http://htkz.cn
引用格式：陈晓阳，高飞，韩翔宇，等. 大模型代码生成技术及航天领域潜在应用[J]. 航天控制，2025，43(1)：8-16.（CHEN Xiaoyang, GAO Fei, HAN Xiangyu, et al. Large model code generation technology and its potential applications to aerospace [J]. Aerospace Control, 2025, 43(1): 8-16.）

大模型代码生成技术及航天领域潜在应用

陈晓阳[1]，高　飞[1]，韩翔宇[1]，马卫华[2]
1. 北京航天自动控制研究所，北京 100854
2. 北京神舟航天软件技术股份有限公司，北京 100094

摘　要　考虑到基于大语言模型(LLMs)的代码生成技术对软件生产力的巨大影响及在航天领域的应用前景广泛，本文从问题背景与定义、典型技术与其在航天领域的潜在应用场景以及应用评价方法3个方面，综述了该技术的最新研究进展，以期为航天领域代码生成技术的相关研究提供指导与启发。首先，从代码生成问题定义及LLMs的结构特点，讨论了LLMs在代码生成方面的基础能力；然后，在此基础上，详述了包括预训练技术、指令微调技术、提示词工程和检索增强技术等实现代码生成的主要方法及其在航天领域的潜在应用场景；接着，从语义相似性和验证数据集两方面，梳理了评估基于LLMs的代码生成技术的主要方法，并分析了它们的特点及局限性；最后讨论了LLMs技术在代码生成问题中所面临的挑战及未来发展方向。

关键词　代码生成；大语言模型；预训练；指令微调；检索增强
中图分类号：TP311　　**文献标识码**：A　　**文章编号**：1006-3242(2025)01-0008-09

Large model code generation technology and its potential applications to aerospace

CHEN Xiaoyang[1], GAO Fei[1], HAN Xiangyu[1], MA Weihua[2]
1. Beijing Aerospace Automatic Control Institute, Beijing 100854, China
2. Beijing Shenzhou Aerospace Software Technology Co., Ltd, Beijing 100094, China

Abstract　Regarding the significant impact of code generation techniques based on large language models (LLMs) on software productivity and their broad application prospects to the aerospace field, the latest research progress on this kind of technology is reviewed from three aspects: problem background and definition, typical technologies and their potential application scenarios in the aerospace domain, and application evaluation methods, with the aim of providing guidance and insights for related research on code generation techniques in the aerospace domain. Firstly, the basic capabilities of LLMs are discussed on code generation according to the features of code generation problem definition and LLMs structures. Then, the main methods for code generation are elaborated, including pre-training, instruction

收稿日期：2024-11-12
作者简介：陈晓阳(2000-)，女，硕士研究生，主要研究方向为基于大语言模型的自动代码生成技术；高　飞(1981-)，男，博士，高级工程师，主要研究方向为统计学习、大模型技术和智能控制，本文通信作者。

fine-tuning, prompt engineering and retrieval-augmented generation as well as their potential application scenarios to the aerospace field. Next, due to the perspectives of semantic similarity and validation datasets, the popular methods are reviewed for evaluating the results of LLM-based code generation techniques and their characteristics and limitations are analyzed. Finally, the challenges are presented and future improvements are proposed.

Key words　Code generation; Large language models; Pre-training; Instruction fine-tuning; Retrieval augmented

0　引　言

近年来,大语言模型(LLMs)技术的发展驱动着人工智能技术快速在各个领域形成了生产力。在软件研制方面,基于LLMs的代码生成技术因其巨大的应用前景和对软件产业格局可能产生的战略影响,已成为计算机科学领域的研究热点。该类技术的目标问题是利用LLMs对自然语言多层次、多尺度的语义理解能力和代码序列的生成能力,将自然语言文本转化为计算机能识别的对应语意的程序代码。从生成规模上看,代码生成可分为符号级、语句级、函数级、功能级和软件级等多个层次。传统方法利用语法树、符号表等形式化的结构和方法,较好地解决了符号级代码生成的问题,但由于语句级以上的代码生成需要基于对需求文本的语义理解,因此,采用形式化的方法较难实现。

随着大语言模型规模的不断增大、训练技术的持续成熟,利用海量文本和程序代码训练出的LLMs展现出了强大的语言理解与生成能力,能处理更加复杂的自然语言描述的需求并生成相应代码。相较于传统方法,其展现出了更高的灵活性和适应性,已初步在代码生成、补全及修复等任务中得到验证。结果表明,其显著提升了自动化编程的水平与软件开发的效率,也提升了代码缺陷检查工作的质量,在协助开发者高效完成重复性任务、缩短开发周期及减少人为错误方面具有重要意义。

航天领域对软件开发的复杂性和可靠性要求较高[1-2],涉及诸如轨道计算[3]、飞控系统[4]及数据处理[5]等多层面的任务,代码生成和验证需求尤为突出。然而,目前基于LLMs的代码生成技术在航天领域的应用尚处于探索阶段,缺乏专门面向航天场景的研究与实践[6]。鉴于LLMs在处理复杂需求文本、生成高效代码方面的潜力,针对航天领域的应用具有重要的前瞻性价值。通过结合航天领域的独特需求,基于LLMs的代码生成技术有望在飞控软件开发、任务规划与调度等方面发挥重要作用,从而推动航天技术的进一步发展。

本文从问题背景、典型技术及其在航天领域的潜在应用场景,以及效果评价方法3个方面,综述了基于LLMs的代码生成技术的最新研究进展,提出了一些针对航天领域的研究展望,为相关领域的研究者与从业者提供有益参考。

1　背　景

1.1　代码生成问题

代码生成,亦称程序合成,指依据用户设定的约束条件自动产生源代码的过程。其核心目标在于通过多样化的输入形式,自动生成满足特定需求的程序代码。该任务可依据输入类型的差异进行分类,涵盖自然语言预测[7]、基于示例或测试用例的预测[8]、跨编程语言的代码转换(即代码翻译)[9]以及错误代码的修复(即程序修复)[10]等情形。

早期的代码生成方法依赖于形式化的逻辑推理和规则系统,旨在通过自动化推导生成符合特定规范的代码,当前航天领域的代码自动生成技术的大量研究也主要集中于此类方法的应用[11]。然而,这些方法在处理复杂程序和多变的编程任务时,面临着较大的局限性。随着深度学习方法的兴起,神经网络方法逐渐成为代码生成领域的主流。例如,引入长短期记忆网络[12]和递归-反向递归神经网络[13]等技术,用于根据特定的归纳偏向生成输出程序。通过对大量程序样本的训练,这些神经网络模型能够在一定程度上学习到程序的结构和模式,从而实现更灵活的代码生成。近年来,基于Transformer架构的大语言模型(如GPT-3[14]和T5[15])通过学习大规模代码和自然语言数据,显著提升了程序合成能力。这些模型通过预训练与微调,深入理解代码结构与语义,并在各类软件开发任务中展现卓

越性能。随着模型规模和计算能力的提升，现代代码生成技术已能应对复杂编程需求，并广泛应用于自动化编程、代码补全与翻译等领域。

1.2 大语言模型

大语言模型的有效性源于其庞大的参数规模、多样化的数据集及强大的算力支持[16]。模型规模扩展显著提升了下游任务性能与样本效率。LLMs常基于Transformer[17]架构，而代码语言模型（Code LLMs）则通过大规模未标记代码语料库预训练获得优势。根据任务需求，Code LLMs可分为仅编码器、仅解码器及编码器-解码器3种架构。

编码器-解码器（Encoder - decoder）[18]架构为序列到序列的模型，由编码器和解码器构成。编码器利用注意力机制处理输入序列信息，解码器则基于已生成词汇逐步输出目标序列。基于此架构的代码语言模型，例如CodeT5[19]、BART[20]和T5[15]，在代码生成、优化及缺陷检测等任务中广泛应用。

仅编码器（Encoder-only）架构，通过编码器网络将输入数据转换为压缩表示，常用于无监督学习任务，如降维和异常检测。在人工智能辅助编程中，基于仅编码器架构的代码语言模型，例如BERT[21]和RoBERTa[22]，主要应用于代码理解任务。

仅解码器（Decoder-only）架构，通过解码器网络生成输出，依赖已生成词汇预测下一词汇，逐步构建完整序列。在代码语言模型中，基于Decoder-only架构的模型，如StarCoder[23]和GPT-3[14]（经特定训练），主要应用于代码生成与补全任务。

2 基于大模型的代码生成技术

基于Transformer架构的大语言模型在多个领域，尤其是代码生成中，取得了显著进展，其经典技术分类及在航天领域的潜在应用场景如图1所示。

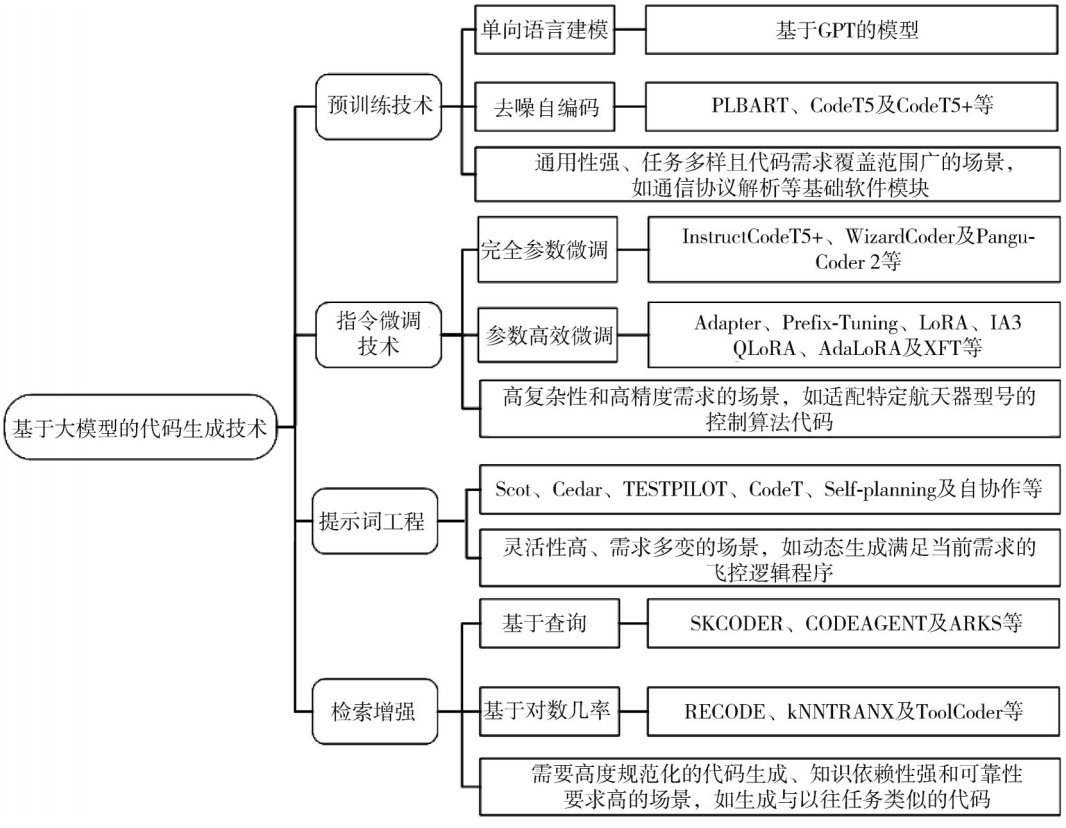

图1 基于大模型的代码生成技术分类

2.1 预训练技术

传统监督学习方法在代码生成任务中受限，特别是人工标注数据的不足，尤其在低资源语言中显著。为应对此挑战，研究者采用大规模未标注代码语料库进行预训练，以全面理解编程语言结构与语义。在航天领域，预训练技术因其通用性强、任务场景多样且代码需求覆盖范围广，展现出重要的应用价值。具体而言，它适用于生成基础的航天器嵌

入式软件模块(如数据采集和通信协议解析等)、提供轨道优化、任务调度和姿态控制等算法的初始实现,以及生成适用于遥测数据、图像处理和科学计算的通用分析工具代码。预训练技术的广泛适配能力,为航天领域复杂场景中的软件开发提供了强有力的基础支持。预训练目标源自自然语言处理并根据源代码特点调整,其中单向语言建模(LM)与去噪自编码(DAE)因其有效性而备受关注。

单向语言建模亦称因果语言建模,基于已生成词汇序列预测下一词汇[24]。在 LM 中,模型预测当前词汇后的下一词,采用因果注意力机制,确保序列生成的因果顺序性,避免未来信息干扰。通过分类头与交叉熵损失函数优化,LM 捕捉语言序列结构与规律。在代码生成中,基于 GPT 的模型经 LM 目标训练[25],高效生成符合上下文要求的源代码。

去噪自编码为序列到序列语言模型的预训练方法[26],通过随机遮蔽、删除或打乱词汇"污染"输入序列,模型需从损坏序列中恢复原始序列。相较于掩码语言模型,DAE 扩展至词汇的排列与丢弃,强化模型对输入序列的理解。DAE 广泛应用于编码器-解码器架构预训练,如 PLBART[27]、CodeT5[19]及其改进版 CodeT5+[28],显著提升模型生成语法与语义准确代码的能力,增强上下文表示学习。

2.2　指令微调技术

指令微调是大语言模型预训练后的关键优化步骤,旨在增强其处理和执行指令的能力。该过程通常在监督学习框架下进行,利用包含结构化自然语言指令的示例数据集对预训练 LLMs 进行微调。这些指令数据集涵盖广泛的任务示例,明确指定模型需完成的任务。指令微调技术在航天应用中能够为高复杂性和高精度需求的场景提供有力支持。例如,飞控系统开发中,它能够生成适配特定航天器型号的控制算法代码;在故障诊断与恢复中,则可以生成针对航天器异常的诊断程序和应急措施逻辑。此外,该技术还适用于复杂算法的实现。通过这种微调方法,模型在航天特定任务中的表现得到有效强化,能够更加准确地应对该领域内的复杂需求。指令微调通过利用指令类型的多样性,作为一种多任务提示训练方法,显著提升了模型对未见任务的泛化能力,从而更有效地适应多种实际应用场景。

完全参数微调(FFT)是指在预训练模型的基础上,更新模型中的所有参数。这一方法通常在具备充足计算资源和大量训练数据的情况下应用[29],因为它能够显著提高模型的性能。FFT 通过全面微调模型,使其能够更好地适应特定任务的要求,从而优化模型的泛化能力。代码生成效果优异的 InstructCodeT5+[19]模型即是 CodeT5+模型使用了 FFT,在 Code Alpaca 数据集上进行指令微调获得。类似地,WizardCoder[30]模型基于 Evol-Instruct 数据合成技术,将 20K 条 Code Alpaca 指令样本扩展为 78K 条数据,并通过 FFT 对 StarCoder[23]基础模型进行微调,从而大幅提升了代码生成能力。进一步的研究,如 Pangu-Coder 2[31],也采用了 Evol-Instruct 方法,并结合强化学习算法 RRTF,进一步优化了代码生成任务中的性能。

为应对 FFT 在计算与时间上的高昂需求,参数高效微调(PEFT)技术得以发展。PEFT 仅更新预训练模型中的部分参数[32],包括参数子集或额外引入的微调参数,从而大幅削减计算与存储成本,同时维持良好模型性能。典型 PEFT 技术包括 Adapter[33]、Prefix-tuning[34]、LoRA[35]、IA3[36]、QLoRA[37]及 AdaLoRA[38]等。其中,LoRA[35]通过因子化权重矩阵更新为两个低秩矩阵,实现参数高效更新,保持原始参数不变,仅训练低秩矩阵,并通过元素级加法与原有权重结合,增强模型能力。近期,XFT[39]结合回收的混合专家(MoE)模型、共享专家机制及新型路由权重归一化策略,显著提升了微调性能。

2.3　提示词工程

提示词工程作为一种高效技术,通过精心设计提示词引导预训练模型完成特定任务,无需额外训练即可显著提升性能。该技术尤其适用于代码生成任务,能在保持模型结构不变的情况下,通过优化提示词增强模型的任务适应性和表现。提示词工程因其灵活性和高适应性,在航天应用中展现出重要的潜在价值。它可以用于原型代码开发,快速生成任务脚本或算法原型,验证新的任务概念或技术;同时,在实时任务生成中,能够动态地创建满足当前需求的飞控逻辑或数据处理程序。此外,提示词工程还可用于代码补全与优化,通过精心设计提示词,生成完整的功能代码或提升现有代码的性能。凭借其高效引导模型输出的能力,提示词工程特别适合航天领域中需求多变且强调任务适配性的应用场景,为解决复杂任务提供了一种快捷且灵活的方法。

Scot[40]方法使用包含代码结构信息(如分支和循环结构)的结构化提示,旨在提升生成代码的准确性。Cedar[41]则采用检索相似任务的代码示例,并将其嵌入提示中,进一步改善代码生成效果。TESTPILOT[42]基于解释器反馈或测试执行结果迭代优化提示,提升生成代码的质量。CodeT[19]框架通过自生成的测试评估生成代码的质量。Self-planning技术[43]利用提示词将规划引入代码生成过程中,从而降低了解决问题的难度。自协作技术[44]通过角色指令,组成了分析师、编码员、测试员团队进行协作编码,展现了卓越的代码生成能力。

2.4 检索增强技术

LLMs虽在多领域成效显著,但仍面临灾难性遗忘及高计算需求等挑战。为应对此,检索增强生成(RAG)技术得以发展,通过外部知识库检索信息,弥补LLMs内置知识不足,显著提升知识密集型任务表现。RAG技术通过结合外部检索能力与精确的知识增强可以为航天任务提供强大的技术支持。在知识驱动的任务实现中,RAG能够通过检索历史任务文档或技术数据库,生成与既往任务相似的代码模板,从而提高开发效率与一致性。此外,在复杂系统集成过程中,RAG技术能够生成与现有仿真系统或任务规划工具兼容的代码,进一步推动航天系统的开发与部署。在代码生成领域,RAG方法主要分为基于查询和基于对数几率两类。

基于查询的RAG通过将检索到的信息与用户的查询结合,构建生成模型的输入提示,广泛应用于代码生成任务中。其核心思想是通过增强提示,将相关的代码示例、API细节、文档信息等与原始查询融合,提升生成模型的输出质量。典型的基于查询的RAG技术包括SKCODER[45],该方法通过检索相关代码片段生成草图模板进行代码生成;CODEAGENT[46]则设计了多种代理,支持Web搜索、文档检索、程序生成和正确性测试;此外,ARKS[47]方法结合了迭代式RAG,通过重新构造查询和更新检索源来优化生成质量。

相比之下,基于对数几率的RAG则在解码过程中通过对数几率(logits)来整合检索到的信息。在这一方法中,生成模型通过对数几率值结合检索内容,调整生成的每一步概率分布。典型的基于对数几率的RAG技术包括RECODE[48],该方法通过编辑距离从自然语言描述中检索并匹配代码,随后提取n-gram行动子树,并通过LSTM生成时在每步解码中利用这些子树的处理结果;kNNTRANX[49]则使用seq2tree模型将自然语言转换为代码抽象语法树(AST),并在每个解码步骤中通过搜索AST前缀数据存储中的隐藏状态来生成新概率,最终通过置信度网络与seq2tree模型的输出合并;ToolCoder[50]通过生成包含特殊标记的代码,并在遇到这些标记时执行在线或离线检索,用API调用填补空白,这是一种特定形式的推测性RAG。

3 效果评价

在代码生成领域,效果评价直接关联于模型输出代码的质量及实用性。效果评价可从多角度进行分类(如图2),以全面且系统地衡量模型性能。

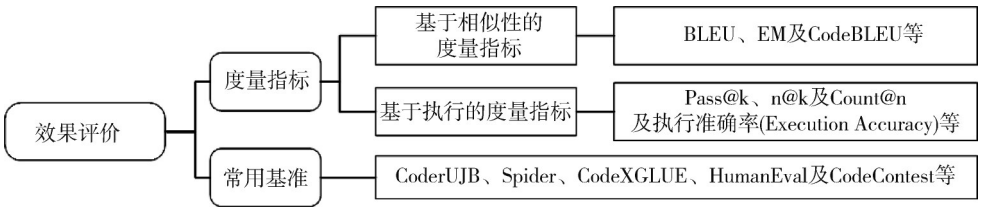

图2 代码生成效果评价分类

3.1 度量指标

在自然语言处理领域,设计有效且可靠的自动评估指标一直是长期存在的挑战。随着生成式模型在代码生成任务中的广泛应用,评估生成内容的质量变得愈发关键。目前,代码生成评估指标主要分为基于相似性和基于执行的两类。

在代码生成任务中,基于相似性的评估方法主要依据生成代码与参考代码的相似度进行评判。BLEU[51]作为一种常见指标,通过计算生成文本与参考文本间n-gram的重叠度来评估相似性,具体方法是统计匹配的n-gram数量并加权平均得出整体评分。在初步评估阶段,BLEU常用于衡量生成代码与参考代码的相似度。然而,鉴于代码语法与语义的复杂性,单纯依赖n-gram匹配可能无法准确反

映代码功能差异,且代码标识符的不同亦可能影响BLEU的准确性。相比之下,EM(Exact match)[52]通过直接比较生成代码与参考代码的完全一致性来评估准确性,适用于判断代码的整体正确性,但对小语法或格式偏差敏感,可能低估模型表现。为克服上述局限,CodeBLEU[53]作为一种针对代码生成的评估方法,在传统BLEU基础上融入了语法和语义信息,通过加权n-gram匹配、抽象语法树(AST)匹配及语义数据流匹配,综合评估代码的语法结构、内在逻辑及功能相似性。通过对比生成代码与参考代码的AST结构,CodeBLEU不仅衡量表面相似度,还深入理解代码功能,提供更为准确的评估。研究[54]表明,CodeBLEU与程序员评分的相关性高于BLEU,且在多种编程语言中表现优异。

尽管基于相似性的评估指标在衡量生成代码的语法结构和文本相似性方面具有优势,但未能充分解决执行结果差异的问题。因此,发展了基于执行的评估标准,该方法通过运行生成代码并对比执行结果,从实际执行层面评估代码的正确性和可靠性。常见的基于执行评估方法包括Pass@k[54]、n@k[55]、Count@n[56]及执行准确率(Execution accuracy)[57]等。其中,Pass@k指标通过衡量生成代码在前k次尝试中通过测试的概率,评估代码生成模型的性能和可靠性,Pass@1反映首次生成代码质量,Pass@5与Pass@10则评估模型在多样性与稳定性上的表现。相较于传统的相似性度量,其实用性和可信度更强,尤其在验证代码实际功能时更为可靠。

3.2　常用基准

代码生成领域中验证代码正确性至关重要。常用基准测试数据集包括CoderUJB[56]和Spider[57]、CodeXGLUE[52]、HumanEval[54]及CodeContest[55]等,涵盖了从简单算法到复杂系统的广泛场景。

CodeXGLUE[52]是一个综合性基准平台,涵盖多种代码理解与生成任务,如代码克隆检测、缺陷检测、文本到代码生成及代码摘要生成等,涉及Java、C/C++及Python等编程语言。该平台数据集经过严格预处理,确保了高质量与一致性,增强了模型的实际应用能力。其显著特点在于数据集的多样性和广泛覆盖,使之成为学术研究与工业应用的关键评估工具。评估指标包含BLEU、EM、Accuracy、F1分数及CodeBLEU等,既有传统标准,也有针对代码生成的专门指标(如EM和CodeBLEU用于文本到代码生成,Accuracy和F1分数用于代码克隆检测),从而实现了全面的性能评估。

HumanEval[54]数据集旨在评估代码生成模型性能,包含164个Python编程任务,每个任务配备自然语言描述及多个测试用例,覆盖从基础控制结构至复杂数据结构的算法,全面测试模型功能性与正确性。所有任务均手工编写,确保质量与独特性。评估主要采用Pass@k指标。

4　挑战与机遇

尽管LLMs在代码生成上取得了显著成效,但仍处在刚刚起步阶段。代码生成的复杂性,如严格的语法要求、隐式的动态执行特性,以及多模块、跨文件、多层次的指代引用关系等都对LLMs的语意理解能力提出了更高的要求。本节从生成和评估两个方面对当前面临的技术挑战性进行了归纳,并提出了未来可能的研究方向。

当前,大模型在生成简单函数级代码时表现优秀,但在处理复杂编程问题,尤其是涉及多个模块或系统级别的代码生成时,效果还有待改进。因此,提升大模型在复杂编程任务中的表现,特别是在处理代码仓库和软件系统问题时的能力,成为未来研究的重要方向。

此外,确保编码安全并将LLMs生成的代码与人类编码偏好对齐是代码生成方面的另一个挑战。现有模型本身的随机性带来了创造性,同时也可能会引入安全漏洞或生成不符合期望规范的代码片段。解决该问题的一个值得探索的方向是将形式化工具与软件工程数据相融合,对LLMs进行规则化约束,从而提升代码生成的安全性;另外,开发对齐学习框架,在训练过程中,让LLMs更好地“内化”人类的编码偏好,从而使生成的代码具有人类代码的“好味道”。

另一个亟待解决的问题是LLMs编码能力评估缺乏全面基准。目前的评估基准主要关注语法正确性和功能准确性,未能充分反映实际开发中的复杂需求。未来随着机器生成代码量的不断增加,对代码的可测试性、可扩展性以及容错性等其他软件工程指标的量化评价表示以及数据集建设也会变得越来越迫切,并成为推动基于LLMs生成代码技术进一步发展的基础。

将代码生成技术应用于航天领域这一高度规

范化的场景也是一个值得关注的方向。目前,研究多集中于通用代码生成,而针对航天领域的专门研究相对较少。航天需求文本通常具备清晰的技术描述和结构化特点,这为构建通用功能函数库提供了基础。通过精确解析需求文本,可以自动生成数据采集、轨道计算及姿态控制等模块,从而提升代码生成的准确性和效率。因此,航天领域的大模型代码生成技术不仅填补了研究空白,还为大模型在特定领域的应用提供了新的发展机遇。

基于大语言模型强大的语义理解能力,目前的代码生成技术从生成代码规模、完整性及语义一致性等方面较传统方法已经有了较大进步,也极大地提升软件自动化编程的水平与效率,直接变成生产力的一部分。未来随着 LLMs 能力的不断增强、Agent 和 RAG 等技术的快速发展,以及评估方法的有效进步,基于 LLMs 的自动化编程系统将能够处理更加复杂的编程任务需求,生成更加精细、安全且可靠的代码,为其在航天领域的应用奠定基础。

参 考 文 献

[1] 杨孟飞, 顾斌, 郭向英, 等. 航天嵌入式软件可信性保障技术及应用研究[J]. 中国科学: 技术科学, 2015, 45(2): 198-203. (YANG Mengfei, GU Bin, GUO Xiangying, et al. Research on reliability assurance technology and application of aerospace embedded software[J]. Science China: Technological Sciences, 2015, 45(2): 198-203.)

[2] 杨孟飞,顾斌,段振华,等. 嵌入式软件智能合成框架及关键科学问题[J]. 中国空间科学技术,2022, 42(4): 1-7. (YANG Mengfei, GU Bin, DUAN Zhenhua, et al. Intelligent synthesis framework of embedded software and key scientific issues[J]. Chinese Journal of Space Science and Technology, 2022, 42(4): 1-7.)

[3] 蒙波,韩潮. 高精度航天器轨道预报仿真软件的研制[J]. 计算机仿真,2008,25(1):62-65+73. (MENG Bo, HAN Chao. Development of high-precision spacecraft orbit prediction simulation software[J]. Computer Simulation, 2008, 25(1): 62-65+73.)

[4] 孙雪娇, 刘学士, 束韶光. 基于实时操作系统的多核分布式飞行软件架构设计[J]. 航天控制, 2023, 41(1): 47-52. (SUN Xuejiao, LIU Xueshi, SHU Shaoguang. Design of multicore distributed flight software architecture based on real-time operating system[J]. Aerospace Control, 2023, 41(1): 47-52.)

[5] 曹雪君, 刘学士, 姚娜, 等. 基于时序数据库与发布订阅机制的环境试验数据处理平台[J]. 航天器环境工程, 2024, 41(3): 379-388. (CAO Xuejun, LIU Xueshi, YAO Na, et al. Environmental test data processing platform based on time-series database and publish-subscribe mechanism[J]. Spacecraft Environment Engineering, 2024, 41(3): 379-388.)

[6] 冯思喆, 杨志斌, 薛垒. 基于AADL的航天嵌入式软件 Ada 代码自动生成方法[J]. 计算机与现代化, 2020, 9(6): 52-59+88. (FENG Sizhe, YANG Zhibin, XUE Lei. Ada code automatic generation method for aerospace embedded software based on AADL[J]. Computer and Modernization, 2020, 9(6): 52-59+88.)

[7] 徐一然,周宇. 基于提示学习的轻量化代码生成方法[J]. 计算机科学,2024, 51(6):61-67. (XU Yiran, ZHOU Yu. Prompt learning based parameter-efficient code generation[J]. Computer Science, 2024, 51(6): 61-67.)

[8] GULWANI S. Automating string processing in spreadsheets using input-output examples [J]. ACM Sigplan Notices, 2011, 46(1): 317-330.

[9] 李征,徐明瑞,吴永豪,等. 基于层次注意力机制的源代码迁移模型[J]. 计算机应用研究,2023, 40(10): 3082-3090. (LI Zheng, XU Mingrui, WU Yonghao, et al. Source code migration model based on hierarchical attention mechanism[J]. Application Research of Computers, 2023, 40(10): 3082-3090.)

[10] 孙昌爱,吴思懿,张守峰,等. 基于模板匹配的BPEL程序故障修复及优化技术[J]. 软件学报, 2024, 35(6): 2844-2862. (SUN Chang′ai, WU Siyi, ZHANG Shoufeng, et al. Fault repair and optimization techniques for BPEL programs based on template matching [J]. Journal of Software, 2024, 35(6): 2844-2862.)

[11] 戚浩楠. 符合航天安全规范的代码自动生成系统研究[D]. 杭州: 浙江工业大学, 2018. (QI Haonan. Research on automatic code generation system complying with aerospace safety specifications[D]. Hangzhou: Zhejiang University of Technology, 2018.)

[12] DONG L, LAPATA M. Language to logical form with neural attention [C]//Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, August 7-12, 2016.

[13] PARISOTTO E, MOHAMED A, SINGH R, et al. Neuro-symbolic program synthesis [C]//International Conference on Learning Representations, Toulon, France, April 24-26, 2017.

[14] BROWN T B , MANN B , RYDER N ,et al. Language models are few-shot learners[EB/OL]. 2020[2024]. https://arxiv.org/abs/2005.14165.

[15] RAFFEL C, SHAZEER N, ROBERTS A, et al. Explor-

ing the limits of transfer learning with a unified text-to-text transformer [J]. Journal of Machine Learning Research, 2020, 21(140): 1-67.

[16] HOFFMANN J, BORGEAUD S, MENSCH A, et al. Training compute-optimal large language models [C]// Proceedings of the 36th International Conference on Neural Information Processing Systems, Los Angeles, USA, November 2- December 9, 2022.

[17] VASWANI, A. Attention is all you need[C]// Proceedings of the Advances in Neural Information Processing Systems 30, Long Beach, USA, December 4-9, 2017.

[18] 董传珂,赵逢禹,刘亚. 基于注意力机制的双编码器代码注释生成[J]. 小型微型计算机系统, 2022, 43(2): 438-442. (DONG Chuanke, ZHAO Fengyu, LIU Ya. Dual encoder code comment generation based on attention mechanism [J]. Journal of Chinese Computer Systems. 2022, 43(2): 438-442.)

[19] WANG Y, WANG W, JOTY S, et al. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation[C]//Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Punta Cana, Dominican Republic, November 7-11, 2021.

[20] LEWIS M, LIU Y, GOYAL N, et al. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension[C]// Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, July 5-10, 2020.

[21] KENTON J D M W C, TOUTANOVA L K. Bert: Pre-training of deep bidirectional transformers for language understanding[C]//Proceedings of NaacL-HLT, Minneapolis, USA, June 2-7, 2019.

[22] LIU Z, LIN W, SHI Y, et al. A robustly optimized BERT pre-training approach with post-training [C]// China National Conference on Chinese Computational Linguistics, Hohhot, China, August 13-15, 2021.

[23] BROWN T B. Language models are few-shot learners [J]. Advances in Neural Information Processing Systems, 2020, 33(1): 1877-1901.

[24] 岳增营,叶霞,刘睿珩. 基于语言模型的预训练技术研究综述[J]. 中文信息学报,2021,35(9):15-29. (YUE Zengying, YE Xia, LIU Ruiheng. A survey on pre-training techniques based on language models [J]. Journal of Chinese Information Processing, 2021, 35(9): 15-29.)

[25] 向历霓,李刚,李海江. 基于知识图谱和GPT模型的可靠性代码自动生成方法[J]. 计算力学学报,2024,41(2):217-225. (XIANG Lini, LI Gang, LI Haijiang. A reliability-based automatic code generation method combining knowledge graph and GPT model [J]. Chinese Journal of Computational Mechanics, 2024, 41(2): 217-225.)

[26] 徐东钦. 基于Transformer预训练模型的语言特征分析及其应用[D]. 苏州: 苏州大学, 2021. (XU Dongqin. Analysis of language features based on transformer pre-trained models and their applications [D]. Suzhou: Soochow University, 2021.)

[27] AHMAD W U, CHAKRABORTY S, RAY B, et al. Unified pre-training for program understanding and generation [C]//Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, June 6-11, 2021.

[28] WANG Y, LE H, GOTMARE A, et al. CodeT5+: Open code large language models for code understanding and generation[C]//Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Sentosa, Singapore, December 6-10, 2023.

[29] 张钦彤,王昱超,王鹤羲,等. 大语言模型微调技术的研究综述[J]. 计算机工程与应用,2024,60(17):17-33. (ZHANG Qintong, WANG Yuchao, WANG Hexi, et al. A survey on fine-tuning techniques for large language models[J]. Computer Engineering and Applications, 2024, 60(17): 17-33.)

[30] LUO Z, XU C, ZHAO P, et al. WizardCoder: Empowering code large language models with evol-instruct[C]// The Twelfth International Conference on Learning Representations, Vienna, Austria, May 7-11, 2024.

[31] SHEN B, ZHANG J, CHEN T, et al. Pangu-coder2: Boosting large language models for code with ranking feedback[EB/OL]. 2023[2024]. https://arxiv. org/abs/2307. 14936.

[32] 王嘉宁. 面向预训练语言模型的提示调优方法[D]. 上海: 华东师范大学, 2024. (WANG Jianing. Prompt tuning methods for pre-trained language models [D]. Shanghai: East China Normal University, 2024.)

[33] HOULSBY N, GIURGIU A, JASTRZEBSKI S, et al. Parameter-efficient transfer learning for NLP[C]//International Conference on Machine Learning, PMLR, CA, USA, June 9-15, 2019.

[34] LI X L, LIANG P. Prefix-Tuning: Optimizing continuous prompts for generation[C]//Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Online, August 1-6,

2021.

[35] HU E J, WALLIS P, ALLEN-ZHU Z, et al. LoRA: Low-rank adaptation of large language models [C]// International Conference on Learning Representations, Online, April 25-29, 2022.

[36] LIU H, TAM D, MUQEETH M, et al. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning[J]. Advances in Neural Information Processing Systems, 2022, 35(1): 1950-1965.

[37] DETTMERS T, PAGNONI A, HOLTZMAN A, et al. Qlora: Efficient finetuning of quantized LLMs [J]. Advances in Neural Information Processing Systems, 2024, 36(1):10088-10115.

[38] ZHANG Q, CHEN M, BUKHARIN A, et al. Adaptive budget allocation for parameter-efficient fine-tuning[C]// International Conference on Learning Representations, Kigali, Rwanda, May 1-5, 2023.

[39] DING Y, LIU J, WEI Y, et al. XFT: Unlocking the power of code instruction tuning by simply merging upcycled mixture-of-experts [EB/OL]. 2024 [2024]. https://arxiv.org/abs/2404.15247.

[40] LI J, LI G, LI Y, et al. Structured chain-of-thought prompting for code generation[EB/OL]. 2023[2024]. https://arxiv.org/abs/2305.06599.

[41] NASHID N, SINTAHA M, MESBAH A. Retrieval-based prompt selection for code-related few-shot learning [C]//2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), Melbourne, Australia, May 14-20, 2023.

[42] SCHÄFER M, NADI S, EGHBALI A, et al. An empirical evaluation of using large language models for automated unit test generation[J]. IEEE Transactions on Software Engineering, 2024, 50(1): 85-105.

[43] JIANG X, DONG Y, WANG L, et al. Self-planning code generation with large language models[J]. ACM Transactions on Software Engineering and Methodology, 2024, 33(7): 1-30.

[44] DONG Y, JIANG X, JIN Z, et al. Self-collaboration code generation via ChatGPT[J]. ACM Transactions on Software Engineering and Methodology, 2024, 33(7): 1-38.

[45] LI J, LI Y, LI G, et al. Skcoder: A sketch-based approach for automatic code generation[C]//2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), Melbourne, Australia, May 14-20, 2023.

[46] ZHANG K, LI J, LI G, et al. Codeagent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges[EB/OL]. 2024 [2024]. https://arxiv.org/abs/2401.07339.

[47] SU H, JIANG S, LAI Y, et al. ARKS: Active retrieval in knowledge soup for code generation[EB/OL]. 2024 [2024]. https://arxiv.org/abs/2402.12317.

[48] HAYATI S A, OLIVIER R, AVVARU P, et al. Retrieval-based neural code generation[C]//Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018.

[49] ZHANG X, ZHOU Y, YANG G, et al. Syntax-aware retrieval augmented code generation[C]//Findings of the Association for Computational Linguistics: EMNLP 2023, Sentosa, Singapore, December 6-10, 2023.

[50] ZHANG K, ZHANG H, LI G, et al. Toolcoder: Teach code generation models to use api search tools[EB/OL]. 2023[2024]. https://arxiv.org/abs/2305.04032.

[51] PAPINENI K, ROUKOS S, WARD T, et al. BLEU: A method for automatic evaluation of machine translation [C]//Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Pennsylvania, USA, July 7-12, 2002.

[52] LU S, GUO D, REN S, et al. CodeXGLUE: A machine learning benchmark dataset for code understanding and generation[C]//Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1), Online, December 5-10, 2021.

[53] REN S, GUO D, LU S, et al. Codebleu: A method for automatic evaluation of code synthesis[EB/OL]. 2020 [2024]. https://arxiv.org/abs/2009.10297.

[54] CHEN M, TWOREK J, JUN H, et al. Evaluating large language models trained on code [EB/OL]. 2021 [2024]. https://arxiv.org/abs/2107.03374.

[55] LI Y, CHOI D, CHUNG J, et al. Competition-level code generation with alphacode[J]. Science, 2022, 378 (6624): 1092-1097.

[56] ZENG Z, WANG Y, XIE R, et al. Coderujb: An executable and unified java benchmark for practical programming scenarios[C]//Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis, Vienna, Austria, September 16-20, 2024.

[57] YU T, ZHANG R, YANG K, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task[C]//Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Brussels, Belgium, October 31-November 4, 2018.