的参考。   1. 技术路线、拟解决的问题及预期成果   **技术路线概述**  本研究的技术路线旨在通过系统化的步骤实现主动逐步精化式 IDE 的设计与实现，以提升软件开发效率和质量。以下是具体步骤：  **一：评估当前技术**  首先，系统回顾大语言模型（LLMs）在代码生成中的应用，评估主流模型如 GPT-4、Codex 的代码能力，并分析集成开发环境（IDE）中主流 AI 代码编辑插件（如 GitHub Copilot、Tabnine）的表现，重点关注代码可信度和用户交互的优缺点。  **二：设计新范式**  基于现有技术的分析，设计一种以伪代码为中间桥梁的主动逐步精化编程新范式。从自然语言需求出发，生成伪代码作为详细设计原型，用户通过人机交互逐步精化伪代码，包括研究人机交互设计、伪代码粒度选择及伪代码与代码间的一致性检测与校正方法。  **三：研究伪代码粒度影响**  通过定量与定性结合的实证研究，探索伪代码粒度对代码生成质量的影响。将伪代码划分为多粒度层次（如高层次、中层次、低层次），并使用标准化代码质量评估指标（如正确性、效率、可读性）和开发者主观评价进行多维度评估，为优化模型设计提供指导。  **四：开发 AI 代码编辑插件**  基于上述新范式，开发一款能实现主动逐步精化的 AI 代码编辑插件，集成到 IDE 中，支持从自然语言生成伪代码、用户逐步精化及最终生成代码，并确保伪代码与代码的一致性。  **五：验证与评估**  通过实际项目开发和用户测试验证插件的有效性，测量软件开发效率和代码质量的提升，收集反馈以迭代优化工具，特别针对本科阶段软件开发需求设计简单易用的交互界面。  **拟解决的问题**  本研究旨在解决以下关键问题：  **一：提升 LLM 生成代码的可信度**  当前 LLM 生成的代码可能存在幻觉（hallucinations），如生成不正确的函数或未满足需求的逻辑。通过主动逐步精化过程，用户可介入修正，提升代码可靠性。  **二：增强人机协作的精化过程**  现有 AI 代码编辑工具如 GitHub Copilot 提供实时代码建议，但缺乏用户深度参与的精化机制。新范式通过伪代码桥梁增强用户控制力。  **三：利用伪代码桥接自然语言与代码**  自然语言需求（如“设计一个用户登录系统”）与代码（如 Python 实现）间存在语义鸿沟，伪代码（如“验证用户名和密码，记录登录时间”）可作为中间表示，优化转换过程。  **四：确定伪代码的最佳粒度**  伪代码粒度过粗可能导致生成代码缺乏细节，过细则增加用户负担。研究需找到平衡点，优化代码质量和用户体验。  **预期成果**  通过本研究，预期取得以下成果：  **一：新型编程范式**  提出一种以伪代码为桥梁的主动逐步精化代码生成框架，为软件开发提供新思路。  **二：伪代码粒度影响的洞察**  系统分析伪代码粒度对代码质量的影响，提供不同粒度选择的指导，优化模型设计。  **三：功能性 AI 代码编辑插件**  开发一款集成新范式的插件，特别适合本科阶段软件开发，具备用户友好界面，提升开发效率和代码质量。  **四：理论贡献**  深化对自然语言与程序代码关系的理解，丰富软件工程理论研究。  **五：实践效益**  通过实际项目验证工具的可行性，为软件工程教学和实践提供参考，降低开发门槛，减少错误和风险。   1. 项目研究进度安排 2. 已有基础    1. 与本项目有关的研究积累和已取得的成绩   **大语言模型在软件开发领域的应用研究**  **代码生成能力评估**  我们对 GPT-4、Codex 等主流大语言模型在代码生成与程序合成方面的表现展开了深入研究。研究发现，这些模型在自动化编码任务中展现出了显著的优势。例如，在 HumanEval 数据集中，GPT-4 的通过率超过 50%，这意味着它能够有效地生成符合要求的代码。在实际应用中，GPT-4 可以快速理解开发人员的需求，并生成相应的代码框架和实现逻辑，大大提高了开发效率。  **软件质量保证效果**  研究表明，经过适当调整的大语言模型不仅能够高效地识别代码中的安全漏洞，还能够依据自然语言规范自动生成测试用例，从而确保软件的稳定性。特别是在安全敏感领域，如金融、医疗等行业，这一技术的应用价值尤为突出。例如，通过对代码的语义分析和逻辑推理，大语言模型可以准确地判断出可能存在的安全隐患，并生成针对性的测试用例，有效地提高了软件的安全性和可靠性。  **软件维护功能**  基于 ChatGPT 等工具，项目能够自动识别并修复潜在的程序错误，简化了维护流程，提高了开发效率。对于长期项目管理而言，这一功能具有显著的优势。例如，当项目代码规模较大、维护难度较高时，ChatGPT 可以快速定位错误代码，并提供相应的修复建议，帮助开发人员快速解决问题，减少了维护成本和时间。  **相关技术研究成果**  **程序规范生成**  SpecGen 技术作为一种创新的程序规范生成方法，能够充分利用大语言模型的强大语言理解和生成能力，生成形式化的程序规范。与传统方法相比，SpecGen 克服了生成简单和基本规范的局限性，能够为复杂程序提供准确、全面的规范描述。例如，在处理一些复杂的算法和数据结构时，SpecGen 可以生成详细的前置条件、后置条件和循环不变式，帮助开发人员更好地理解程序的行为和功能，提高了程序的可维护性和可靠性。  **意图提取与代码细化**  通过对代码审查中的评论进行深入分析，我们能够准确地提取出开发者的意图，并据此进行代码细化。这种方法将复杂的代码修改任务分解为明确的意图和相应的代码生成步骤，大大提高了代码修改的准确性和效率。例如，当开发者提出 “优化算法性能” 的意图时，系统可以通过对代码的分析和理解，自动生成优化后的代码实现，减少了人工干预的成本和错误率。  **代码补全与检索增强**  FT2Ra 方法是一种基于检索增强和微调技术的代码补全方法，它通过引入检索机制和多轮迭代的微调过程，显著提高了 token-level 和 line-level 代码补全任务的性能。在实际应用中，FT2Ra 能够快速从大量的代码数据中检索出与当前代码上下文相关的信息，并利用这些信息对模型进行微调，从而生成更加准确和完整的代码补全结果。例如，在开发人员编写代码时，FT2Ra 可以根据已有的代码上下文，快速提供相关的代码片段和函数调用建议，帮助开发人员提高开发效率。  **开发工具与插件的研究**  我们计划开发一款 AI 代码编辑插件，该插件将支持从自然语言生成伪代码，并通过用户的逐步精化，最终生成符合要求的代码。同时，插件将确保生成的代码与伪代码保持一致，为开发人员提供了一种高效、便捷的开发工具。在开发过程中，我们将充分利用大语言模型的语言理解和生成能力，以及现有的开发技术和框架，打造一款功能强大、易于使用的 AI 代码编辑插件。   * 1. 已具备的条件，尚缺少的条件及解决方法   **已具备的条件**  **技术知识储备**  团队成员具备一定的人工智能与软件工程等领域的专业知识，掌握一定大语言模型、程序设计等相关技术，怀有学习的热情和信心，能够为项目的技术开发提供支持。  **研究基础**  项目组已对大语言模型在代码生成、软件质量保证、软件维护等方面的应用进行了深入研究，积累了丰富的实验数据和研究经验。通过对相关技术的研究，掌握了大语言模型的基本原理和应用方法，为项目的开展奠定了坚实的理论基础。  **实验环境与工具**  具备先进的实验设备和充足的计算资源，包括高性能计算机、GPU 服务器等，能够满足大语言模型的训练和实验需求。同时，我们还拥有完善的开发工具和调试环境，如代码编辑器、编译器、调试器等，为项目的开发提供了有力的技术支持。  **尚缺少的条件及解决方法**  **高质量数据集**  **缺少情况**：目前缺乏针对本科阶段软件开发需求的高质量、特定领域的数据集，这可能会影响模型的训练效果和泛化能力。  **解决方法**  **数据收集**：积极收集开源数据集，如 GitHub 代码库、CodeReview 数据集等，从中筛选出符合本科阶段软件开发特点的代码样本。  **数据标注**：结合实际项目开发，对收集到的数据进行人工标注，确保数据的准确性和完整性。  **数据扩充**：采用数据增强技术，如数据复制、随机扰动等，扩充数据集的规模，提高模型的训练效果。  **理论研究深度**  **缺少情况**：在伪代码粒度、生成方法、模型优化等方面的理论研究还不够深入，缺乏系统的理论框架和数学模型支持。  **解决方法**  **加强理论学习**：组织团队成员深入学习相关领域的理论知识，包括自然语言处理、程序分析、机器学习等，为项目的理论研究提供坚实的基础。  **开展实证研究**：通过大量的实验和数据分析，探索伪代码粒度对代码生成质量的影响规律，研究不同生成方法的优缺点，优化模型的结构和参数，提高模型的性能和效率。  **建立理论模型**：基于实证研究结果，建立相应的理论模型，如伪代码粒度与代码质量之间的数学模型、模型优化的目标函数等，为项目的理论研究提供系统的框架和指导。  **实践经验丰富度**  **缺少情况**：目前团队在将研究成果应用于实际开发中的经验还比较有限，对实际开发过程中可能遇到的问题和挑战缺乏充分的认识和应对能力。  **解决方法**  **参与实际项目**：积极组织团队成员参与实际项目开发，将研究成果应用于实际场景中，通过实践不断积累经验，提高团队的实践能力和解决问题的能力。  **与企业合作**：加强与软件开发企业的合作，了解企业的实际需求和开发流程，将研究成果与企业的实际应用场景相结合，为企业提供更加实用的技术解决方案。  **开展培训与交流**：定期组织团队内部的培训和交流活动，分享实践经验和心得体会，促进团队成员之间的相互学习和提高。 |

1. **经费预算**

|  |  |  |
| --- | --- | --- |
| 开支科目 | 预算经费（元） | 主要用途 |
|
| 预算经费总额 |  |  |
| 1. 业务费 |  |  |
| （1）计算、分析、测试费 |  |  |
| （2）能源动力费 |  |  |
| （3）会议、差旅费 |  |  |
| （4）文献检索费 |  |  |
| （5）论文出版费 |  |  |
| 2. 仪器设备购置费 |  |  |
| 3. 实验装置试制费 |  |  |
| 4. 材料费 |  |  |
| 1. 其他 |  |  |

1. **指导教师意见**

|  |
| --- |
|  |