

一种代码生成方法及相关设备

5 本申请要求于 2023 年 12 月 14 日提交国家知识产权局、申请号为 202311723682.4、发明名称为“一种代码生成方法及相关设备”的中国专利申请的优先权，以及要求于 2024 年 03 月 15 日提交国家知识产权局、申请号为 202410305635.6、发明名称为“一种代码生成方法及相关设备”的中国专利申请的优先权，其全部内容通过引用结合在本申请中。

技术领域

10 本申请涉及人工智能 (artificial intelligence, AI) 技术领域，尤其涉及一种代码生成方法、代码开发平台、计算设备集群、计算机可读存储介质以及计算机程序产品。

背景技术

15 随着软件规模和复杂度的增加，越来越多的开发者尝试使用代码生成 (Code Generation) 技术进行软件开发。代码生成技术致力于减少开发者的人工编程工作量、提高代码研发效率，备受软件工程 (Software Engineering, SE) 学术界和工业界的广泛关注。近年来，得益于人工智能 (artificial intelligence, AI) 研究在自然语言处理方面的进展和在大语言模型 (large language model) 方面的突破，大语言模型相关成果促使代码生成技术从学术研究阶段逐步迈向实际落地应用阶段，各类基于大语言模型的智能编程助手产品不断涌现。

20 智能编程助手产品主要聚焦于文本代码转换 (Text2Code) 场景，用于根据开发者以自然语言描述的需求生成实现上述需求的代码。具体地，开发者在编写代码的过程中完成代码功能注释的编写，并触发代码生成。然后，智能编程助手产品可以根据开发者提供的注释及相关信息利用生成式预训练转换模型 (Generative Pre-trained Transformer, GPT) 生成实现了注释所描述功能的代码片段。智能编程助手产品可以以推荐的形式将代码片段呈现给开发者，让开发者决定接受或拒绝其推荐，或接受后再进行进一步修改。

25 然而，在真实应用场景的代码开发项目中，业务逻辑代码生成效果较差，生成代码中包括完全不存在的方法或变量，上述不存在的方法或变量也称作幻觉，如此导致代码接受率较低，难以满足业务需求。

发明内容

30 本申请提供了一种代码生成方法，该方法赋予语言模型自主调用语言服务等外部工具的能力，使得语言模型在推理的过程中能够模拟人类开发者使用语言服务辅助语言模型进行代码生成，缓解甚至杜绝语言模型在代码生成过程中产生的幻觉问题，提升智能编程助手在实际开发场景中的代码生成可用性和准确率。本申请还提供了与上述方法对应的代码开发平台、计算设备集群、计算机可读存储介质以及计算机程序产品。

35 第一方面，本申请提供一种代码生成方法。该方法可以由代码开发平台执行。代码开发平台可以是软件，该软件可以是独立运行的软件，或者是集成于其他软件，例如是集成开发环境内的功能模块或者是集成开发环境的插件，或者是代码编辑器。软件可以部署在计算设备集群，计算设备集群执行软件系统的程序代码，从而执行本申请的代码生成方法。在一些示例中，代码开发平台也可以是硬件，例如是提供代码生成能力的计算设备集群，计算设备集群运行时，执行本申请的代码生成方法。

40 具体地，代码开发平台接收代码生成任务，该代码生成任务用于根据用户的输入信息进行代码生成。然后代码开发平台根据代码生成任务提取代码上下文。其中，代码上下文包括项目中与待生成代码相关的代码片段。代码开发平台根据代码上下文，通过语言模型确定是否调用语言服务的应用程序编程接口 (application programming interface, API)。当确定调用语言服务的 API，代码开发平台可以向语言服务发送 API 调用请求，获得语言服务的 API 调用响应。其中，语言服务用于辅助生成代码，API 调用响应包括语言服务的至少一个推荐结果。代码开发平台根据代码上下文和语言服务的 API 调用响应，通过语言模型推理以获得生成代码。

45

该方法赋予语言模型自主调用语言服务等外部工具的能力，使得语言模型在推理的过程中能够模拟人类开发者使用语言服务如集成开发环境的自动补全、提示推荐、跳转查看等功能辅助该语言模型进行代码生成，缓解甚至杜绝语言模型在代码生成过程中产生的幻觉问题，进而避免生成代码直接造成 IDE 中的报错，最终提升智能编程助手在实际开发场景中的代码生成可用性和准确率。

5 在一些可能的实现方式中，代码开发平台可以根据代码上下文，通过语言模型进行代码生成得到当前结果。若当前结果中包括 API 调用标识符，确定调用语言服务的 API。该方法通过基于代码上下文，通过语言模型进行代码生成，然后根据语言模型输出的当前结果是否包括 API 调用标识符，从而决策是否调用语言服务的 API，从而实现支持无 API 调用的代码生成或基于 API 调用的代码生成，在保障代码生成质量的前提下，提升了兼容性。

10 在一些可能的实现方式中，代码开发平台还可以通过语言模型预测是否请求用户协助。其中，用户协助是指用户参与代码生成过程，为语言模型进行代码生成提供帮助。若是，代码开发平台可以接收用户的响应，用户的响应包括用户对 API 调用响应的选择结果。

该方法通过引入用户协助，让用户选择从语言服务的返回结果中选择一个或多个结果，模拟类似人类开发者编程过程中从 IDE 补全列表中选择的过程，从而达到人机协作结对编程的效果，由此可以进一步提升代码生成质量，提高代码接受率。

15 在一些可能的实现方式中，代码开发平台可以获取语言模型在未加入 API 调用响应的困惑度和加入 API 调用响应的困惑度。其中，困惑度用于度量一个概率分布或概率模型预测样本的好坏程度。低困惑度的概率模型能更好地预测样本。具体到本申请，概率模型可以是语言模型，因此，困惑度可以用于度量语言模型预测代码的好坏程度。通常情况下，困惑度越小，表示语言模型越准确，代码生成效果越好。当困惑度的差值大于阈值时，代码开发平台可以确定请求用户协助。

20 如此，可以通过量化的困惑度来度量语言模型预测代码的好坏程度，进而决策是否请求用户协助，从而为语言模型进行代码生成提供增益。

在一些可能的实现方式中，代码开发平台可以检测语言模型的生成结果中是否包括请求用户协助的标识符。若是，代码开发平台确定请求用户协助。该方法支持代码开发平台根据语言模型的生成结果自主选择是否请求用户协助，在保障代码生成质量的前提下，提升了兼容性和可用性。

25 在一些可能的实现方式中，代码开发平台还可以根据代码上下文，确定语言服务的 API 调用的类型、参数或时机中的至少一项。相应地，代码开发平台可以根据语言服务的 API 调用的类型或参数，生成 API 调用请求，然后根据时机，向语言服务发送 API 调用请求。

30 在该方法中，代码开发平台可以动态地根据代码上下文选择 API 调用的类型、参数、时机，而不局限于选取的一到两种固定的语言服务，且可以由语言模型自主决定调用时机，更贴合真实的代码开发场景，由此可以提升代码生成质量。

在一些可能的实现方式中，代码开发平台还可以获取训练集，训练集中包括携带 API 调用的训练数据，携带 API 调用的训练数据包括项目级的代码上下文样本、API 调用响应样本和预测代码样本。然后代码开发平台可以根据训练集对基座模型进行微调，获得语言模型。

35 该方法通过采用携带 API 调用的训练数据对基座模型进行微调，可以实现训练与推理过程对齐，相比于仅在推理阶段用外部干预影响模型行为的方式，例如是只在推理阶段通过外部介入的方式用语言服务/静态分析方法试图影响大模型的单步采样概率，本申请在模型训练阶段即在训练数据中加入工具调用行为，通过训练可以使得语言模型更符合预期行为，赋予语言模型自主获取外部信息、减少幻觉的能力。

40 在一些可能的实现方式中，训练集中还包括未携带 API 调用的训练数据，未携带 API 调用的训练数据包括代码上下文样本和预测代码样本。相应地，代码开发平台可以根据携带 API 调用的训练数据和未携带 API 调用的训练数据，对基座模型进行混合微调，获得语言模型。

通过使用未携带 API 调用的训练数据进行微调，可以保持代码生成能力，通过使用携带 API 调用的训练数据进行微调可以赋予 API 调用能力、外部信息(如 API 调用响应)的利用能力，具有较高可用性。

在一些可能的实现方式中，训练数据通过如下方式收集：

45 代码开发平台根据基于人类开发者调用语言服务的场景构造的启发式规则，生成训练数据；或者，代码开发平台根据集成开发环境 IDE 的交互事件记录，构造训练数据；或者，

代码开发平台根据用户的标注信息，构造训练数据。

其中，代码开发平台根据基于人类开发者调用语言服务的场景构造的启发式规则，生成训练数据，可以实现以较低成本构建语言模型的基本能力。代码开发平台根据集成开发环境 IDE 的交互事件记录，构造训练数据，如此可以通过收集人类开发者的真实行为，提高训练数据的可靠性和多样性，以提高语言模型的泛化能力。代码开发平台根据用户标注代码发生幻觉的标注信息，构造训练数据，可以强化幻觉高频发生场景下语言模型的应对能力。

在一些可能的实现方式中，语言服务包括辅助编程服务。其中，辅助编程服务用于辅助代码开发，例如辅助人类开发者开发代码，或者是辅助语言模型生成代码。例如，语言服务可以是 IDE 中的代码补全服务、提示推荐服务、跳转查看服务。该方法不局限于特定的语言服务类型，可以支持调用多种辅助编程服务，以辅助语言模型生成代码，提高代码生成质量。

在一些可能的实现方式中，代码开发平台可以根据代码上下文和语言服务的 API 调用请求、语言服务的 API 调用响应，通过语言模型推理以获得生成代码。该方法以代码上文-语言服务的 API 调用请求-语言服务的 API 调用响应的结构构建输入给语言模型的提示信息，模拟人类开发者开发时的流程，不仅能让语言模型自主权衡，综合利用上下文信息和语言服务的信息，同时也能聚焦当前的关键信息，提高语言模型的预测质量，缓解幻觉问题。

第二方面，本申请提供一种代码开发平台。代码开发平台包括：

交互模块，用于接收代码生成任务，代码生成任务用于根据用户的输入信息进行代码生成；

代码上下文处理模块，用于根据代码生成任务提取代码上下文，代码上下文包括项目中与待生成代码相关的代码片段；

推理模块，用于根据代码上下文，通过语言模型确定是否调用语言服务的应用程序编程接口 API；

代码上下文处理模块，还用于当确定调用语言服务的 API，向语言服务发送 API 调用请求，获得语言服务的 API 调用响应，语言服务用于辅助生成代码，API 调用响应包括语言服务的至少一个推荐结果；

推理模块，还用于根据代码上下文和语言服务的 API 调用响应，通过语言模型推理以获得生成代码。

在一些可能的实现方式中，推理模块具体用于：

根据代码上下文，通过语言模型进行代码生成得到当前结果；

若当前结果中包括 API 调用标识符，确定调用语言服务的 API。

在一些可能的实现方式中，推理模块还用于：

通过语言模型预测是否请求用户协助；

交互模块，还用于若是，接收用户的响应，用户的响应包括用户对 API 调用响应的选择结果。

在一些可能的实现方式中，推理模块具体用于：

获取语言模型在未加入 API 调用响应的困惑度和加入 API 调用响应的困惑度；

当困惑度的差值大于阈值时，确定请求用户协助。

在一些可能的实现方式中，推理模块具体用于：

检测语言模型的生成结果中是否包括请求用户协助的标识符；

若是，确定请求用户协助。

在一些可能的实现方式中，代码上下文处理模块还用于：

根据代码上下文，确定语言服务的 API 调用的类型、参数或时机中的至少一项；

代码上下文处理模块具体用于：

根据语言服务的 API 调用的类型或参数，生成 API 调用请求；

根据时机，向语言服务发送 API 调用请求。

在一些可能的实现方式中，代码开发平台还包括：

训练模块，用于获取训练集，训练集中包括携带 API 调用的训练数据，携带 API 调用的训练数据包括项目级的代码上下文样本、API 调用响应样本和预测代码样本；

训练模块，还用于根据训练集对基座模型进行微调，获得语言模型。

在一些可能的实现方式中，训练集中还包括未携带 API 调用的训练数据，未携带 API 调用的训练数据包括代码上下文样本和预测代码样本；

训练模块具体用于：

根据携带 API 调用的训练数据和未携带 API 调用的训练数据，对基座模型进行混合微调，获得语言模型。

在一些可能的实现方式中，训练模块具体用于：

- 5 根据基于人类开发者调用语言服务的场景构造的启发式规则，生成训练数据；或者，
根据集成开发环境 IDE 的交互事件记录，构造训练数据；或者，
根据用户的标注信息，构造训练数据。

在一些可能的实现方式中，语言服务包括辅助编程服务。

- 10 第三方面，本申请提供一种计算设备集群。所述计算设备集群包括至少一台计算设备，所述至少一台计算设备包括至少一个处理器和至少一个存储器。所述至少一个处理器、所述至少一个存储器进行相互的通信。所述至少一个处理器用于执行所述至少一个存储器中存储的指令，以使得计算设备或计算设备集群执行如第一方面或第一方面的任一种实现方式所述的代码生成方法。

第四方面，本申请提供一种计算机可读存储介质，所述计算机可读存储介质中存储有指令，所述指令指示计算设备或计算设备集群执行上述第一方面或第一方面的任一种实现方式所述的代码生成方法。

- 15 第五方面，本申请提供了一种包含指令的计算机程序产品，当其在计算设备或计算设备集群上运行时，使得计算设备或计算设备集群执行上述第一方面或第一方面的任一种实现方式所述的代码生成方法。
本申请在上述各方面提供的实现方式的基础上，还可以进行进一步组合以提供更多实现方式。

附图说明

- 20 为了更清楚地说明本申请实施例的技术方法，下面将对实施例中所需使用的附图作以简单地介绍。
图 1 为本申请提供的一种代码开发平台的架构示意图；
图 2 为本申请提供的一种流失推理的流程图；
图 3 为本申请提供的一种多任务混合微调的示例图；
图 4 为本申请提供的一种代码生成方法的流程图；
25 图 5 为本申请提供的一种代码编辑界面的示意图；
图 6 为本申请提供的一种推理阶段代码生成的交互示意图；
图 7 为本申请提供的一种训练语言模型的流程图；
图 8 为本申请提供的一种训练数据的代码格式示意图；
图 9 为本申请提供的一种包括链式 API 调用的训练数据的代码格式示意图；
30 图 10 为本申请提供的一种代码开发平台的结构示意图；
图 11 为本申请提供的一种计算设备的结构示意图；
图 12 为本申请提供的一种计算设备集群的结构示意图；
图 13 为本申请提供的另一种计算设备集群的结构示意图；
图 14 为本申请提供的又一种计算设备集群的结构示意图。

具体实施方式

本申请实施例中的术语“第一”、“第二”仅用于描述目的，而不能理解为指示或暗示相对重要性或者隐含指明所指示的技术特征的数量。由此，限定有“第一”、“第二”的特征可以明示或者隐含地包括一个或者更多个该特征。

- 40 首先对本申请实施例中所涉及到的一些技术术语进行介绍。

代码生成 (Code Generation)，是指根据用户的输入信息，例如是不完整的代码、自然语言描述，通过自动化工具或技术，自动生成代码，以使代码完整或者实现自然语言描述所描述的功能。根据生成代码粒度不同，代码生成可以包括行级代码生成、方法级代码 (或称为函数级代码) 生成。

- 45 大语言模型 (Large Language Model, LLM)，是一种语言模型，由具有许多参数 (通常数十亿个权重或更多) 的人工神经网络组成，使用自监督学习或半监督学习对大量未标记文本进行训练。根据模型架构不同，大语言模型可以分为不同类型。在代码生成领域应用比较广泛的大语言模型包括生成式预训

练转换模型 (Generative Pre-trained Transformer, GPT)。

基于 GPT 等大语言模型的智能编程助手产品支持根据开发者提供的注释及相关信息生成实现注释所描述功能的代码片段。公测阶段的用户反馈及大规模调查结果表明,上述智能编程助手产品能够有效降低开发者在实际编写代码与搜索知识、查阅文档、寻找可复用组件等活动之间频繁切换带来的成本,从而提高了软件开发的效率。然而,在真实应用场景的代码开发项目中,业务逻辑代码生成效果较差,具体表现为生成代码中含有完全不存在的方法或变量 (AI 领域通常称之为幻觉)。

发明人经过研究发现,智能编程助手产品中的代码生成模型通常基于 GPT 等 LLM 训练得到,代码生成模型的训练和推理方式源自于自然语言处理 (Natural Language Processing, NLP) 技术,未充分考虑代码项目的逻辑结构及代码文件内容之间普遍存在的调用关系。具体而言:

在模型训练阶段,LLM 对训练语料的处理是以自然语言方式顺序处理,与实际代码开发的逻辑习惯不符。而且,LLM 采用的训练方式是基于文本形式的前文-后文自回归式训练,没有为模型引入上下文内容以及调用外部工具 (例如是自动补全、提示推荐等工具) 的能力。

在模型推理阶段,基于自回归训练的 LLM 可感知的上下文范围天然局限于当前文件的文本内容,生成代码中容易包含对项目不存在代码的调用,却无法正确调用项目内已存在的代码。

由此可见,与其他软件开发工具相比,基于 LLM 的智能编程助手产品仍处于起步阶段。如何消除或缓解代码生成的幻觉仍需要在实践中进行持续的探索。为消除或缓解代码生成中的幻觉,业界提出了不同优化方案。

一种典型的优化方案是使用全局上下文来引导代码的语言模型的方法,该方法通过使用监视器 (monitors) 来对语言模型进行指导。具体地,该方案利用静态分析生成代码补全建议,并通过改变语言模型输出的概率分布来实现代码生成的约束。实验结果表明,该方法可以显著提高代码生成的质量。

然而,该方法通过规则定义引入 monitors,实验中仅支持了少数情况或场景。而且,由于预训练时的数据组织和训练目标定义方式,仅依赖大模型本身无法判断上下文是否充足和真实,已有方法无法动态、自主、即时地引入外部信息,预防幻觉问题的发生。

有鉴于此,本申请提供一种代码生成方法。该方法旨在赋予语言模型 (如 LLM) 自主调用软件开发工具 (也可以统称为语言服务),例如是软件开发工具通过应用程序编程接口 (application programming interface, API) 提供的能力,以缓解当前智能编程助手在代码生成方面的局限性,创新性地提出了一种通过让语言模型自主调用语言服务解决代码生成幻觉的方法。该方法受人类开发者编程过程启发,旨在模拟人类在集成开发环境 (Integrated Development Environment, IDE) 中编写代码的过程 (即人类开发者的编程现场),通过训练让语言模型通过调用语言服务等工具来使用 IDE (或其他辅助编程服务,统称为语言服务) 来辅助自己“编写”代码。具体而言,通过构建包含语言服务调用行为的训练集,可以通过工具学习 (Tool Learning, 一种让语言模型学会生成工具调用的训练方法) 训练语言模型自主调用语言服务的能力,让语言模型在推理的过程中能够模拟人类使用 IDE 的自动补全、提示推荐、跳转查看等功能辅助自己进行代码生成,甚至允许人类开发者帮助语言模型从语言服务提供的信息中进行选择,从而有效地缓解幻觉现象导致的代码生成实用性和接受率低的问题。

其中,代码生成方法可以由代码开发平台执行。代码开发平台可以是软件,该软件可以是独立运行的软件,或者是集成于其他软件,例如是 IDE 内的功能模块或者是集成于 IDE 的插件,或者是代码编辑器。软件可以部署在计算设备集群,计算设备集群执行软件系统的程序代码,从而执行本申请的代码生成方法。其中,软件可以是以软件包的形式提供给用户,例如软件可以是以客户端代码编辑器或 IDE 的新功能,随着版本更新迭代提供给用户,或者是以基于预训练语言模型的代码生成插件的新特性,随着版本更新迭代提供给用户。用户可以在本地数据中心或者私有云运行软件包,以部署软件。或者,软件可以以云服务,例如是软件即服务 (software as a service, SaaS) 的形式提供给用户使用。例如,软件可以是以云端代码编辑器或开发环境的辅助编码功能出现,通过云服务的形式对外暴露功能接口,例如是应用程序编程接口 (application programming interface, API),其他工具可通过调用接口的方式使用上述代码生成功能或能力。在一些可能的实现方式中,代码开发平台也可以是硬件,例如是提供代码生成能力的计算设备集群,计算设备集群运行时,执行本申请的代码生成方法。

具体地,代码开发平台可以接收代码生成任务,代码生成任务用于根据用户的输入信息进行代码生

成，代码开发平台根据代码生成任务提取代码上下文，代码上下文包括项目中与待生成代码相关的代码片段。然后代码开发平台根据代码上下文，通过语言模型确定是否调用语言服务的应用程序编程接口 API。当确定调用语言服务的 API，代码开发平台向语言服务发送 API 调用请求，获得语言服务的 API 调用响应。其中，语言服务用于辅助生成代码，API 调用响应可以包括语言服务的至少一个推荐结果，例如是自动补全服务的至少一个补全结果、提示推荐服务的至少一个提示或者跳转查看服务推荐的至少一段代码。代码开发平台根据代码上下文和语言服务的 API 调用响应，通过语言模型推理以获得生成代码。

该方法赋予语言模型自主调用语言服务等外部工具的能力，使得语言模型在推理的过程中能够模拟人类开发者使用语言服务如 IDE 的自动补全、提示推荐、跳转查看等功能辅助该语言模型进行代码生成，缓解甚至杜绝语言模型在代码生成过程中产生的幻觉问题，进而避免生成代码直接造成 IDE 中的报错，最终提升智能编程助手在实际开发场景中的代码生成可用性和准确率。

为了使得本申请的技术方案更加清楚、易于理解，下面先对本申请的代码开发平台的系统架构进行介绍。

参见图 1 所示的一种代码开发平台的架构示意图，代码开发平台 10 包括推理平台 100，推理平台 100 用于基于训练好的语言模型进行推理，从而获得生成代码。进一步地，代码开发平台 10 还可以包括训练平台 200，训练平台 200 用于基于携带 API 调用的训练数据，对基座模型进行微调，获得语言模型。该语言模型是用于生成代码的模型，也称作代码生成模型。代码生成模型可以为大模型如 LLM，基于此，代码生成模型可以为代码生成大模型，或称作代码大模型、生成式代码大模型。

其中，推理过程在推理阶段进行，训练过程在训练阶段进行。下面分别对推理平台 100 的推理过程、训练平台 200 的训练过程进行详细说明。

推理平台 100 用于接收代码生成任务，代码生成任务用于根据用户的输入信息进行代码生成，然后根据代码生成任务提取代码上下文，代码上下文包括项目中与待生成代码相关的代码片段。例如，推理平台 100 可以包括代码上下文处理模块，代码上下文处理模块用于接收代码生成任务，根据代码生成任务提取代码上下文。推理平台 100 还用于根据代码上下文，通过语言模型确定是否调用语言服务的 API。在图 1 的示例中，语言模型可以为生成式代码大模型，语言服务可以为辅助编程服务，如 IDE 中的代码补全服务、提示推荐服务、跳转查看服务，语言服务用于辅助生成代码。例如，推理平台 100 可以包括推理服务，代码上下文处理模块可以将代码上下文提供给推理服务，推理服务可以根据代码上下文，调用生成式代码大模型进行推理，以确定是否调用辅助编程服务的 API。具体地，生成结果中包括 API 调用标识符(token)，例如是 ls_request 或 ls_req，其中，ls 代表语言服务器 language server，req 代表 request，则表示需要调用语言服务的 API。

推理平台 100 还用于当确定调用语言服务的 API，向语言服务发送 API 调用请求，获得语言服务的 API 调用响应，根据代码上下文和语言服务的 API 调用响应，通过语言模型推理以获得生成代码。其中，API 调用响应包括语言服务的至少一个推荐结果。进一步地，为了提升语言模型的性能，推理平台 100 可以根据代码上下文、语言服务的 API 调用请求和语言服务的 API 调用响应，通过语言模型推理以获得生成代码。如图 1 所示，推理平台 100 中的代码上下文处理模块可以向辅助编程服务发送 API 调用请求，获得辅助编程服务返回的 API 调用响应，上下文处理模块可以将代码上下文和 API 调用请求、API 调用响应拼接，获得新的代码上下文，以进行流式推理循环。具体地，上下文处理模块可以将新的上下文提供给推理服务，由推理服务调用生成式代码大模型继续进行代码生成。

图 2 示出了一种流式推理的流程图，在推理阶段，推理平台 100 可使用可实时干预的流式推理，实现语言模型与语言服务的交互。基本流程如图 2 所示，推理服务基于代码上下文，调用生成式代码大模型流式生成得到当前结果，判断当前结果是否包括<ls_request>，若否，则继续生成；若是，调用语言服务，将语言服务的 API 响应拼接至代码上下文继续生成。循环上述流式推理，直到终止或截断，得到最终的代码片段（生成代码）。

在一些可能的实现方式中，推理平台 100 还支持人类开发者对代码生成进行干预。具体地，推理平台 100 允许人类开发者（如用户）在推理过程中协助语言模型推理。例如，推理平台 100（或推理服务）

可以监控加入 API 调用响应（语言服务返回结果）后语言模型的困惑度（Perplexity, PPL）的变化，可在 PPL 明显升高的时候生成（或插入）请求用户协助的 token，让用户选择从语言服务的返回结果中选择，模拟类似人类开发者编程过程中从 IDE 补全列表中选择的过程，从而达到人机协作结对编程的效果。其中，用户协助是指用户参与代码生成过程，为语言模型进行代码生成提供帮助。例如，用户可以从语言服务输出的至少一个推荐结果中选择一个推荐结果，提供给语言模型进行代码生成。其中，语言服务可以为代码补全服务，推荐结果可以是代码补全服务推荐的至少一个代码补全结果。

本申请实现代码生成的关键在于语言模型，下面对训练平台 200 训练得到语言模型的过程进行介绍。

训练平台 200 用于获取训练集，训练集中包括携带 API 调用的训练数据，携带 API 调用的训练数据可以包括项目级的代码上下文样本、API 调用响应样本和预测代码样本，例如是从海量的代码仓库中抽取的项目级的代码上下文样本、API 调用响应样本、预测代码样本，然后根据训练集对基座模型进行微调获得语言模型。其中，微调（Fine-tuning）是指在预训练模型（如基座模型）基础上，使用特定的数据集进行进一步的训练（或称为再训练），以使训练后的模型适应特定任务或领域。进一步地，训练集中还包括未携带 API 调用的训练数据，训练平台 200 用于根据携带 API 调用的训练数据和未携带 API 调用的训练数据，对基座模型进行混合微调获得语言模型。混合微调，也称作多任务混合微调，是指在针对多任务的混合数据集（例如是混合有携带 API 调用的训练数据和未携带 API 调用的训练数据的训练集）上对预训练模型进行再训练，以便于提高训练后的模型在所有任务上的性能。

仍以图 1 示例说明，训练集可以为图 1 中的代码数据集，携带 API 调用的训练数据包括代码上下文样本、API 调用响应样本以及预测代码样本，未携带 API 调用的训练数据包括代码上下文样本和预测代码样本。在训练阶段，训练平台 200 可以对海量软件代码仓中的代码进行处理，以构造训练数据。

为了构造高质量、多样化的训练集让语言模型学习调用 API 的时机和方法，训练平台可以通过不同方式或渠道收集训练数据。

一种方式可以为，训练平台 200 用于根据基于人类开发者调用语言服务的场景构造的启发式规则，生成训练数据。其中，启发式规则，也称作启发式算法（heuristics algorithm）、启发式规则算法，是一种根据经验规则进行发现的方法。启发式规则的特点是在解决问题时，利用过去的经验，选择已经行之有效的办法，而不是系统地、以确定的步骤去寻求答案。启发式规则可以实现在有限的搜索空间内，大大减少尝试的数量，迅速地解决问题。在本实施例中，训练平台 200 可以基于人类开发者大概率会调用语言服务的场景，如成员引用、对象构造、函数调用等场景，构造启发式规则算法，从而实现以较低成本构建语言模型的基本能力。

另一种方式可以为，训练平台 200 可以根据集成开发环境 IDE 的交互事件记录，构造训练数据。具体地，训练平台 200 可以从交互事件记录中抽取 API 调用行为，根据 API 调用行为提取 API 调用响应样本，并从项目的代码仓库中提取上下文样本、预测代码样本（期望生成代码），然后将上下文样本、API 调用响应样本、预测代码样本组装为训练数据。通过收集人类开发者的真实行为，构造训练数据，可以提高训练数据的可靠性和多样性，以提高语言模型的泛化能力。

还有一种方式可以为，训练平台 200 根据用户的标注信息，构造训练数据。其中，标注信息用于标注代码发生幻觉。通过人工标注的方式，可以强化幻觉高频发生场景下语言模型的应对能力。

其中，训练平台 200 在通过上述人工标注、构造启发式规则以及提取 IDE 的交互事件记录，抽取语言服务的调用行为，可以进行预处理，例如是进行分类、抽取、筛选、去重，获得人类开发者编码现场 IDE 使用行为数据。训练平台 200 可以对人类开发者编码现场 IDE 使用行为数据进行分词、编码、切分、组合，以构造训练数据。其中，切分、组合是指将数据按不同任务分类（如有 API 调用和无 API 调用）然后拼接成指定形式的训练数据。

训练平台 200 可以建立扩展词表以构建训练数据。为了模拟人类开发者写代码时触发语言服务调用请求获取辅助信息的行为，本方案引入用于标注 API 调用请求的特殊 token 如<ls_request>以及<ls_response>，当模型生成此 token 时，开始自主调用 API 并接收结果。进一步地，本方案还可以引入请求用户协助进行选择 token 如<user_request>以及<user_response>，用于标注请求人类协助的时机并接收用户的响应。扩展词表可以包括上述 token，进一步地，扩展词表还可以包括<req>、<res>，训练平台 200 可以从扩展词表中选择相应的 token，以构造训练数据。

其中，训练平台 200 根据 API 调用 token 生成 API 调用请求，将 API 调用请求与语言服务对 API 调用请求的响应结果（也即 API 调用响应）加入到训练数据。同时，API 调用支持语言模型链式调用 API，即收到响应结果后继续发出 API 调用请求，以模拟面对复杂场景时人类的多次调用行为。

训练平台 200 可以使用多任务混合微调方法，让语言模型保持普通代码生成能力的同时，赋予其调用语言服务、利用外部信息的能力。例如，训练平台可以采用三类任务混合微调。三类任务可以包括：

- a) 不涉及 API 调用的普通代码生成（给定代码上下文，预测生成代码）；
- b) API 调用请求生成（给定代码上下文，预测 API 调用）；
- c) 基于 API 调用响应（返回结果）的代码生成（例如是给定代码上下文和 API 调用请求及 API 调用响应，预测生成代码）。

图 3 为多任务混合微调的示例图，多任务包括任务一、任务二和任务三，其中，任务一为无 API 调用的代码生成，任务二为 API 调用生成，任务三为有 API 调用的代码生成。通过任务一进行微调，可以保持代码生成能力，通过任务二进行微调可以赋予 API 调用能力，通过任务三进行微调可以赋予外部信息（如 API 调用响应）的利用能力。

如此，可以实现训练与推理过程对齐，构建以代码大模型为调度中心的智能化框架。训练阶段通过构建高质量、多样化的训练数据，赋予语言模型调用语言服务的能力，让语言模型作为头脑和核心指挥调度，语言服务作为工具辅助，人类开发者作为结对编程者提供协助。相比于仅在推理阶段用外部干预影响模型行为的方式，例如是只在推理阶段通过外部介入的方式用语言服务/静态分析方法试图影响大模型的单步采样概率，本方案在模型训练阶段即在训练数据中加入工具调用行为，通过训练可以使得语言模型更符合预期行为即模拟人类得到 IDE 的提示，赋予语言模型自主获取外部信息、减少幻觉的能力。

在本申请中，语言模型被赋予调用语言服务自主、动态、即时地获取外部上下文信息的自主能力，语言模型可在推理过程中自主生成语言服务的 API 调用 token，动态地根据当前的代码上下文选择语言服务的 API 调用的类型、参数或时机，然后发出语言服务调用请求。语言服务的 API 调用请求可以被外部程序执行，外部程序返回的执行结果可以追加到当前上下文（如当前的代码上下文），继续让语言模型进行推理。

相比于将信息获取隔离于语言模型之外的方式，本方案让语言模型自主选择语言服务等外部工具的调用时机，自行获取与当前生成相关的外部信息，更贴近于实际编程中人类开发者所需的背景知识范围，对面向对象代码的生成尤其重要。而且，相比于外部信息仅可以对语言模型的行为提供宏观参考或干预的方式，本方案可以将静态分析的结果嵌入语言模型的代码上下文，让模型自主选择如何利用外部信息。

相比于全自动化的代码生成，本方案还允许可选的人类干预，即让人类在模型难以判断如何利用外部信息的时候，基于自身的知识和能力协助模型做出选择。

虽然近两年的代码生成技术已经取得了很大进步，但在可预见的未来仍旧无法代替人类开发者，后者所具有的全局观、领域知识、联想与推理能力等是当前人工智能模型很难拥有的，相关技术发挥较好的场景依然是比较通用的简单编程任务。因此，目前相关产品对于自己的定位，也越来越趋向于作为一个提供辅助功能的助手，而非全自动编程的机器人。在此背景下，本申请的意义在于结合软件分析工具和人工智能模型各自的优势，最大化发挥代码生成工具为开发者带来的赋能和增益，帮助开发者完成复杂任务的高效率和高质量开发。

基于前述的代码开发平台 10，本申请还提供一种代码生成方法。下面结合附图，对本申请的代码生成方法进行详细介绍。

参见图 4 所示的一种代码生成方法的流程图，该方法包括如下步骤：

S402、代码开发平台 10 接收代码生成任务。

代码生成任务用于根据用户的输入信息进行代码生成。代码生成任务可以是软件项目（Software Project）中触发的、用于生成代码的任务。软件项目也可以简称为项目，是开发者在进行软件开发时，针对需要开发的软件创建的工程文件。项目中可以包括多个代码文件，例如是用于实现软件的不同功能或特性的代码文件。项目中的代码文件可以是一个用户独立开发，也可以是多个用户协同开发。在进行开发时，用户可以使用代码生成能力进行代码自动生成。

用户的输入信息可以包括代码生成任务的任务描述信息或输入代码中的至少一项。其中，任务描述信息可以是自然语言形式的任务描述。例如，代码生成任务用于生成目标方法或目标函数的代码时，任务描述信息可以是创建上述目标方法或目标函数的需求描述，任务描述信息通常可以以注释形式输入。输入代码可以包括函数声明或方法声明。以函数声明示例说明，函数声明中可以包括函数名、参数名。相应地，代码生成任务可以是生成函数体的代码片段。

具体地，代码开发平台 10 可以向用户呈现代码编辑界面，其中，代码编辑界面可以是代码编辑器的用户界面，代码编辑界面可以是图形用户界面（graphical user interface, GUI），或者是命令用户界面（command user interface, CUI）。

为了便于描述，本申请以代码编辑界面为 GUI 示例说明。如图 5 所示，代码编辑界面 500 可以包括针对代码文件的编辑窗口 502，用户可以在编辑窗口 502 输入任务描述信息 504，任务描述信息 504 用于描述需要生成的代码，该示例中，任务描述信息 504 可以为“Creat a http sever instance and start it”，以及输入方法声明 506，例如为“public void init () {}”。用户可以触发代码编辑界面 500 的代码生成控件 508，从而触发代码生成操作。相应地，代码开发平台 10 可以接收用户输入的任务描述信息 504、方法声明 506（输入代码），进而基于该任务描述信息 504、方法声明 506 进行代码生成。

需要说明，图 5 仅仅是以用户的输入信息包括任务描述信息和输入代码示例说明，在本申请实施其他可能的实现方式中，用户的输入信息也可以包括任务描述信息，或者包括输入代码。例如，用户在编辑一些方法的代码时，可以直接编写方法，而无需添加注释形式的任务描述信息。又例如，用户在编辑一些方法的代码时，可以添加注释形式的任务描述信息，而无需输入代码。其中，在输入注释形式的任务描述信息 504 时，用户可以先输入注释的关键字，例如为“/*”字符、@字符或#字符，然后再输入任务描述信息，如此可以避免执行代码文件时执行上述任务描述信息 504。

此外，图 5 是以用户触发代码生成控件 508 以触发代码生成操作示例说明，在实际应用时，代码开发平台 10 也支持通过其他方式触发代码生成操作。例如，代码开发平台 10 也支持快捷键或菜单（如右键菜单）触发代码生成操作。

S404、代码开发平台 10 根据代码生成任务提取代码上下文。

代码上下文包括项目中与待生成代码相关的代码片段。该代码上下文为项目级上下文，不仅包括当前代码文件中的上下文（也称作文件内上下文），还包括跨文件的上下文。跨文件的上下文可以为项目中其他代码文件的上下文。项目中其他代码文件可以是当前代码文件引用或导入的其他代码文件。

S406、代码开发平台 10 根据代码上下文，通过语言模型确定是否调用语言服务的 API。若是，则执行 S408。

具体地，代码开发平台 10 可以根据代码上下文，通过语言模型进行代码生成得到当前结果。若当前结果中包括 API 调用标识符，例如包括<ls_request>或<request>、<req>，则代码开发平台 10 确定调用语言服务的 API。

S408、代码开发平台 10 向语言服务发送 API 调用请求，获得语言服务的 API 调用响应。

其中，代码开发平台 10 可以根据代码上下文，确定语言服务的 API 调用的类型、参数或时机中的至少一项。相应地，代码开发平台 10 可以根据语言服务的 API 调用的类型或参数，生成 API 调用请求。例如，代码开发平台 10 可以根据相应类型的 API 调用模板，组装 API 调用的参数，以生成 API 调用请求。代码开发平台 10 可以根据基于代码上下文确定的时机，向语言服务发送 API 调用请求。

S410、代码开发平台 10 根据代码上下文和语言服务的 API 调用响应，通过语言模型推理以获得生成代码。

代码开发平台 10 可以将代码上下文和语言服务的 API 调用响应进行拼接，将拼接结果（新的代码上下文）输入语言模型，通过语言模型推理以获得生成代码。其中，语言模型可以为代码大模型，代码开发平台 10 可以将拼接结果，以 prompt 形式输入语言模型，语言模型可以根据 prompt 进行推理，代码开发平台 10 可以获取语言模型推理的生成代码。进一步地，为了使得语言模型获得更丰富的信息，代码开发平台 10 还可以将语言服务的 API 调用请求也加入拼接。例如，代码开发平台 10 可以将代码上下文和语言服务的 API 调用请求、语言服务的 API 调用响应进行拼接，将拼接结果以 prompt 形式输入语言模型进行推理，以获得生成代码。其中，代码开发平台 10 在将代码上下文、语言服务的 API 调用请

求、语言服务的 API 调用响应进行拼接生成 Prompt 时，还可以在 prompt 模板的指定位置添加代码上下文、API 调用请求、API 调用响应，从而向语言模型显示提供不同类型的信息。

在一些可能的实现方式中，代码开发平台 10 还可以通过语言模型确定是否请求用户协助，若是，代码开发平台 10 还可以接收用户的响应，用户的响应包括用户对 API 调用响应的选择结果。代码开发平台 10 可以将 API 调用请求、用户对 API 调用响应的选择结果和代码上下文拼接，将拼接结果输入语言模型，通过语言模型推理获得生成代码。

其中，代码开发平台 10 获取语言模型在未加入 API 调用响应的困惑度和加入 API 调用响应的困惑度。困惑度用于度量一个概率分布或概率模型预测样本的好坏程度。低困惑度的概率模型能更好地预测样本。具体到本申请，概率模型可以是语言模型，因此，困惑度可以用于度量语言模型预测代码的好坏程度。一个语言模型可以看成是在整个句子或者文段上的概率分布。语言模型可以根据每个词 (token) 来估计一行代码或一段代码出现的概率，而且可以使用一行代码或一段代码的长度 (一行代码或一段代码包括的 token 数量) 进行标准化 (normalize)。在一些示例中，困惑度也可以通过平均分支系数 (average branch factor) 表征，平均分支系数表示预测下一个 token 时可以有多种选择。通常情况下，困惑度越小，表示语言模型越准确，代码生成效果越好。当困惑度的差值大于阈值时，例如是语言模型在未加入 API 调用响应的困惑度与语言模型在加入 API 调用响应的困惑度的差值大于阈值时，代码开发平台 10 确定请求用户协助。

其中，代码开发平台 10 可以检测语言模型的生成结果中是否包括请求用户协助的标识符。具体地，代码开发平台 10 可以在语言模型未加入 API 调用响应的困惑度和加入 API 调用响应的困惑度的差值大于阈值时，生成请求用户协助的标识符，例如为 <user_request>，代码开发平台 10 可以检测生成结果中是否包括 <user_request>。若是，代码开发平台 10 可以确定请求用户协助。

基于上述内容描述，本申请的代码生成方法支持语言模型在推理过程中自主生成 API 调用 token，发出语言服务的 API 调用请求，可以动态地根据代码上下文选择 API 调用的类型、参数、时机，而不局限于选取的一到两种固定的语言服务，且可以由语言模型自主决定调用时机，更贴合真实的代码开发场景，而且该方法以代码上下文、语言服务的 API 调用请求和语言服务的 API 调用响应的结构构建输入给语言模型的上下文，模拟人类开发者开发时的流程，不仅能让模型自主权衡，综合利用项目级的代码上下文和语言服务的信息，同时也能聚焦当前的关键信息，提高语言模型的预测质量，缓解幻觉问题。

接下来，将从人机交互方面阐述本申请在推理阶段可能的实现方案。

与其他同类工具一样，本申请的主要实现形式是作为代码编辑器或 IDE 的扩展功能或插件，因此前端界面体现为内嵌于 IDE 中的代码生成、补全等辅助编程工具 (也称作辅助编码工具)。目前，很多代码生成的插件工具都提供了以上类似的体验，与此类工具不同的是，本方案提供如下选项，标记语言服务等外部工具的调用位置，点击即可显示该语言服务的 API 调用的具体信息，包括调用的 API 名、参数和返回内容，如图 6 所示。

参见图 6，用户在代码编辑器或 IDE 的代码编辑界面触发代码生成任务，语言模型可以在图 6 中的 MatcherUtils.处生成 <ls_request>，语言模型检测到 <ls_request>，可以调用语言服务的 API，然后语言服务返回 API 调用响应，具体为 LSP::getCompletion()，语言模型根据拼接的代码上下文和 API 的调用请求、API 调用响应继续生成代码，重复上述过程，直到生成终止符或生成截断，本次代码生成任务完成。

以上对语言模型的推理过程进行了介绍，下面对语言模型的训练过程介绍。

参见图 7 所示的一种训练语言模型的流程示意图，具体包括如下步骤：

S702、代码开发平台 10 获取训练集。

训练集中包括携带 API 调用的训练数据。为了赋予语言模型自主调用语言服务的 API、与语言服务等外部工具实时交互的能力，本申请引入用于标注 API 调用请求和 API 调用响应的特殊 token，例如以 <ls_request>和</ls_request>作为 API 调用请求 token，<ls_response>和</ls_response>作为 API 调用响应 token，然后将 API 调用请求和 API 调用响应加入到代码上下文中，获得携带 API 调用的训练数据。其中，语言服务可以为辅助编程服务，如 IDE 的代码生成服务、代码补全服务。

为了便于理解，本申请还提供了携带 API 调用的训练数据的示例。图 8 示出了一条训练数据的代码格式，该训练数据中包括代码上下文、API 调用请求和 API 调用响应，其中，代码上下文包括图 8 中 <ls_request>之前的代码片段，API 调用请求包括<ls_request>之后的代码片段，如 LSP::getCompletion()，API 调用响应包括<ls_response>之后的代码片段。其中，API 调用响应可以插入在 API 调用请求中。

进一步地，API 调用形式设计允许语言模型链式调用 API，以模拟面对复杂场景时人类的行为。链式调用 API 的代码格式可以参见图 9。例如，针对一个 API 调用请求，插入 API 调用响应后，可以继续发起 API 调用请求，插入与该 API 调用请求对应的 API 调用响应。在图 9 的示例中，训练数据中插入有 LSP::getCompletion() 的 API 调用响应，例如是 parseURIPatternLable、parseURIPatternTemplate、parseURIRefAttrInConfig，还继续插入有如下 API 调用请求：LSP::getDef (“parseURIRefAttrInConfig”) 以及与该 API 调用请求对应的 API 调用响应。

为了构造高质量、多样化的训练集，让语言模型学习调用 API 的时机和方法，代码开发平台 10 可以通过不同方式或渠道收集训练数据。具体地，代码开发平台 10 可以根据基于人类开发者调用语言服务的场景构造的启发式规则，生成训练数据。基于人类开发者大概率会调用语言服务的场景，如成员引用、对象构造、函数调用等场景，构造启发式规则算法，可以实现以较低成本构建语言模型的基本能力。或者，代码开发平台 10 可以根据集成开发环境 IDE 的交互事件记录，构造训练数据。代码开发平台 10 通过收集人类开发者的真实行为，可以提高训练数据的可靠性和多样性，以提高语言模型的泛化能力。代码开发平台 10 也可以根据用户的标注信息，构造训练数据。标注信息用于标注代码发生幻觉。通过人工标注的方式，可以强化幻觉高频发生场景下语言模型的应对能力。

S704、代码开发平台 10 根据训练集对基座模型进行微调，获得语言模型。

代码开发平台 10 可以根据训练集，通过工具学习对基座模型进行微调，以获得语言模型。其中，基座模型可以是预训练模型，例如是预训练的 LLM。工具学习主要可以分为两类：工具增强学习（Tool-augmented Learning）和工具导向学习（Tool-oriented Learning）。其核心差异在于学习过程中，是通过工具执行来增强基座模型（工具为 AI 服务），还是通过基座模型优化工具的使用（AI 为工具服务）。

以工具增强学习示例说明。工具增强学习利用各种外部工具的执行结果，增强基座模型的性能。在这一范式中，语言服务等外部工具的执行结果被视为辅助生成高质量输出的外部信息。将语言服务等外部工具的返回结果或输出作为额外信息辅助语言模型进行代码生成，解决语言模型尤其是 LLM 生成过程中的事实性不准确的问题。

在一些可能的实现方式中，训练集中还包括未携带 API 调用的训练数据。未携带 API 调用的训练数据包括代码上下文样本和预测代码样本。相应地，代码开发平台 10 可以根据携带 API 调用的训练数据和未携带 API 调用的训练数据，对基座模型进行混合微调，获得语言模型。

需要说明的是，图 7 实施例可以独立实施，也可以与前述的实施例结合，本申请对此不作限制。

该方法在训练阶段通过构建高质量、多样化的训练数据，赋予语言模型调用语言服务的能力，让语言模型作为核心进行指挥调度，语言服务作为工具辅助，如此以实现训练与推理过程对齐，构建以语言模型为调度中心的智能化框架。

基于前述的代码生成方法，本申请还提供一种代码开发平台 10。如图 10 所示，该代码开发平台 10 包括：

交互模块 1002，用于接收代码生成任务，代码生成任务用于根据用户的输入信息进行代码生成；

代码上下文处理模块 1004，用于根据代码生成任务提取代码上下文，代码上下文包括项目中与待生成代码相关的代码片段；

推理模块 1006，用于根据代码上下文，通过语言模型确定是否调用语言服务的应用程序编程接口 API；

代码上下文处理模块 1004，还用于当确定调用语言服务的 API，向语言服务发送 API 调用请求，获得语言服务的 API 调用响应，语言服务用于辅助生成代码，API 调用响应包括语言服务的至少一个推荐结果；

推理模块 1006，还用于根据代码上下文和语言服务的 API 调用响应，通过语言模型推理以获得生

成代码。

其中，交互模块 1002、代码上下文处理模块 1004、推理模块 1006 可以是图 1 的实施例中推理平台 100 中的模块。交互模块 1002 在图 1 中未示出，代码上下文处理模块 1004 可以是图 1 中进行代码上下文处理的模块，推理模块 1006 可以是图 1 中进行推理的模块，可以通过调用语言模型实现。

5 示例性地，上述交互模块 1002、代码上下文处理模块 1004、推理模块 1006 可以通过硬件实现，或者可以通过软件实现。

其中，当通过软件实现时，上述交互模块 1002、代码上下文处理模块 1004、推理模块 1006 可以是运行在计算设备上的应用程序，如计算引擎等。该应用程序可以通过虚拟化服务提供给用户使用。虚拟化服务可以包括虚拟机（virtual machine，VM）服务、裸金属服务器（bare metal server，BMS）服务以及容器（container）服务。其中，VM 服务可以通过虚拟化技术在多个物理主机上虚拟出虚拟机（virtual machine，VM）资源池以为用户按需提供 VM 进行使用的服务。BMS 服务是在多个物理主机上虚拟出 BMS 资源池以为用户按需提供 BMS 进行使用的服务。容器服务是在多个物理主机上虚拟出容器资源池以为用户按需提供容器进行使用的服务。VM 是模拟出来的一台虚拟的计算机，也即逻辑上的一台计算机。BMS 是一种可弹性伸缩的高性能计算服务，计算性能与传统物理机无差别，具有安全物理隔离的特点。容器是一种内核虚拟化技术，可以提供轻量级的虚拟化，以达到隔离用户空间、进程和资源的目的。应理解，上述虚拟化服务中的 VM 服务、BMS 服务以及容器服务仅仅是作为具体的事例，在实际应用中，虚拟化服务还可以其他轻量级或者重量级的虚拟化服务，此处不作具体限定。

当通过硬件实现时，交互模块 1002、代码上下文处理模块 1004、推理模块 1006 中可以包括至少一个计算设备，如服务器等。或者，交互模块 1002、代码上下文处理模块 1004、推理模块 1006 也可以是利用专用集成电路（application-specific integrated circuit，ASIC）实现、或可编程逻辑器件（programmable logic device，PLD）实现的设备等。其中，上述 PLD 可以是复杂程序逻辑器件（complex programmable logical device，CPLD）、现场可编程门阵列（field-programmable gate array，FPGA）、通用阵列逻辑（generic array logic，GAL）或其任意组合实现。

25 在一些可能的实现方式中，推理模块 1006 具体用于：
根据代码上下文，通过语言模型进行代码生成得到当前结果；
若当前结果中包括 API 调用标识符，确定调用语言服务的 API。
在一些可能的实现方式中，推理模块 1006 还用于：
通过语言模型预测是否请求用户协助；
交互模块 1002，还用于若是，接收用户的响应，用户的响应包括用户对 API 调用响应的选择结果。
30 在一些可能的实现方式中，推理模块 1006 具体用于：
获取语言模型在未加入 API 调用响应的困惑度和加入 API 调用响应的困惑度；
当困惑度的差值大于阈值时，确定请求用户协助。
在一些可能的实现方式中，推理模块 1006 具体用于：
检测语言模型的生成结果中是否包括请求用户协助的标识符；
35 若是，确定请求用户协助。
在一些可能的实现方式中，代码上下文处理模块 1004 还用于：
根据代码上下文，确定语言服务的 API 调用的类型、参数或时机中的至少一项；
代码上下文处理模块 1004 具体用于：
根据语言服务的 API 调用的类型或参数，生成 API 调用请求；
40 根据时机，向语言服务发送 API 调用请求。
在一些可能的实现方式中，代码开发平台 10 还包括：
训练模块 1008，用于获取训练集，训练集中包括携带 API 调用的训练数据，携带 API 调用的训练数据包括项目级的代码上下文样本、API 调用响应样本和预测代码样本；
训练模块 1008，还用于根据训练集对基座模型进行微调，获得语言模型。
45 训练模块 1008 可以通过硬件实现，或者可以通过软件实现。
其中，当通过软件实现时，上述训练模块 1008 可以是运行在计算设备上的应用程序，如计算引擎

等。该应用程序可以通过虚拟化服务，如 VM 服务、BMS 服务或 container 服务提供给用户使用。当通过硬件实现时，训练模块 1008 中可以包括至少一个计算设备，如服务器等。或者，训练模块 1008 也可以是利用 ASIC 实现、或 PLD 实现的设备等。

在一些可能的实现方式中，训练集中还包括未携带 API 调用的训练数据，未携带 API 调用的训练数据包括代码上下文样本和预测代码样本；

训练模块 1008 具体用于：

根据携带 API 调用的训练数据和未携带 API 调用的训练数据，对基座模型进行混合微调，获得语言模型。

在一些可能的实现方式中，训练模块 1008 具体用于：

根据基于人类开发者调用语言服务的场景构造的启发式规则，生成训练数据；或者，

根据集成开发环境 IDE 的交互事件记录，构造训练数据；或者，

根据用户的标注信息，构造训练数据。

在一些可能的实现方式中，语言服务包括辅助编程服务。

本申请还提供一种计算设备 1100。如图 11 所示，计算设备 1100 包括：总线 1102、处理器 1104、存储器 1106 和通信接口 1108。处理器 1104、存储器 1106 和通信接口 1108 之间通过总线 1102 通信。计算设备 1100 可以是服务器或终端设备。应理解，本申请不限定计算设备 1100 中的处理器、存储器的个数。

总线 1102 可以是外设部件互连标准（peripheral component interconnect, PCI）总线或扩展工业标准结构（extended industry standard architecture, EISA）总线等。总线可以分为地址总线、数据总线、控制总线等。为便于表示，图 11 中仅用一条线表示，但并不表示仅有一根总线或一种类型的总线。总线 1102 可包括在计算设备 1100 各个部件（例如，存储器 1106、处理器 1104、通信接口 1108）之间传递信息的通路。

处理器 1104 可以包括中央处理器（central processing unit, CPU）、图形处理器（graphics processing unit, GPU）、微处理器（micro processor, MP）或者数字信号处理器（digital signal processor, DSP）等处理器中的任意一种或多种。

存储器 1106 可以包括易失性存储器（volatile memory），例如随机存取存储器（random access memory, RAM）。存储器 1106 还可以包括非易失性存储器（non-volatile memory），例如只读存储器（read-only memory, ROM）、快闪存储器、机械硬盘（hard disk drive, HDD）或固态硬盘（solid state drive, SSD）。存储器 1106 中存储有可执行的程序代码，处理器 1104 执行该可执行的程序代码以实现前述代码生成方法。具体的，存储器 1106 上存有代码开发平台 10 用于执行代码生成方法的指令。

通信接口 1108 使用例如但不限于网络接口卡、收发器一类的收发模块，来实现计算设备 1100 与其他设备或通信网络之间的通信。

本申请实施例还提供了一种计算设备集群。该计算设备集群包括至少一台计算设备。该计算设备可以是服务器，例如是中心服务器、边缘服务器，或者是本地数据中心中的本地服务器。在一些实施例中，计算设备也可以是台式机、笔记本电脑或者智能手机等终端设备。

如图 12 所示，所述计算设备集群包括至少一个计算设备 1100。计算设备集群中的一个或多个计算设备 1100 中的存储器 1106 中可以存有相同的代码开发平台 10 用于执行代码生成方法的指令。

在一些可能的实现方式中，该计算设备集群中的一个或多个计算设备 1100 也可以用于执行代码开发平台 10 用于执行代码生方法的部分指令。换言之，一个或多个计算设备 1100 的组合可以共同执行代码开发平台 10 用于执行代码生成方法的指令。

需要说明的是，计算设备集群中的不同的计算设备 1100 中的存储器 1106 可以存储不同的指令，用于执行代码开发平台 10 的部分功能。

图 13 示出了一种可能的实现方式。如图 13 所示，两个计算设备 1100A 和 1100B 通过通信接口 1108 实现连接。计算设备 1100A 中的存储器上存有利于执行交互模块 1002、代码上下文处理模块 1004

的功能的指令。计算设备1100B中的存储器上存有助于执行推理模块1006的功能的指令。进一步地，计算设备1100B的存储器上还可以存储有助于执行训练模块1008的功能的指令。换言之，计算设备1100A和1100B的存储器1106共同存储了代码开放平台10有助于执行代码生成方法的指令。

图13所示的计算设备集群之间的连接方式可以是考虑到本申请提供的代码生成方法需要大量的算力资源进行模型推理或模型训练。因此，考虑将推理模块1006、训练模块1008实现的功能交由计算设备1100B执行。

应理解，图13中示出的计算设备1100A的功能也可以由多个计算设备1100完成。同样，计算设备1100B的功能也可以由多个计算设备1100完成。

在一些可能的实现方式中，计算设备集群中的一个或多个计算设备可以通过网络连接。其中，所述网络可以是广域网或局域网等等。图14示出了一种可能的实现方式。如图14所示，两个计算设备1100C和1100D之间通过网络进行连接。具体地，通过各个计算设备中的通信接口与所述网络进行连接。在这一类可能的实现方式中，计算设备1100C中的存储器1106中存有执行交互模块1002、代码上下文处理模块1004的功能的指令。同时，计算设备1100D中的存储器1106中存有执行推理模块1006、训练模块1008的功能的指令。

图14所示的计算设备集群之间的连接方式可以是考虑到本申请提供的代码生成方法需要较多的算力资源进行模型推理或模型训练，因此考虑将推理模块1006、训练模块1008实现的功能交由计算设备1100D执行。

应理解，图14中示出的计算设备1100C的功能也可以由多个计算设备1100完成。同样，计算设备1100D的功能也可以由多个计算设备1100完成。

本申请实施例还提供了一种计算机可读存储介质。所述计算机可读存储介质可以是计算设备能够存储的任何可用介质或者是包含一个或多个可用介质的数据中心等数据存储设备。所述可用介质可以是磁性介质，（例如，软盘、硬盘、磁带）、光介质（例如，DVD）、或者半导体介质（例如固态硬盘）等。该计算机可读存储介质包括指令，所述指令指示计算设备执行上述应用于代码开发平台10有助于执行代码生成方法。

本申请实施例还提供了一种包含指令的计算机程序产品。所述计算机程序产品可以是包含指令的，能够运行在计算设备上或被储存在任何可用介质中的软件或程序产品。当所述计算机程序产品在至少一个计算机设备上运行时，使得至少一个计算设备执行上述代码生成方法。

最后应说明的是：以上实施例仅用以说明本发明的技术方案，而非对其限制；尽管参照前述实施例对本发明进行了详细的说明，本领域的普通技术人员应当理解：其依然可以对前述各实施例所记载的技术方案进行修改，或者对其中部分技术特征进行等同替换；而这些修改或者替换，并不使相应技术方案

权 利 要 求 书

1. 一种代码生成方法，其特征在于，所述方法包括：

代码开发平台接收代码生成任务，所述代码生成任务用于根据用户的输入信息进行代码生成；

5 所述代码开发平台根据所述代码生成任务提取代码上下文，所述代码上下文包括项目中与待生成代码相关的代码片段；

所述代码开发平台根据所述代码上下文，通过语言模型确定是否调用语言服务的应用程序编程接口 API；

10 当确定调用所述语言服务的 API，所述代码开发平台向所述语言服务发送 API 调用请求，获得所述语言服务的 API 调用响应，所述语言服务用于辅助生成代码，所述 API 调用响应包括所述语言服务的至少一个推荐结果；

所述代码开发平台根据所述代码上下文和所述语言服务的 API 调用响应，通过所述语言模型推理以获得生成代码。

2. 根据权利要求 1 所述的方法，其特征在于，所述代码开发平台根据所述代码上下文，通过语言模型确定是否调用语言服务的应用程序编程接口 API，包括：

15 所述代码开发平台根据所述代码上下文，通过语言模型进行代码生成得到当前结果；

若当前结果中包括 API 调用标识符，确定调用语言服务的 API。

3. 根据权利要求 1 或 2 所述的方法，其特征在于，所述方法还包括：

所述代码开发平台通过所述语言模型预测是否请求用户协助；

20 若是，所述代码开发平台接收所述用户的响应，所述用户的响应包括所述用户对所述 API 调用响应的选择结果。

4. 根据权利要求 3 所述的方法，其特征在于，所述代码开发平台通过所述语言模型确定是否请求用户协助，包括：

所述代码开发平台获取所述语言模型在未加入所述 API 调用响应的困惑度和加入所述 API 调用响应的困惑度；

25 当所述困惑度的差值大于阈值时，所述代码开发平台确定请求用户协助。

5. 根据权利要求 3 所述的方法，其特征在于，所述代码开发平台通过所述语言模型确定是否请求用户协助，包括：

所述代码开发平台检测所述语言模型的生成结果中是否包括请求用户协助的标识符；

若是，所述代码开发平台确定请求所述用户协助。

30 6. 根据权利要求 1 至 5 任一项所述的方法，其特征在于，所述方法还包括：

所述代码开发平台根据所述代码上下文，确定所述语言服务的 API 调用的类型、参数或时机中的至少一项；

所述代码开发平台向所述语言服务发送 API 调用请求，包括：

所述代码开发平台根据所述语言服务的 API 调用的类型或参数，生成 API 调用请求；

35 所述代码开发平台根据所述时机，向所述语言服务发送所述 API 调用请求。

7. 根据权利要求 1 至 6 任一项所述的方法，其特征在于，所述方法还包括：

所述代码开发平台获取训练集，所述训练集中包括携带 API 调用的训练数据，所述携带 API 调用的训练数据包括项目级的代码上下文样本、API 调用响应样本和预测代码样本；

所述代码开发平台根据所述训练集对基座模型进行微调，获得所述语言模型。

40 8. 根据权利要求 7 所述的方法，其特征在于，所述训练集中还包括未携带 API 调用的训练数据，所述未携带 API 调用的训练数据包括代码上下文样本和预测代码样本；

所述代码开发平台根据所述训练集对所述基座模型进行微调，获得所述语言模型，包括：

所述代码开发平台根据所述携带 API 调用的训练数据和所述未携带 API 调用的训练数据，对所述基座模型进行混合微调，获得所述语言模型。

45 9. 根据权利要求 7 或 8 所述的方法，其特征在于，所述训练数据通过如下方式收集：

所述代码开发平台根据基于人类开发者调用语言服务的场景构造的启发式规则，生成所述训练数据；或者，

所述代码开发平台根据集成开发环境 IDE 的交互事件记录，构造所述训练数据；或者，
所述代码开发平台根据所述用户的标注信息，构造所述训练数据。

5 10. 根据权利要求 1 至 9 任一项所述的方法，其特征在于，所述语言服务包括辅助编程服务。

11. 一种代码开发平台，其特征在于，所述代码开发平台包括：

交互模块，用于接收代码生成任务，所述代码生成任务用于根据用户的输入信息进行代码生成；

代码上下文处理模块，用于根据所述代码生成任务提取代码上下文，所述代码上下文包括项目中与待生成代码相关的代码片段；

10 推理模块，用于根据所述代码上下文，通过语言模型确定是否调用语言服务的应用程序编程接口 API；

所述代码上下文处理模块，还用于当确定调用所述语言服务的 API，向所述语言服务发送 API 调用请求，获得所述语言服务的 API 调用响应，所述语言服务用于辅助生成代码，所述 API 调用响应包括所述语言服务的至少一个推荐结果；

15 所述推理模块，还用于根据所述代码上下文和所述语言服务的 API 调用响应，通过所述语言模型推理以获得生成代码。

12. 根据权利要求 11 所述的代码开发平台，其特征在于，所述推理模块具体用于：

根据所述代码上下文，通过语言模型进行代码生成得到当前结果；

若当前结果中包括 API 调用标识符，确定调用语言服务的 API。

20 13. 根据权利要求 11 或 12 所述的代码开发平台，其特征在于，所述推理模块还用于：

通过所述语言模型预测是否请求用户协助；

所述交互模块，还用于若是，接收所述用户的响应，所述用户的响应包括所述用户对所述 API 调用响应的选择结果。

14. 根据权利要求 13 所述的代码开发平台，其特征在于，所述推理模块具体用于：

25 获取所述语言模型在未加入所述 API 调用响应的困惑度和加入所述 API 调用响应的困惑度；

当所述困惑度的差值大于阈值时，确定请求用户协助。

15. 根据权利要求 13 所述的代码开发平台，其特征在于，所述推理模块具体用于：

检测所述语言模型的生成结果中是否包括请求用户协助的标识符；

若是，确定请求所述用户协助。

30 16. 根据权利要求 11 至 15 任一项所述的代码开发平台，其特征在于，所述代码上下文处理模块还用于：

根据所述代码上下文，确定所述语言服务的 API 调用的类型、参数或时机中的至少一项；

所述代码上下文处理模块具体用于：

根据所述语言服务的 API 调用的类型或参数，生成 API 调用请求；

35 根据所述时机，向所述语言服务发送所述 API 调用请求。

17. 根据权利要求 11 至 16 任一项所述的代码开发平台，其特征在于，所述代码开发平台还包括：

训练模块，用于获取训练集，所述训练集中包括携带 API 调用的训练数据，所述携带 API 调用的训练数据包括项目级的代码上下文样本、API 调用响应样本和预测代码样本；

所述训练模块，还用于根据所述训练集对基座模型进行微调，获得所述语言模型。

40 18. 根据权利要求 17 所述的代码开发平台，其特征在于，所述训练集中还包括未携带 API 调用的训练数据，所述未携带 API 调用的训练数据包括代码上下文样本和预测代码样本；

所述训练模块具体用于：

根据所述携带 API 调用的训练数据和所述未携带 API 调用的训练数据，对所述基座模型进行混合微调，获得所述语言模型。

45 19. 根据权利要求 17 或 18 所述的代码开发平台，其特征在于，所述训练模块具体用于：

根据基于人类开发者调用语言服务的场景构造的启发式规则，生成所述训练数据；或者，

根据集成开发环境 IDE 的交互事件记录，构造所述训练数据；或者，

根据所述用户的标注信息，构造所述训练数据。

20. 根据权利要求 11 至 19 任一项所述的代码开发平台，其特征在于，所述语言服务包括辅助编程服务。

5 21. 一种计算设备集群，其特征在于，所述计算设备集群包括至少一台计算设备，所述至少一台计算设备包括至少一个处理器和至少一个存储器，所述至少一个存储器中存储有计算机可读指令；所述至少一个处理器执行所述计算机可读指令，以使得所述计算设备集群执行如权利要求 1 至 10 中任一项所述的代码生成方法。

10 22. 一种计算机可读存储介质，其特征在于，包括计算机可读指令；所述计算机可读指令用于实现权利要求 1 至 10 任一项所述的代码生成方法。

23. 一种计算机程序产品，其特征在于，包括计算机可读指令；所述计算机可读指令用于实现权利要求 1 至 10 任一项所述的代码生成方法。