针对 ABAP 代码生成的大语言模型基准测试：通过编译器反馈进行迭代改进的实证研究

预印本

Stephan Wallraven
科隆应用技术大学 stephan_andreas.wallraven@smail.th-koeln.de

Tim Köhne科隆应用技术大学
tim.koehne@smail.th-koeln.de

Hartmut Westenberger科隆应用技术大学
hartmut.westenberger@th-koeln.de

Andreas MoserCONSILIO GmbH
andreas.moser@consilio-gmbh.de

arXiv:2601.15188v1 [cs.SE] 21 Jan 2026

摘要

本研究考察了大语言模型（LLM）生成 ABAP 代码的表现。尽管生成式人工智能已成功应用于许多编程语言，但迄今几乎没有针对 ABAP 代码生成的系统性分析。本研究旨在通过实证分析，探究各类大语言模型在多大程度上能够生成语法正确且可运行的 ABAP 代码、它们利用编译器反馈进行迭代改进的效果如何，以及哪些任务类型构成了特殊挑战。为此，我们开展了一项包含 180 个任务的基准测试，其中包括改编后的 HumanEval 任务和实际 SAP 场景。结果显示，不同模型的表现差异显著：能力更强的大语言模型经过多轮迭代后，成功率可达约 75%，并能从编译器反馈中获益良多；而规模较小的模型表现则明显较弱。总体而言，本研究凸显了强大大语言模型在 ABAP 开发流程中的巨大潜力，尤其是在迭代错误纠正方面。

关键词 大语言模型 · ABAP · 代码生成 · 基准测试 · HumanEval · 编译器反馈 k

1 引言

1.1 背景与动机

近年来，大语言模型迅速发展，代码生成也日益融入软件开发流程。ChatGPT、Claude-Sonnet 以及 GitHub Copilot 等专门的软件开发工具，令人印象深刻地展现了生成式人工智能系统自动生成、解释和优化程序代码的能力 [1]。迄今为止，这些发展主要聚焦于 Python 或 JavaScript 等广泛使用的编程语言，这些语言拥有大量训练数据和基准测试。在这些语言中，大语言模型生成的程序已经展现出很高的正确性 [2–4]。然而，ABAP 等特定领域的企业语言迄今受到的研究关注要少得多，而 ABAP 在 SAP 系统中发挥着核心作用。人工智能用于自动代码生成有望带来显著的经济效益，因为它能够提高开发流程的效率，并大幅减少常规任务。

SAP 是 ERP 市场的领先服务提供商之一，并对企业的数字化转型产生了重大影响。SAP 等 ERP 系统对于控制各类规模企业的业务流程并整合业务逻辑至关重要。对于许多 SAP 系统而言，ABAP 仍是调整和扩展业务逻辑的核心编程语言。SAP 系统的广泛应用以及庞大的 ABAP 社区，使任何生产力提升（例如通过人工智能辅助代码生成实现的提升）都对许多企业具有现实意义和重要的经济价值 [5]。

1.2 问题陈述

尽管生成式人工智能领域总体上取得了进展，但迄今为止，针对 ABAP 代码生成的大语言模型应用几乎未得到研究。ABAP 是一种特定领域的专有语言，带来了特殊挑战，因为它深深嵌入 SAP 生态系统的架构和语义之中。它具有特定的语言概念，例如内表、数据声明、模块化程序结构，以及与 SAP事务和数据库的紧密交互。由于可能缺乏领域知识，通用大语言模型或许无法充分理解这些概念。
领域知识不足或许是一个核心问题，这在一定程度上可归因于缺少合适的训练数据。例如，Peter Färbinger 在一篇2024 [6]的博客文章中写道：“人工智能接收的训练数据越多，就可能变得越智能。举个例子：SAP 目前仍未能成功推出 ABAP 副驾驶。ABAP 是 SAP 自有的编程语言。全球范围内可用于训练 OpenAI 的 ChatGPT 这类生成式人工智能的 ABAP 程序代码实在太少。”SAP 及其合作伙伴近期发表的博客文章证实了这一观点：由于公开训练数据匮乏，ABAP 是一种“低资源语言”，因此生成式人工智能模型在处理 ABAP 时，往往不如处理 Python 或 Java 等语言时表现出色。SAP 和 NVIDIA 员工组成的一个工作组在 2025年 的一篇博客文章中指出：“由于 ABAP 是一种闭源的低资源语言，通用商业大语言模型往往难以生成语法正确的代码，也难以遵循 ABAP 云编程的最佳实践。与其在 Python 和 Java 等更常用语言上的能力相比，它们在 ABAP 上的表现较差。” [7]。

除了数据可用性之外，评估生成的 ABAP 程序也面临方法学挑战。由于 ABAP 是一种强类型语言，并且依赖 SAP ABAP运行时环境，因此自动评估语法和语义正确性十分复杂。此外，大语言模型究竟能在多大程度上生成在功能上正确且在实践中有用的 ABAP 代码，仍不明确；尤其是在涉及多个表、事务或逻辑依赖关系的场景中。尽管人工智能辅助的ABAP代码生成具有巨大的经济重要性，但这一领域仍处于起步阶段。例如，现有少数关于 ABAP 代码生成的研究之一显示，其结果有限，而且往往基于少量测试用例或纯人工评估（见第 2 章）。因此，针对大语言模型生成语法和语义均正确的 ABAP 代码的性能开展系统性实证研究，仍存在研究空白。以往的研究未能对生成模型在这一特定领域环境中的可靠性、错误率或局限性给出可信的结论。

1.3 研究目标与研究问题

本研究旨在将基准测试方法应用于 ABAP 代码生成，为实证研究并系统评估大语言模型生成 ABAP 代码的性能奠定基础。借助这一环境，可以考察大语言模型在多大程度上能够生成语法正确、语义准确且具有实际应用价值的 ABAP 程序。研究重点是比较不同模型在正确应用简单语言结构方面的能力，例如内表、数据声明和模块化概念；这些都是 SAP 实际开发场景中的关键要素。生成的程序将依据既定的评估指标进行评估，包括正确性和功能性。为确保评估结果可复现，本研究将构建一个标准化测试环境。此外，还将分析生成代码中的错误，以识别代码生成中的典型弱点和系统性模式。另一个目标是探究能否通过一个将编译器错误消息纳入反馈的迭代过程来改进代码生成。

基于上述目标，本研究将围绕以下研究问题展开：

1. 大语言模型生成语法和语义都正确的 ABAP 代码的能力有多强？2. 加入 ABAP 编译器错误消息能否提高代码质量？3. 哪些编程任务对 ABAP 代码生成构成了特殊挑战?

2 相关工作

近年来，大语言模型在自动代码生成领域的应用日益受到关注。多项研究表明，大语言模型或许能够直接根据描述生成代码，从而让经验较少的开发人员也能通过用自然语言提出需求来创建可运行的程序 [2, 8]。

Chen 等人开发的 HumanEval 已成为系统评估此类模型代码生成能力的核心基准测试方法之一。凭借清晰的结构和广泛的认可度，该框架如今被视为程序合成的标准指标，并常作为比较

不同模型架构的基础 [9]。它由以自然语言表述的编程任务组成，并通过预定义的单元测试进行验证，从而能够客观衡量功能正确性 [3]。

尽管 HumanEval 主要评估大语言模型在代码生成方面的通用能力，但对于本研究而言，关键问题是如何在 SAP 等特定企业环境中运用这类方法。在 SAP 环境中，AWS SAP ABAP 助手或 SAP Joule for Developers 等工具旨在支持根据自然语言提示生成 ABAP 代码，从而拓展大语言模型在实际开发任务中的应用范围 [10]。

Gałazka 和 Szyma´nski 关于 ABAP 代码生成的研究表明，大语言模型不仅能生成简单程序，还能运用现代语言特性创建更复杂的 ABAP 对象 [11]。然而，先前的大部分研究都基于有限的数据集，因此目前只能在有限程度上评估其结果能否推广至规模更大或投入生产的场景。研究还表明，将大语言模型集成到 SAP 系统后，可以部分自动化某些开发任务，例如生成代码工件或测试；而类似聊天机器人的交互形式，则有助于处理单个、特定于上下文的任务 [10, 11]。

这类方法常被提及的一项优势是有望提高效率。根据目前的研究发现，生成式人工智能可以加快从概念描述到可执行代码的转换，并自动完成某些常规任务，例如报告生成 [7, 11]。此外，据报告，AWS SAP 助手或 Joule for Developers 等工具能够支持遗留代码的文档编写和分析，从而有助于缩短项目周期 [10]。为确保生成程序的质量，人们采用了各种技术手段。有关受限解码的研究结果表明，使用 ABAP 语法进行基于令牌的验证，有助于生成语法正确的建议，并提高云任务的成功率 [7]。此外，研究强调，应编写结构良好的提示，以获得对上下文敏感且一致的结果。训练数据的质量和所选模型配置也被视为影响模型性能的关键因素 [11]。

尽管取得了这些进展，仍存在显著局限。研究表明，大语言模型在复杂、新颖或跨系统的编程任务中尤其容易触及性能上限，因此仍然离不开人类专业知识[8, 11]。此外，人们还讨论了与自动生成代码相关的安全问题和伦理问题 [8]。Nvidia Inference Microservices 等方案可以优化延迟和吞吐量，但同时也表明，验证和性能优化仍是将基于大语言模型的代码生成用于生产环境时面临的核心挑战 [7]。

3 方法论

针对大语言模型生成 ABAP 代码的实证研究，其实验流程基于几个核心方面：以包含 180 项任务的基准测试作为数据基础，涵盖通用算法问题（基于 HumanEval）和特定的 SAP 场景。这些任务分为五个主题类别，例如 字符串处理 和 *ABAP* 数据库操作。研究特别关注通过迭代反馈过程提高代码生成成功的概率：每项任务最多经过五轮反馈循环，在此过程中，系统会向大语言模型提供来自 ABAP 编译器或自动化单元测试的反馈。成功与否依据功能正确性进行衡量，并通过这些自动化单元测试加以验证。此外，研究还进行了详细的错误分析和 Kaplan-Meier 生存分析，以评估学习曲线。整个流程完全自动化，采用标准化的 SAP 环境（Docker 镜像），并通过 Python 控制程序和 ADT 接口进行访问。这确保了结果具有可复现性。

图 1 以图示方式展示了基准测试的基本流程。下文将详细介绍该基准测试的结构和方法论。

3.1 基准测试的结构和方法论

开发的基准测试共包含 180 项测试任务，分为两组：164 项基于 HumanEval 数据集 [3], 的标准化任务，该数据集最初用于评估 Python 代码生成；以及 16 项 ABAP 专属任务。这 16 项专属任务源自 SAP 的实际场景，侧重于数据库表的处理。此外，研究还设置了五个任务类别，并由人工将每项任务归入一个或多个类别。字符串处理 类别包含 76 项任务，列表或数组操作 类别包含 108 项，数学计算 类别包含 93 项，逻辑条件类别包含 125 项，而*ABAP* 数据库操作 类别包含 16 项。该基准测试的数据集见附录 A。

这种分类有助于针对不同问题类型，对模型性能进行细致分析，同时体现算法基础和真实的 ABAP 使用场景。测试用例涵盖不同的难度级别和

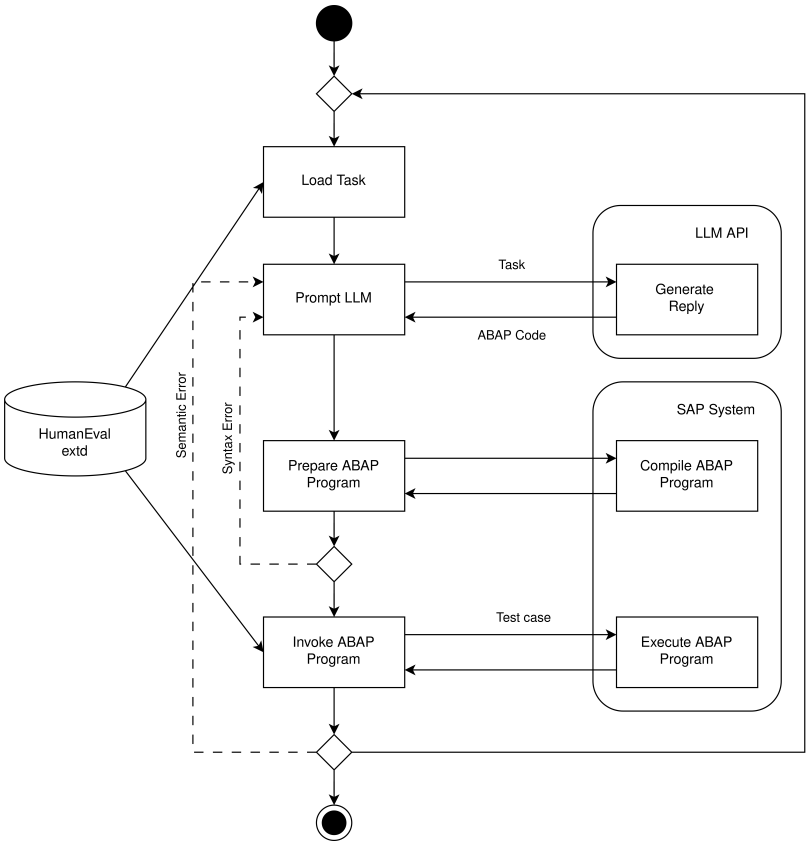


图 1：基准测试的大致流程。

任务类型，从而能够广泛评估这些模型。恒定的系统提示词（见代码示例 1）作为交互的基础，用于明确规范交互、清晰定义任务，并提高生成的答案的一致性。系统提示词的具体内容专门针对 ABAP 制定，以避免实现偏差。制定时考虑了以下方面：要求使用带有静态方法的全局类，因为测试设置要求采用这种结构，且这能确保结果具有可比性。统一定义 SAP 运行时环境的版本，可避免版本差异导致误读或代码生成错误。统一的小数分隔符可确保数值处理一致，因为 ABAP 理论上支持两种形式。采用 RETURNING 参数、每个类仅限一个公共方法，以及命名约定（类名以“Z”开头）等开发规则，以确保测试的统一性、可追溯性和自动化。需要同时提供类定义和实现代码，因为评估时两者缺一不可。只返回代码、不附加解释，可以减少与自动化相关的错误，并便于在测试系统中处理。这些有针对性的规范确保测试执行具有可复现性、有效性和系统可比性。鉴于大语言模型具有随机性，每项任务总共重复十次，以便进行统计上稳健的评估。每个单项任务的求解过程中最多提供五轮反馈循环，让大语言模型有机会修正错误；设定五轮的依据是对开发人员手动提出问题次数的现实估计。

本次基准测试在多个大语言模型上进行。根据模型的不同，输入长度受到限制：对于不具备推理功能的模型，每条消息的最大响应长度设为 5000 个词元；对于推理模型，则提高至 10000 个词元，因为这类模型通常会生成更长的输出。对模型输出进行限制，是为了避免答案无限延伸或过长，同时也为完整解决问题留出足够空间。除 GPT-5 外，所有模型的温度参数均设为 0.2，以提高生成答案的可预测性和一致性。这在代码生成场景中尤为重要，因为稳定且确定的输出有助于验证结果并确保其可复现。不过，温度参数对模型性能及生成内容质量的影响也是一个重要方面，应在后续研究中进行系统分析。由于 OpenAI 的限制，GPT-5 的温度无法调整，因此我们无法对其使用相同的设置。

3.2 大语言模型的选择

本次基准测试选取了若干具有不同特性和侧重点的大语言模型。选择这些大语言模型并非旨在全面测试所有可用模型，而是着重比较选定的开源模型与领先的闭源模型；这些模型都是当前研究中使用最广泛的大语言模型之一。研究目标是探究 ABAP 代码生成场景中的基本扩展效应和架构差异。

研究重点之一是模型的多样性：既纳入了 Llama-3.3-70B-Instruct 和 Codestral-22B 等开源模型，也纳入了 GPT-5 和 Claude-Sonnet-4 等闭源模型，以便比较免费使用的方案与专有方案。此外，还关注了架构和模型规模的多样性：模型范围从约 70 亿参数的较小模型（如 Qwen2.5-Coder），延伸至 GPT-OSS-120B 等大型开源模型。研究还纳入了 GPT-5 和 Claude-Sonnet-4 等专有模型，其参数数量估计要高得多，涵盖了更高的能力层级。这使我们能够得出有关潜在扩展效应及其对生成的答案质量影响的结论。

另一个考量因素是模型的专业化程度。研究既考察了 GPT-5 等通用大语言模型（它们经过训练，能够执行多种任务），也考察了 Qwen3-Coder 等专门针对代码训练的模型。通过这种组合，可以比较普遍适用的模型与专门针对代码生成进行优化的模型。此外，所有使用的模型均有指令微调版本，这对于在真实的提示条件下测试代码生成尤为重要。所选模型中的大多数也属于研究中经常使用的知名大语言模型，这有助于与其他科学研究进行比较。通过使用成熟的模型，可以结合现有研究对结果进行归类和验证，从而提高分析的有效性和可理解性。

最终选定的模型如下：Llama-3.3-70B-Instruct、Codestral-22B、Qwen2.5-Coder 和 Qwen3-Coder、GPT-OSS-20B 和 GPT-OSS-120B，以及 GPT-5-2025-08-07 和 Claude-Sonnet-4-20250514。基准测试评估于2025年8月和9月进行。

3.3 评估环境

为确保可复现性，采用官方 SAP Docker 镜像 *sapse/abap-cloud-developer-trial:2023* 作为基础，该镜像提供了标准化的 ABAP 开发环境。借助该环境，我们可以使用最新的 ABAP Cloud 语言版本 5。镜像未经任何修改，以确保测试基础保持一致且具有可比性。基准测试过程本身使用 Python 实现。Python 控制端与 ABAP 环境之间的通信通过 *ADT* 接口的 Python 实现完成，从而实现对开发环境的程序化访问。与各种大语言模型的通信通过 OpenAI Python 库完成，该库支持通过不同服务提供商连接相应的模型。Claude 是一个例外；要使用批处理 API，必须使用 Claude Python 库。

在单次测试运行中，Python 脚本首先向正在研究的相应大语言模型发送请求，要求其为指定的编程任务生成 ABAP 源代码。随后，系统会在 SAP 服务器上自动创建一个新的 ABAP 类，其名称由模型的回答生成。生成的代码会插入该类中，并保存和激活。接着，系统会创建预定义的 ABAP 单元测试类，用于检查生成的程序代码的功能和正确性。在服务器上执行测试后，结果会返回给 Python 脚本，由脚本进行分析，评估测试运行成功或失败，并在必要时识别根本错误类型。

错误可能发生在不同阶段，例如类创建、激活、语法检查、插入源代码或测试执行期间。所有错误类型都会被系统地记录，并纳入后续评估。如果发生错误，系统会再次向大语言模型发送请求，并利用 SAP 服务器返回的错误消息生成修正后的程序。此请求过程最多可进行五次迭代；若提前得到无错误的解决方案，则会提前结束。SAP 系统通过语法验证以及我们用于语义验证的单元测试来判定是否已得到无错误的解决方案。最后，生成的代码会从开发环境中删除，以免影响后续测试用例。

在实际实现中，这一流程以批处理的形式进行。首先，所有请求都会发送给大语言模型，并将其回答保存在本地；随后在 SAP 服务器上执行这些回答，并将结果返回给 Python。接着便可进行下一批大语言模型请求、执行和评估。这种流程划分将测试用例的生成与执行明确分开，从而在执行时间和稳定性方面带来优势。

3.4 评估标准

对生成的解决方案的评估基于对成功与失败的系统性研究。主要衡量标准是功能正确性，通过在 SAP 服务器上运行 ABAP 自动化单元测试来验证。

为进行更细致的评估，结果将从以下几个方面进行细分：

- 错误分类：如果出现失败，则会确定失败原因，并区分错误发生的阶段：类创建期间、语法检查期间，或仅在执行单元测试期间。此外，还会对语法错误进行详细分析，以归纳模型的典型错误特征（例如声明错误或结构错误）。
- 迭代成功：衡量模型经过多少次反馈迭代后能够给出正确解答。据此可以得出有关学习曲线以及编译器反馈在错误纠正和单元测试成功（即语义正确性）方面效果的结论。
- 模型比较：比较所有受测大语言模型的性能数据，以展示专有架构、开源架构以及专用代码模型之间的差异。
- 任务类别：根据第 3.1 章介绍的五种任务类别分析结果。这种方法不仅可以整体评估模型的优缺点，还能针对特定问题类型进行细分，例如字符串处理、列表或数组操作、数学计算和逻辑条件。尤其需要关注*ABAP* 数据库操作 这一类别，因为这些任务来自实际应用场景，因此对于在 SAP 系统中调整业务逻辑具有很高的实践价值。

对于这些随时间或步骤变化的过程的统计评估，采用 Kaplan-Meier 生存分析[12, 13]。在这里，“生存”被定义为错误状态的持续存在，即未能通过语法检查或最终单元测试。“事件”（在“失败”或“死亡”的意义上）因此对应于代码通过验证。分析以离散的时间步进行，每个时间步对应一次反馈迭代。除了估算生存概率外，该方法还提供 95% 置信区间，从而能够对性能差异的显著性和稳健性做出有依据的评估。Kaplan-Meier 方法尤其适用于此类分析，因为它不仅能够对连续迭代中成功概率的时间变化进行建模，还会将未成功的数据纳入考量 [12, 13]。这一点很重要，因为并非所有测试运行都能在规定的查询次数内成功完成。借助生存曲线，不仅可以评估最终是否成功，还可以评估模型达到正确解的速度。

4 研究

4.1 结果

4.1.1 不同反馈轮次之间的差异

由于 Codestral-22B 在所有反馈轮次中的成功率均为 0%，为确保可读性，图2、图3和图4中未展示其结果。图2和表1概述了其他模型在所有反馈轮次中的成功率。从中已可看出，各模型对 SAP 编译器反馈的反应截然不同。

对各反馈轮次累计成功率的分析展现了迭代式编译器反馈的有效性。根据第3.4节所述的 Kaplan-Meier方法，图3中的生存曲线显示了各模型在六个观测区间（第0至第5轮）内克服错误状态的概率。结果揭示了不同大语言模型之间显著的性能差异，并表明所有模型均能从自动化反馈中受益，而错误纠正的有效性与具体模型版本高度相关。

GPT-5 和 Claude-Sonnet-4 的表现远胜其他模型。Claude-Sonnet-4 在初始轮次（第 0 轮）取得最佳开局，平均成功率为 24.1%；而在随后的反馈轮次中，GPT-5 展现出更出色的学习曲线，在第五轮反馈后达到最高累计成功率 77.1%。这两个模型的很大一部分成功都来自前两轮反馈，其中 GPT-5 的成功率几乎增长至原来的三倍，从第 0 轮的 19.28% 升至第 2 轮的 52.78%。

相比之下，GPT-OSS（120B 和 20B）以及 Qwen模型等开源模型在 ABAP 方面的基础能力较弱，但也能从反馈中获得显著提升。尤其值得注意的是

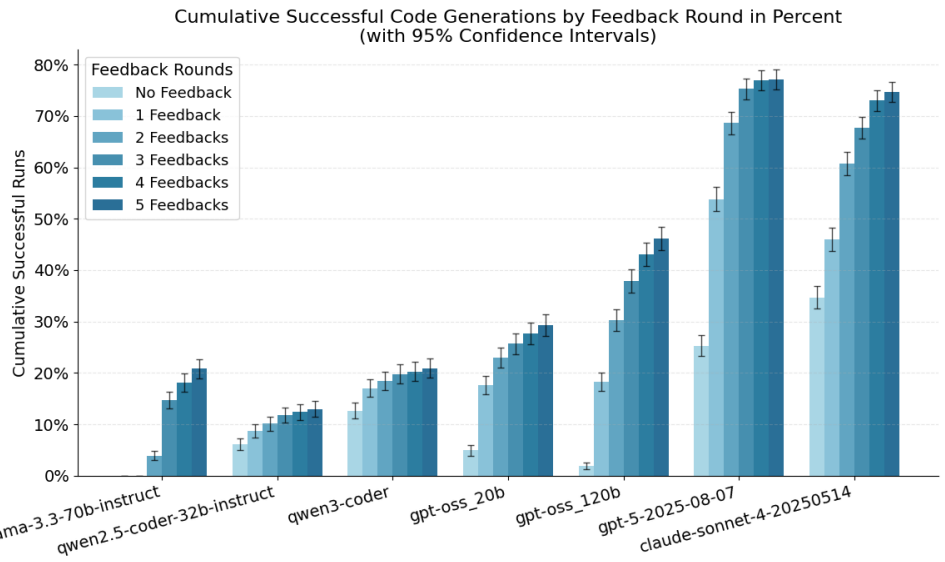


图 2：按模型和反馈轮次细分的基准测试结果（百分比）。

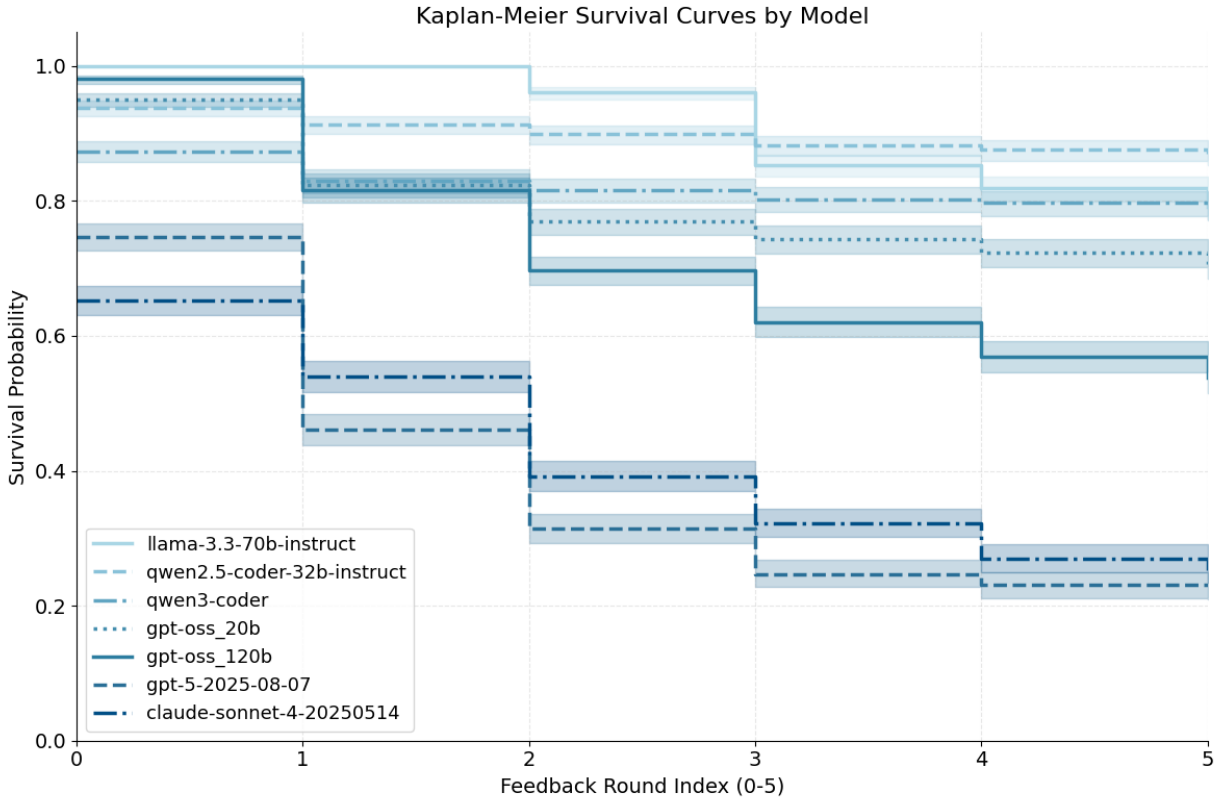


图 3：Kaplan-Meier生存曲线，展示在 0–5 次反馈迭代中仍存在测试错误的概率（基于 10 × 180 次测试运行）

Llama-3.3-70B-Instru ct 在第一轮中未能提交任何有效的代码贡献，但最终达到了 a

在各反馈轮次中，累计成功率达到 20.8%。这凸显了自动反馈机制对于原生领域专业知识较弱的模型的重要性。

图 3 还显示，成功概率的最大提升发生在最初几轮反馈迭代中，尤其是对于能力更强的模型。同时可以看到，在第五次迭代中，成功率仍有可测量的提升。这表明，随着反馈轮次的增加，性能仍有进一步提升的可能，学习曲线尚未完全趋于平缓。

4.1.2 不同任务类型之间的差异

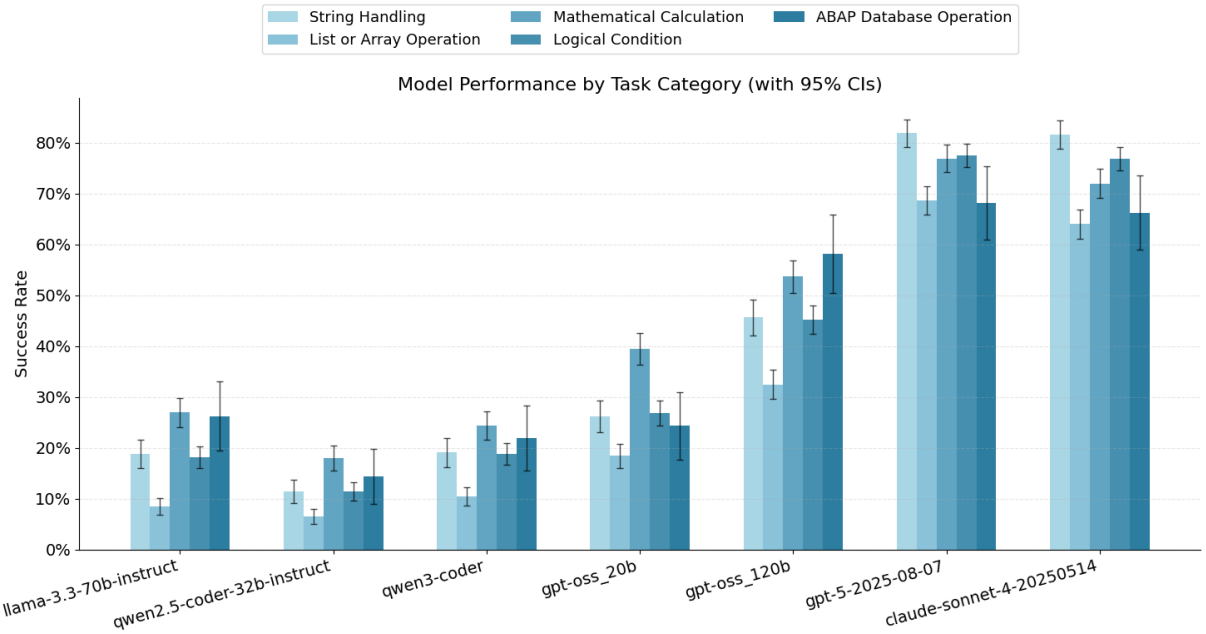


图 4：按模型和任务重点细分的基准测试结果（百分比）。

图 4 和表2 展示了模型在五种任务类别中的成功率，分别为：字符串处理、列表或数组操作、数学计算、逻辑条件以及*ABAP* 数据库操作。结果表明，在大多数类别中，模型的表现都较为稳定，其中 GPT-5 和 Claude-Sonnet-4 尤其能够稳健地应对不同的问题定义。较小的开源模型在数学任务和逻辑检查中更容易出错，而 ABAP 数据库过程并未显现出难度显著增加的情况。图 5 展示了模型在 ABAP 专项子集中的详细表现。这印证了此前观察到的性能差距。

GPT-5 的表现尤其引人注目：尽管其在第 0 轮的起步表现中等，但展现出了极高效的错误后处理能力，最终达到约 0.32 的累积生存概率（对应约 68% 的成功率）。在这一子集中，Claude-Sonnet-4 的初始表现略低于总体平均水平，起始生存概率约为 89%（成功率为 11%），但通过反馈同样逐渐接近领先阵营。另一个值得一提的是 Llama-3.3-70B-Instruct 模型：它在这一复杂子类别中直到第二轮反馈起才取得显著成功，这凸显了迭代反馈机制对于特定领域任务的必要性。

对所有运行中评估模型均无法解决的任务进行分析后发现，列表或数组操作 类别最具挑战性，在 108 个任务中有 19 个未解决。其次是逻辑条件 （125 个任务中有 13 个未解决）和数学计算 （93 个任务中有 11 个未解决）。值得注意的是，在*ABAP* 数据库操作类别中，每个任务都至少被一个模型成功解决（见图6）。未解决任务的完整列表见附录 4。

对表现最强的两个模型 GPT-5 和 Claude-Sonnet-4 按任务进行的成功分布分析，也揭示了双峰模式（见图 7）。经过五次迭代后，模型要么几乎毫无错误地解决任务（

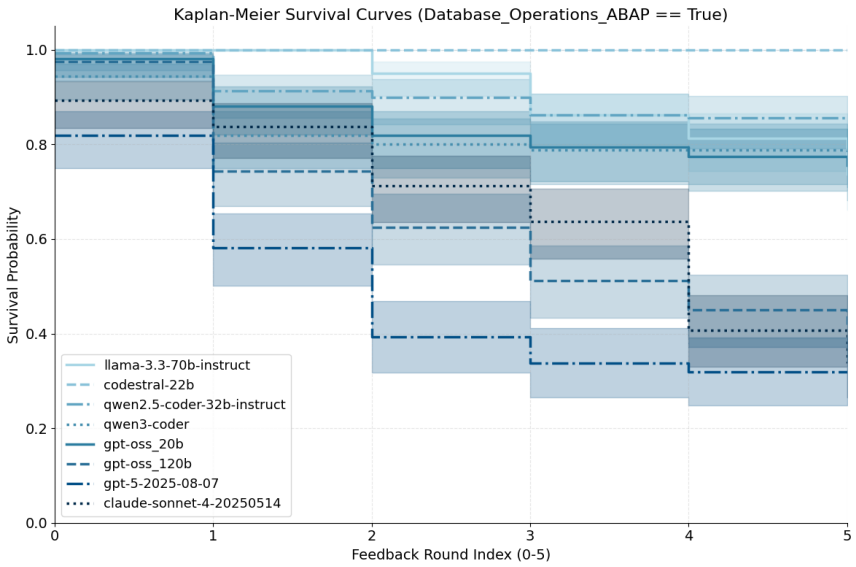


图 5：按大语言模型区分的 ABAP 数据库操作任务的 Kaplan-Meier生存曲线。

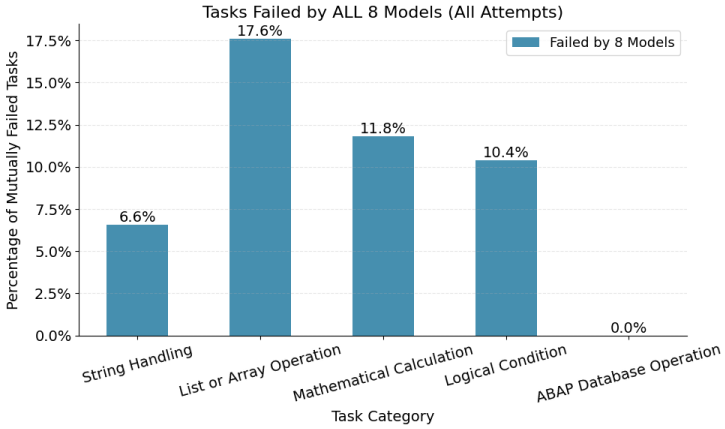


图6：在所有尝试中，八个模型均未能解决的各类别任务所占百分比。

在 91-100% 的情况下），要么几乎完全失败（在 0-10% 的情况下）。只有在极少数情况下，任务的成功率才会处于“中等”水平，即大约一半的尝试能够解决任务。这表明，成功与否在很大程度上取决于具体的任务定义。

4.1.3 错误分布

在 SAP系统中，错误按照固定顺序检测：首先创建类，然后对源代码进行语法检查，最后执行单元测试，如图8所示。

早期验证阶段出现的错误会自动阻止后续检查，因此错误率的解读取决于相应的验证阶段。评估错误时，只区分是否发生错误；尤其是语法错误，可能会同时出现多个，但此处只统计是否存在该类别的错误。所有数值见表3。对生成的 ABAP类进行分析后发现，受检模型呈现出明显不同的错误模式（见图9）。

Codestral-22B 在类创建中的错误率尤其高（82.72%），而语法错误（17.28%）和单元测试错误（0.00%）从未出现。这清楚表明，该模型主要因无法满足基本结构要求而失败。相比之下，GPT-5 生成 ABAP 类时，类创建中的错误率较低（0.00%），而语

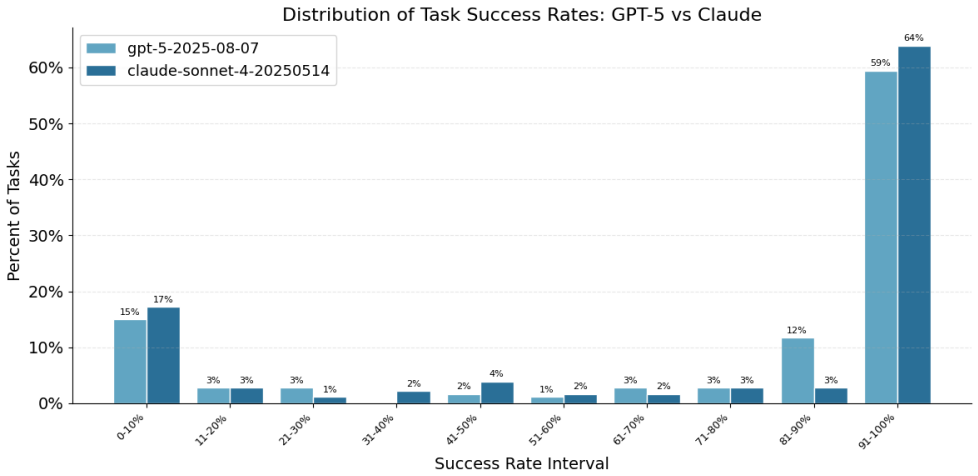


图 7：GPT-5 和 Claude-Sonnet-4 在各任务成功率上的频率分布，展示了双峰成功特征。

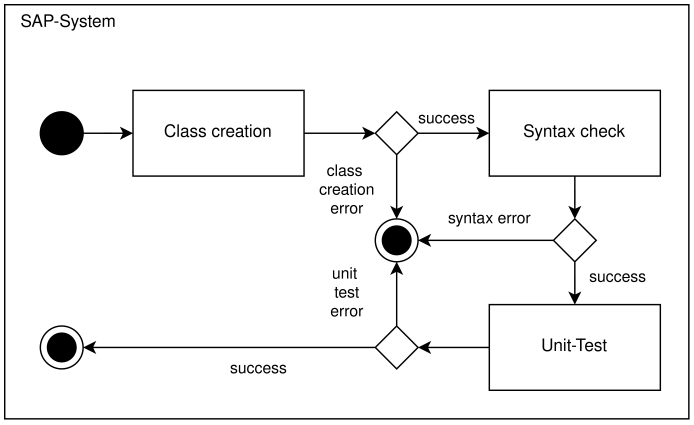


图8：SAP系统中的错误检查流程图。

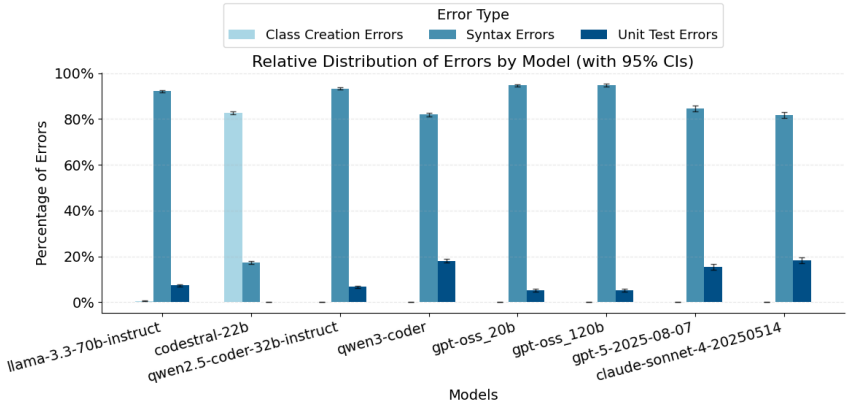


图9：各模型的错误类型分布（百分比）。

法错误率较高（84.69%）。因此，只有较小一部分生成结果能够进入单元测试阶段，此阶段的错误率为 15.31%。这表明 GPT-5 能够生成结构有效的类，但

主要难以保证语法正确性，而单元测试中的语义问题则处于次要地位。其他模型也呈现出类似的错误模式，但其主要失败的验证阶段有所不同。GPT-OSS-120B 和 GPT-OSS-20B 主要在语法检查阶段失败，错误率分别为 94.81% 和 94.78%；其中，GPT-OSS-120B 从未出现类创建错误，而 GPT-OSS-20B 在 0.01% 的案例中出现了类创建错误。Llama-3.3-70B-Instruct 和 Qwen2.5-Coder-32B-Instruct 更频繁地进入测试阶段，分别有 7.40% 和 6.67% 的案例出现单元测试错误。Qwen3-Coder 和 Claude-Sonnet-4 更常在单元测试阶段失败，错误率分别为 18.01% 和 18.22%，表明它们在所需逻辑的语义实现方面面临挑战。

图10展示了对语法错误的详细分析，细分了检查过程中出现的所有语法错误。考虑以下错误类型：

- 词法和词元错误是指代码中涉及无效字符或意外符号的错误。
- 结构和格式错误涉及代码整体布局与组织方面的问题，例如缺少括号或缩进错误。

- 声明错误发生在变量或数据结构声明不正确或缺失时。
- 类型和转换错误与数据类型之间的不匹配或不正确转换有关。
- 面向对象编程特有错误涉及与面向对象编程概念相关的错误，例如类或方法定义错误。

显然，各模型的差异不仅体现在生成错误的数量上，也体现在错误类型上。Llama-3.3-70B-Instruct 生成的词法和词元错误（25.61%）与结构和格式错误（23.34%）所占比例相对均衡。相比之下，Qwen2.5-Coder-32B-Instruct 主要产生词法和词元错误（51.01%），而结构错误仅占 9.91%。Codestral-22B 几乎完全因声明错误（60.76%）和结构问题（38.85%）而失败；几乎不会出现语法错误。GPT-5 的词法错误占中等比例（41.43%），在面向对象编程特有方面的错误率相对较低（1.79%），而类型和转换错误（7.51%）以及声明错误（43.02%）占主导。Claude-Sonnet-4 主要产生类型和转换错误（55.17%），而词法和词元错误仅占 16.62%。完整的语法错误分布见表5。

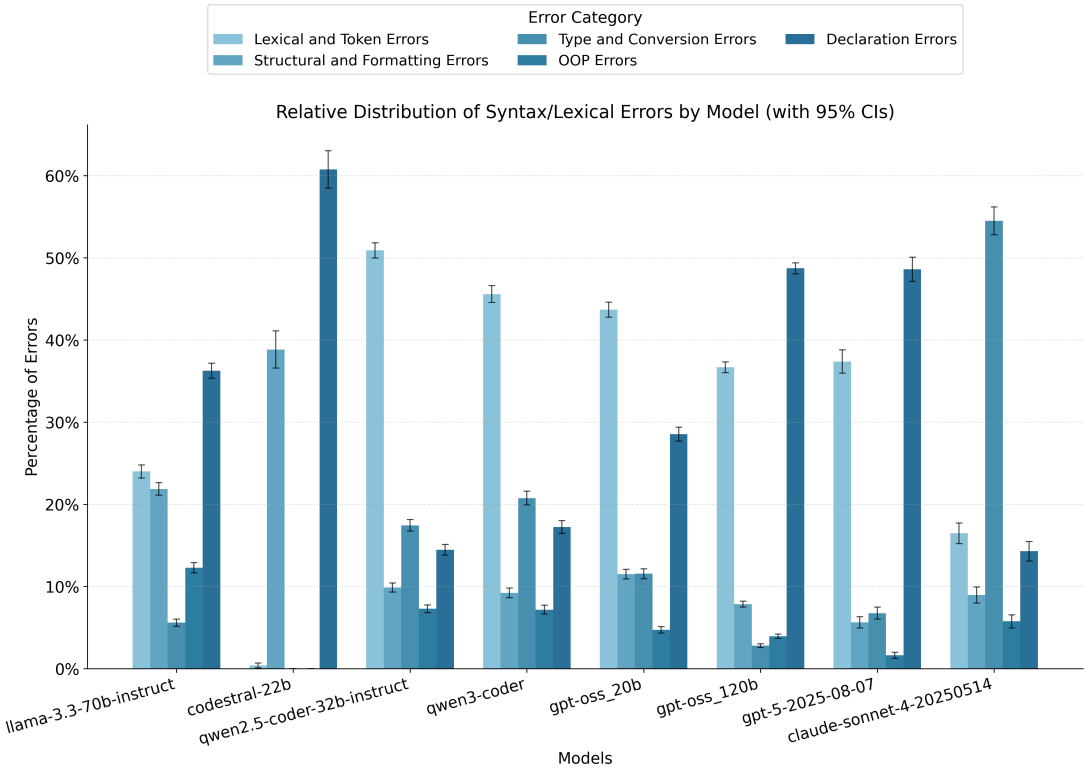


图10：各模型语法错误类别的百分比分布。

这些结果补充了验证阶段分析，并进一步说明，各模型不仅在特定阶段的失败频率不同，在语法和结构方面也呈现出各自的错误特征。

4.2 评估

分析结果表明，大语言模型在 ABAP 代码生成方面具有相当大的潜力。结果很大程度上取决于利用编译器反馈进行迭代改进的能力。尤其是 GPT-5 和 Claude-Sonnet-4 等能力更强的模型，在经过五个反馈周期后，成功率达到 74% 至 77% 以上，并展现出处理错误消息的突出能力。早期迭代中显著的性能提升，例如 GPT-5 的成功率在第二轮从 19.28% 提高到 58.67%，充分说明了这一流程的有效性。
另一个核心发现是，GPT-OSS模型也能够在多轮迭代中取得显著改进。尽管模型复杂度较低，它们仍能借助反馈构建稳健的解决方案，这凸显了适应错误的能力并非单纯取决于模型规模。Qwen-Coder模型在首次迭代中的表现相对较好，但后续几轮几乎没有提升。这种停滞表明，尽管它们能够可靠地复现基本的 ABAP 结构，但利用编译器错误消息中的语义信息进行针对性优化的能力有限。Llama-3.3-70B-Instruct则形成鲜明对比：该模型起初表现较弱，但在迭代过程中显著进步，最终成功率超过 20.83%，这表明它具备一定的基础知识，并能通过迭代反馈将其应用于 ABAP 语法。Codestral-22B模型则表现出一些特殊问题。它无法可靠地遵循所要求的 Markdown代码块格式，也不能始终将说明文字与代码分开。因此，几乎无法清晰提取生成的 ABAP 代码，这大大增加了验证的难度，也限制了答案的可用性。因此，问题与其说出在代码质量本身，不如说是未能遵守正式的提示规范。

对任务类型的分析表明，其难度各不相同：尤其是列表或数组操作 会给模型带来结构性挑战（共有 19 项任务始终未能解决），而至少有一个模型能够解决所有*ABAP* 数据库操作 任务。尽管如此，与模型质量相比，任务类型的影响仍较为有限。GPT-5 和 Claude-Sonnet-4 即使在复杂类别中也表现出色，这表明模型架构和训练数据质量是取得成功的决定性因素。某些模型为何在特定类别中的表现显著优于其他模型，仍有待进一步研究；其中，训练数据中领域知识的深度可能发挥着重要作用。

总体而言，研究表明，在 ABAP项目中成功使用大语言模型，很大程度上取决于模型有效利用编译器反馈的能力。有些模型初始表现不错，但几乎没有进步；另一些模型则尽管起步较弱，仍呈现出明显的学习曲线。同时，结果表明，有些任务即使反复获得编译器反馈，也无法取得实质性改进，这说明模型在解决特定类型问题时存在固有局限。

5 结论与展望

5.1 结论

我们的研究结果表明，基准测试是衡量大语言模型在 ABAP 代码生成方面能力的适当方法。尽管所考察的大语言模型范围有限，ABAP基准测试任务的数量也很少，但结果显示，目前大语言模型在不经过迭代的情况下，生成语法和语义均正确的 ABAP 代码的能力仍然有限。表现最佳的模型 GPT-5 和 Claude-Sonnet-4，在首次请求中给出了正确答案的比例分别为 19.28% 和 24.11%。其他模型的表现远低于此，其中部分模型几乎没有提供任何可用结果，甚至完全没有。因此，对于第一个研究问题，可以得出这样的结论：大语言模型基本上能够生成 ABAP 代码，但不经过迭代时所能达到的质量在很大程度上取决于模型，而且所有系统的质量都远未达到实际应用所需的水平。

对于第二个研究问题，迭代式编译器反馈被证明是提升质量的关键手段，其效果很大程度上取决于模型。领先的 GPT-5 和 Claude-Sonnet-4 在经过五轮迭代后达到了一定的性能水平，使其有望为交互式开发流程提供支持；而其他模型的性能差距则明显更大。尽管取得了可衡量的进步，Llama-3.3-70B-Instruct 和 Qwen模型的成功率仍然较低，严重限制了其实用性。Codestral-22B 的表现尤其令人担忧：由于在遵循提示规范方面存在形式上的缺陷，它在所有轮次中均完全失败。就实际应用而言，这意味着目前尤其是 GPT-5 和 Claude-Sonnet-4 这类模型，结合迭代反馈，能够为 SAP开发人员提供实质性支持。

第三个研究问题探讨了任务类型对 ABAP 代码生成难度的影响。分析表明，某些任务类别对大语言模型构成的挑战明显大于其他类别。尤其是列表或数组操作 表现棘手，共有 19 项任务未能解决，错误率最高，这表明模型在正确映射复杂控制结构和数据结构方面存在困难。相比之下，特定领域的*ABAP* 数据库操作 在所有情况下都至少能由一个模型成功完成。不过，总体而言，与模型之间的性能差异相比，任务类型的影响较为有限。GPT-5 和 Claude-Sonnet-4 等高性能模型即使在复杂类别中也能取得稳定的结果，这表明模型架构和训练数据质量是代码生成成功的决定性因素。因此，尽管任务类型对成功率有可衡量的影响，但与模型之间十分明显的差异相比，其影响较小。因此，LLM 的选择仍然是决定可实现代码质量的关键因素。

尽管通过迭代取得了具有统计显著性的改进，本研究仍揭示出明显的局限：双峰成功模式表明，反馈主要有助于克服语法障碍。然而，如果模型缺乏完成任务所需的基本逻辑理解，即使进行大量迭代，也几乎无法找到解决方案。共有 19 项任务始终未能解决，这表明仍有一大类算法问题尚未得到解决，且所有受测大语言模型都未能掌握这些问题。因此，尽管大语言模型在辅助工作流方面潜力巨大，但目前还不足以支持在 ABAP 环境中可靠地实现完全自主代码生成。

5.2 展望

本研究存在若干局限，在解读结果时必须加以考虑。由于这是一个研究尚不充分的领域，目前关于大语言模型的 ABAP 代码生成的科学研究还很少，这既增加了理论定位的难度，也使得与现有结果的比较更为困难。此外，缺乏成熟的、针对 ABAP 的基准测试，无法开展标准化性能评估。本研究的实证范围也有限，尤其是在需要直接集成真实 SAP 系统和数据库表的任务方面。更全面的评估或许能在这方面提供更多见解；尤其是，未来研究应进一步改进目前有限的任务数量，并扩大针对 ABAP 特定任务的多样性。

第三个研究问题的结果表明，某些任务类型，尤其是涉及复杂数据和控制结构的任务，对大语言模型构成了特殊挑战。然而，在本研究范围内，无法最终明确其原因。因此，未来的研究应进一步探讨这些任务的哪些语言或概念属性导致了所观察到的困难，以及领域专有知识在训练数据中发挥了怎样的作用。此外，温度这一生成参数的影响也是进一步研究的一个重要切入点。本研究采用了固定的温度，因此其对成功分布可能产生的影响尚未得到考察。可以预期，较高的温度会导致结果出现更大的方差，这一点反映在第 7 节所示的成功分布中部的近似值上。

与此同时，这些局限性也体现了该研究领域巨大的发展潜力。大语言模型的持续发展，以及人工智能日益融入 SAP 生态系统（例如通过 Joule for Developers 等工具），表明 ABAP 代码生成的质量将继续提升。尤其是与 SAP系统的直接连接以及对高质量 ABAP 数据的访问，为此提供了巨大潜力。从长远来看，对迭代代码改进流程进行广泛自动化也将成为现实。
基于这些考量，我们提出了若干进一步的研究问题，包括其他大语言模型是否适用于 ABAP 代码生成、如何有针对性地处理任务特定弱点，以及如何在实践中实现完全自动化的基于反馈的代码生成和验证流程。

6 致谢

作者谨此感谢“KISSKI”（“KI-Servicezentrum fürSensibleund Kritische Infrastrukturen”）和“THK-AI Forschungscluster”提供的计算资源，以及 Stefan Breuer 和 Pascal Cerfontaine 提供的组织支持。

参考文献

[1] M.-F. Wong、S. Guo、C.-N. Hang、S.-W. Ho 和 C.-W. Tan，“用于人工智能辅助编程的大型代码自然语言生成与理解：综述”，《熵》，第 25 卷，第 6 期，2023 年。 [在线]。可访问：https://www.mdpi.com/1099-4300/25/6/888[2] H. Yu、B. Shen、D. Ran、J. Zhang、Q. Zhang、Y. Ma、G. Liang、Y. Li、Q. Wang 和 T. Xie，“CodeREval：基于生成式预训练模型的实用代码生成基准测试”，载于 IEEE/ACM 第 46 届国际软件工程大会论文集，ICSE ’24 系列。美国纽约州纽约市：计算机协会，2024 年。 [在线]。可访问：https://doi.org/10.1145/3597503.3623316[3] M. Chen、J. Tworek、H. Jun、Q. Yuan、H. P. d. O. Pinto、J. Kaplan、H. Edwards、Y. Burda、N. Joseph、G. Brockman、A. Ray、R. Puri、G. Krueger、M. Petrov、H. Khlaaf、G. Sastry、P. Mishkin、B. Chan、S. Gray、N. Ryder、M. Pavlov、A. Power、L. Kaiser、M. Bavarian、C. Winter、P. Tillet、F. P. Such、D. Cummings、M. Plappert、F. Chantzis、E. Barnes、A. Herbert-Voss、W. H. Guss、A. Nichol、A. Paino、N. Tezak、J. Tang、I. Babuschkin、S. Balaji、S. Jain、W. Saunders、C. Hesse、A. N. Carr、J. Leike、J. Achiam、V. Misra、E. Morikawa、A. Radford、M. Knight、M. Brundage、M. Murati、K. Mayer、P. Welinder、B. McGrew、D. Amodei、S. McCandlish、I. Sutskever 和 W. Zaremba，“对基于代码训练的大语言模型进行评估”，2021 年 7 月，arXiv:2107.03374 [cs]. [在线]。可访问：http://arxiv.org/abs/2107.03374 [4] J. Cao、Z. Chen、J. Wu、S.-C. Cheung 和 C. Xu，“JavaBench：用于评估大语言模型的面向对象代码生成基准测试”，载于 IEEE/ACM 第 39 届自动化软件工程大会论文集。美国加利福尼亚州萨克拉门托：ACM，2024 年 10 月，第 870–882 页。 [在线]。可访问：https://dl.acm.org/doi/10.1145/3691620.3695470[5]SAP，“SAP——截至 2024 年的全球营收”，2025 年，访问日期：2025 年 12 月 27 日。 [在线]。可访问：
h
tt
ps://de.statista.com/statistik/daten/studie/28261/umfrage/umsatz-des-unternehmens-sap-seit-dem-jahr-2001/ [6] P. M. Färbinger，“SAP 人工智能反乌托邦”，https://e3mag.com/de/sap-ki-dystopie/，2024 年 3 月，《E3杂志》，访问日期：2025 年 12 月 27 日。[7]S. Sohrabij，“SAP 借助受限解码和 NVIDIA NIM 提升 ABAP 代码生成能力”，2025 年 5 月。 [在线]。可访问：https://community.sap.com/t5/technology-blog-posts-by-sap/sap-boosts-abap-code-generation-with-constrained-decoding-and-nvidia-nim/ba-p/14101647[8] J. Jiang、F. Wang、J. Shen、S. Kim 和 S. Kim，“用于代码生成的大语言模型综述”，《ACM 软件工程与方法论汇刊》，2025 年 7 月，已正式接收。 [在线]。可访问：https://doi.org/10.1145/3747588[9]K. Wang、T. Li、X. Zhang、C. Wang、W. Sun、Y. Liu 和 B. Shi，“软件开发生命周期视角：代码大语言模型与智能体基准测试综述”，2025 年。 [在线]。可访问：https://arxiv.org/abs/2505.05283[10] M. Figueiredo，“人工智能之旅即将启程……”，载于《借助 SAP 和 Amazon Bedrock 实现生成式人工智能》。美国加利福尼亚州伯克利：Apress，2025 年，第 375–406 页。 [在线]。可访问：https://link.springer.com/10.1007/979-8-8688-0968-2_7[11] M. Gałazka 和 J. Szyma´ński，“在 SAP ABAP开发中运用人工智能”，载于《人工智能与机器学习》，K. S. Soliman 编。瑞士 Cham：Springer Nature Switzerland，2025 年，第 2300 卷，第 206–217 页，丛书名称：计算机与信息科学通信。 [在线]。可访问：https://link.springer.com/10.1007/978-3-031-79086-7_18[12] E. L. Kaplan 和 P. Meier，“基于不完整观测的非参数估计”，《美国统计学会会刊》，第 53 卷，第 282 期，第 457–481 页，1958 年。[13] D. G. Kleinbaum 和 M. Klein，“Kaplan-Meier生存曲线与 Log-Rank检验”，载于《生存分析：自学教程》，D. G. Kleinbaum 和 M. Klein 编。Springer，2012 年，第 55–96 页。 [在线]。可访问：https://doi.org/10.1007/978-1-4419-6646-9_2

A 附录

所用代码可在 GitHub 上获取：
https://github.com/timkoehne/LLM-Benchmark-ABAP-Code-Generation
所用数据集可在 HuggingFace 上获取：
https://huggingface.co/datasets/timkoehne/LLM-ABAP-Code-Generation-Benchmark

```
你是一位 ABAP 专家开发人员，专门为 SAP 系统生成高质量的 ABAP 类。你编写带有静态方法的全局 ABAP 类。系
```

列表 1：用于大语言模型 ABAP 代码生成的系统提示词

模型	第 0 轮	第 1 轮	第 2 轮	第 3 轮	第 4 轮	第 5 轮
Llama-3.3-70B-Instruct	0.00%	0.00%	3.67%	12.17%	16.89%	20.83%
Codestral-22B	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Qwen2.5-Coder-32B-Instruct	6.00%	8.50%	10.06%	11.72%	12.44%	13.00%
Qwen3-Coder	11.72%	16.22%	17.89%	19.28%	20.17%	20.94%
GPT-OSS-20B	3.11%	12.94%	20.06%	24.39%	26.78%	29.33%
GPT-OSS-120B	1.44%	13.61%	26.33%	35.28%	41.78%	46.17%
GPT-5（2025年-08-07）	19.28%	41.83%	58.67%	68.56%	74.22%	77.11%
Claude-Sonnet-4（2025-05-14）	24.11%	41.50%	52.78%	63.17%	69.83%	74.67%

表1：不同模型在各反馈轮次中的性能提升。

模型	字符串处理	列表或数组操作	数学计算	逻辑条件	ABAP 数据库操作
Llama-3.3-70B-Instruct	18.82%	8.52%	26.99%	18.24%	26.25%
Codestral-22B	0.00%	0.00%	0.00%	0.00%	0.00%
Qwen2.5-Coder-32B-Instruct	11.45%	6.48%	17.96%	11.44%	14.37%
Qwen3-Coder	19.08%	10.46%	24.41%	18.80%	21.88%
GPT-OSS-20B	26.18%	18.43%	39.46%	26.80%	24.37%
GPT-OSS-120B	45.66%	32.50%	53.66%	45.20%	58.13%
GPT-5（2025-08-07）	81.84%	68.61%	76.88%	77.52%	68.13%
Claude-Sonnet-4（2025-05-14）	81.58%	63.98%	71.94%	76.80%	66.25%

表2：不同模型在不同编码任务类别中的性能指标。

模型	类创建错误	语法错误	单元测试错误
Llama-3.3-70B-Instruct	0.47%	92.13%	7.40%
Codestral-22B	82.72%	17.28%	0.00%
Qwen2.5-Coder-32B-Instruct	0.05%	93.29%	6.67%
Qwen3-Coder	0.00%	81.99%	18.01%
GPT-OSS-20B	0.01%	94.78%	5.20%
GPT-OSS-120B	0.00%	94.81%	5.19%
GPT-5-2025-08-07	0.00%	84.69%	15.31%
Claude-Sonnet-4-20250514	0.00%	81.78%	18.22%

表3：不同模型按类型划分的响应错误。

提示 ID	类别
0	列表或数组操作、数学计算、逻辑条件
4	列表或数组操作、数学计算
6	字符串处理、列表或数组操作、数学计算、逻辑条件
8	列表或数组操作，数学计算
20	列表或数组操作，数学计算，逻辑条件
21	列表或数组操作，数学计算
22	列表或数组操作，逻辑条件
26	列表或数组操作，逻辑条件
28	字符串处理，列表或数组操作
34	列表或数组操作
35	列表或数组操作、数学计算、逻辑条件
61	字符串处理、逻辑条件
81	列表或数组操作、逻辑条件
85	列表或数组操作、数学计算、逻辑条件
88	列表或数组操作、数学计算、逻辑条件
90	列表或数组操作
104	字符串处理、列表或数组操作、逻辑条件
129	列表或数组操作、数学计算、逻辑条件
135	列表或数组操作、逻辑条件
145	字符串处理、列表或数组操作、数学计算

表4：所有模型均未通过的提示词 ID 列表。提示词内容见附录 A。

模型	词法与令牌错误	结构与格式错误	类型与转换错误	面向对象编程错误	声明错误
Llama-3.3-70B-Instruct	25.61%	23.34%	6.61%	13.10%	31.34%
Codestral-22B	0.40%	38.85%	0.00%	0.00%	60.76%
Qwen2.5-Coder-32B-Instruct	51.01%	9.91%	17.77%	7.35%	13.97%
Qwen3-Coder	46.00%	9.31%	23.34%	7.24%	14.11%
GPT-OSS-20B	48.11%	12.67%	13.21%	5.30%	20.71%
GPT-OSS-120B	39.33%	8.44%	3.07%	4.46%	44.70%
GPT-5-2025-08-07	41.43%	6.25%	7.51%	1.79%	43.02%
Claude-Sonnet-4-20250514	16.62%	9.07%	55.17%	5.87%	13.27%

表5：各模型的语法错误分类。

Nr	字符串处理	列表或数组操作	数学计算	逻辑条件	ABAP 数据库操作
0		X	X	X	
1	X	X	X	X	
2			X		
3		X	X	X	
4		X	X		
5		X		X	
6	X	X	X	X	
7	X	X		X	

表 6：提示/任务及其类别分配列表。

Nr	字符串处理	列表或数组操作	数学计算	逻辑条件	ABAP 数据库操作
8		X	X		
9		X	X	X	
10	X			X	
11	X	X		X	
12	X	X		X	
13			X	X	
14	X	X			
15	X	X			
16	X				
17	X	X		X	
18	X				
19	X	X		X	
20		X	X	X	
21		X	X		
22		X		X	
23	X				
24			X	X	
25		X	X	X	
26		X		X	
27	X				
28	X	X			
29	X	X		X	
30		X		X	
31			X	X	
32		X	X	X	
33		X	X	X	
34		X			
35		X	X	X	
36	X	X	X	X	
37		X		X	
38	X	X	X	X	
39		X	X	X	
40		X	X	X	
41			X		
42		X	X		
43		X	X	X	
44	X		X	X	
45			X		
46		X	X	X	
47		X	X	X	
48	X			X	
49			X		
50	X	X	X		
51	X				
52		X		X	
53			X		
54	X			X	
55			X		
56	X			X	
57		X		X	
58		X		X	
59			X	X	
60		X	X		
61	X			X	
62		X	X		
63			X		

表 6：提示/任务及其类别分配列表。

Nr	字符串处理	列表或数组操作	数学计算	逻辑条件	ABAP 数据库操作
64	X		X		
65	X		X		
66	X		X	X	
67	X		X		
68		X		X	
69		X		X	
70		X		X	
71			X	X	
72		X	X	X	
73		X		X	
74	X	X		X	
75			X	X	
76			X	X	
77			X	X	
78	X		X		
79	X		X		
80	X			X	
81		X		X	
82	X		X	X	
83			X		
84	X		X		
85		X	X	X	
86	X				
87		X		X	
88		X	X	X	
89	X				
90		X			
91	X			X	
92			X	X	
93	X			X	
94		X	X		
95	X			X	
96		X	X	X	
97			X		
98	X			X	
99	X			X	
100		X	X		
101	X	X		X	
102				X	
103	X		X		
104	X	X		X	
105	X	X		X	
106		X	X	X	
107	X			X	
108	X	X		X	
109		X		X	
110		X		X	
111	X				
112	X			X	
113	X	X			
114		X	X	X	
115		X		X	
116	X	X	X	X	
117	X	X		X	
118	X			X	
119	X			X	

表 6：提示/任务及其类别分配列表。

Nr	字符串处理	列表或数组操作	数学计算	逻辑条件	ABAP 数据库操作
120		X			
121		X	X	X	
122	X	X	X	X	
123		X	X	X	
124	X			X	
125	X	X		X	
126		X		X	
127			X	X	
128		X	X	X	
129		X	X	X	
130		X	X	X	
131	X		X	X	
132	X			X	
133		X	X		
134	X			X	
135		X		X	
136		X			
137	X			X	
138			X	X	
139			X		
140	X			X	
141	X			X	
142		X	X	X	
143	X		X	X	
144	X		X	X	
145	X	X	X		
146		X		X	
147		X	X	X	
148		X	X	X	
149	X	X			
150			X	X	
151		X	X	X	
152		X	X		
153	X	X		X	
154	X			X	
155	X	X		X	
156	X		X	X	
157			X	X	
158	X	X		X	
159		X	X	X	
160		X	X	X	
161	X				
162	X				
163		X		X	
erp_000					X
erp_001		X			X
erp_002		X			X
erp_003		X			X
erp_004				X	X
erp_005		X	X	X	X
erp_006		X	X	X	X
erp_007		X	X	X	X
erp_008	X	X		X	X
erp_009		X	X	X	X
erp_010		X		X	X
erp_011		X	X	X	X

表 6：提示/任务及其类别分配列表。

Nr	字符串处理	列表或数组 操作	数学 计算	逻辑条件	ABAP 数据库 操作
erp_012		X	X		X
erp_013		X		X	X
erp_014		X		X	X
erp_015		X	X		X

表 6：提示/任务及其类别分配列表。