(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 118519660 A
(43)申请公布日 2024.08.20

(21)申请号 202410797752.9

(22)申请日 2024.06.19

(71)申请人 北京百度网讯科技有限公司
地址 100085 北京市海淀区上地十街10号百度大厦2层

(72)发明人 吴京京 贺思俊 陈泽裕 马艳军

(74)专利代理机构 中科专利商标代理有限责任公司 11021
专利代理师 鄢功军

(51)Int.Cl.
G06F 8/65(2018.01)
G06F 40/30(2020.01)
G06F 40/205(2020.01)
G06F 40/186(2020.01)

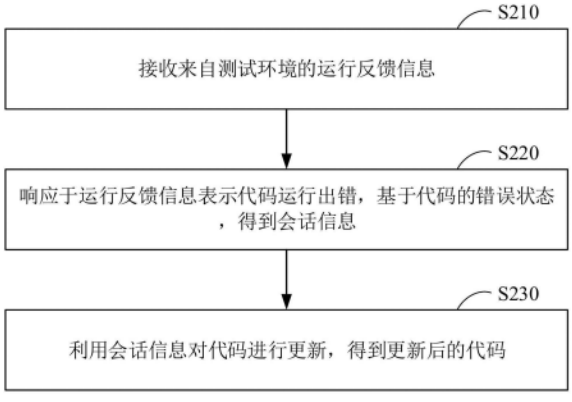
权利要求书3页 说明书14页 附图7页

(54)发明名称

基于大模型的代码更新方法、代码生成方法、任务处理方法及代码解释器

(57)摘要

本公开提供了基于大模型的代码更新方法、代码生成方法、任务处理方法及代码解释器，涉及人工智能技术领域，尤其涉及深度学习、自然语言处理、软件工程、大模型技术领域。具体实现方案为：接收来自测试环境的运行反馈信息，运行反馈信息用于表示代码在测试环境中的运行结果，代码是利用大语言模型，基于用户意图生成的；响应于运行反馈信息表示代码运行出错，基于代码的错误状态，得到会话信息；以及利用会话信息对代码进行更新，得到更新后的代码。



CN 118519660 A

1.一种基于大模型的代码更新方法,包括:

接收来自测试环境的运行反馈信息,所述运行反馈信息用于表示代码在所述测试环境中的运行结果,所述代码是利用大语言模型,基于用户意图生成的;

响应于所述运行反馈信息表示所述代码运行出错,基于所述代码的错误状态,得到会话信息;以及

利用所述会话信息对所述代码进行更新,得到更新后的代码。

2.根据权利要求1所述的方法,其中,所述代码的错误状态是基于所述代码的已更新次数来确定的;

所述方法还包括:

在所述代码的已更新次数小于或等于预设值的情况下,确定所述代码的错误状态表示为第一状态;以及

在所述代码的已更新次数大于所述预设值的情况下,确定所述代码的错误状态表示为第二状态。

3.根据权利要求2所述的方法,其中,所述基于所述代码的错误状态,得到会话信息,包括:

在所述代码的错误状态表示为所述第一状态的情况下,基于所述运行反馈信息中包括的错误码,确定当前错误信息;以及

基于所述当前错误信息和工具调用信息,得到所述会话信息。

4.根据权利要求2所述的方法,其中,所述基于所述代码的错误状态,得到会话信息,包括:

在所述代码的错误状态表示为所述第二状态的情况下,基于所述运行反馈信息中包括的错误码,确定当前错误信息,所述当前错误信息表示当前代码更新轮次的错误信息;以及

基于所述当前错误信息、所述代码的历史错误信息、所述代码的历史会话信息和所述代码的元信息,得到所述会话信息;

其中,所述历史错误信息包括多个历史代码更新轮次各自的错误信息,所述历史会话信息包括所述多个历史代码更新轮次各自的会话信息,所述元信息包括代码生成任务信息、模型信息和工具调用信息中的至少一项。

5.根据权利要求4所述的方法,其中,所述基于所述当前错误信息、所述代码的历史错误信息、所述代码的历史会话信息和所述代码的元信息,得到所述会话信息,包括:

基于信息选择策略,从所述代码的历史错误信息中选择得到目标错误信息;以及

将所述当前错误信息、所述目标错误信息、所述历史会话信息和所述元信息进行拼接,得到所述会话信息。

6.根据权利要求5所述的方法,其中,所述基于信息选择策略,从所述代码的历史错误信息中选择得到目标错误信息,包括:

从所述多个历史代码更新轮次各自的错误信息中,选择与所述当前代码更新轮次最接近的历史代码更新轮次的错误信息作为目标错误信息;或者

从所述多个历史代码更新轮次各自的错误信息中,选择与所述错误码对应的错误信息作为所述目标错误信息。

7.根据权利要求2所述的方法,还包括:

响应于利用所述错误反思信息对所述代码进行更新,基于自增策略调整所述代码的已更新次数。

8.根据权利要求1所述的方法,其中,所述利用所述会话信息对所述代码进行更新,得到更新后的代码,包括:

将所述会话信息和所述代码填入提示模板,得到提示文本;以及

将所述提示文本输入大语言模型,以便所述大语言模型以所述会话信息作为提示信息,基于所述代码进行新代码的生成,得到所述更新后的代码。

9.根据权利要求1所述的方法,还包括:

响应于所述运行反馈信息表示所述代码运行正常,确定所述代码为符合所述用户意图的目标代码。

10.一种代码生成方法,包括:

响应于代码生成请求,基于所述代码生成请求包括的意图信息,生成初始代码;以及

在测试环境中,基于当前运行的代码的错误状态,对所述当前运行的代码进行迭代更新,以得到目标代码,其中,在第一个代码更新轮次中,所述当前运行的代码为所述初始代码。

11.一种任务处理方法,包括:

响应于任务处理请求,解析所述任务处理请求包括的目标任务的任务文本,得到意图信息;

基于所述意图信息,生成初始代码;

在测试环境中,基于当前运行的代码的错误状态,对所述当前运行的代码进行迭代更新,以得到目标代码,其中,在第一个代码更新轮次中,所述当前运行的代码为所述初始代码;以及

在生产环境中运行所述目标代码,以执行所述目标任务。

12.一种代码解释器,包括:

代码沙盒,被配置为提供测试环境,所述代码沙盒被配置为接收代码,在所述测试环境中运行所述代码,并生成所述代码的运行反馈信息,其中,所述代码是由大语言模型生成的;

错误反思模块,被配置为接收所述运行反馈信息,响应于所述运行反馈信息表示所述代码运行出错,基于所述代码的错误状态,得到会话信息,并向所述大语言模型发送所述会话信息,其中,所述大语言模型被配置为基于所述会话信息和所述代码,生成更新后的代码。

13.根据权利要求1所述的代码解释器,还包括:

错误管理模块,被配置为记录多个错误码,和与所述错误码对应的错误信息;

其中,所述错误反思模块,被配置为在所述代码的错误状态表示为第一状态的情况下,通过所述错误管理模块,获取与所述运行反馈信息中包括的错误码对应的当前错误信息,并基于所述当前错误信息和工具调用信息,得到所述会话信息;

其中,所述第一状态表示所述代码的已更新次数小于或等于预设值。

14.根据权利要求1所述的代码解释器,还包括:

记忆管理模块,被配置为记录所述代码的元信息,并记录历史代码更新轮次中的错误

信息和会话信息,得到历史错误信息和历史会话信息;

其中,所述错误反思模块,被配置为在所述代码的错误状态表示为第二状态的情况下,通过所述错误管理模块,获取与所述运行反馈信息中包括的错误码对应的当前错误信息,通过所述记忆管理模块,获取所述元信息、所述历史错误信息和所述历史会话信息,并基于所述当前错误信息、所述元信息、所述历史错误信息和所述历史会话信息,得到所述会话信息;

其中,所述第二状态表示所述代码的已更新次数大于预设值。

15.一种基于大模型的代码更新装置,包括:

接收模块,用于接收来自测试环境的运行反馈信息,所述运行反馈信息用于表示代码在所述测试环境中的运行结果,所述代码是利用大语言模型,基于用户意图生成的;

第一处理模块,用于响应于所述运行反馈信息表示所述代码运行出错,基于所述代码的错误状态,得到会话信息;以及

第一更新模块,用于利用所述会话信息对所述代码进行更新,得到更新后的代码。

16.一种代码生成装置,包括:

第一生成模块,用于响应于代码生成请求,基于所述代码生成请求包括的意图信息,生成初始代码;以及

第二更新模块,用于在测试环境中,基于当前运行的代码的错误状态,对所述当前运行的代码进行迭代更新,以得到目标代码,其中,在第一个代码更新轮次中,所述当前运行的代码为所述初始代码。

17.一种任务处理装置,包括:

解析模块,用于响应于任务处理请求,解析所述任务处理请求包括的目标任务的任务文本,得到意图信息;

第二生成模块,用于基于所述意图信息,生成初始代码;

第三更新模块,用于在测试环境中,基于当前运行的代码的错误状态,对所述当前运行的代码进行迭代更新,以得到目标代码,其中,在第一个代码更新轮次中,所述当前运行的代码为所述初始代码;以及

第二处理模块,用于在生产环境中运行所述目标代码,以执行所述目标任务。

18.一种电子设备,包括:

至少一个处理器;以及

与所述至少一个处理器通信连接的存储器;其中,

所述存储器存储有可被所述至少一个处理器执行的指令,所述指令被所述至少一个处理器执行,以使所述至少一个处理器能够执行权利要求1-11中任一项所述的方法。

19.一种存储有计算机指令的非瞬时计算机可读存储介质,其中,所述计算机指令用于使所述计算机执行根据权利要求1-11中任一项所述的方法。

20.一种计算机程序产品,包括计算机程序,所述计算机程序在被处理器执行时实现根据权利要求1-11中任一项所述的方法。

基于大模型的代码更新方法、代码生成方法、任务处理方法及代码解释器

技术领域

[0001] 本公开涉及人工智能技术领域，尤其涉及深度学习、自然语言处理、软件工程、大模型技术领域，更具体地，涉及一种基于大模型的代码更新方法、代码生成方法、任务处理方法及代码解释器。

背景技术

[0002] 随着大模型的不断发展，其在专业领域的应用也越来越广泛，例如，在软件工程领域，代码大模型能够根据用户的上下文和需求生成高质量代码，从而完成在代码上的需求。

发明内容

[0003] 本公开提供了一种基于大模型的代码更新方法、代码生成方法、任务处理方法及代码解释器。

[0004] 根据本公开的一方面，提供了一种基于大模型的代码更新方法，包括：接收来自测试环境的运行反馈信息，上述运行反馈信息用于表示代码在上述测试环境中的运行结果，上述代码是利用大语言模型，基于用户意图生成的；响应于上述运行反馈信息表示上述代码运行出错，基于上述代码的错误状态，得到会话信息；以及利用上述会话信息对上述代码进行更新，得到更新后的代码。

[0005] 根据本公开的另一方面，提供了一种代码生成方法，包括：响应于代码生成请求，基于上述代码生成请求包括的意图信息，生成初始代码；以及在测试环境中，基于当前运行的代码的错误状态，对上述当前运行的代码进行迭代更新，以得到目标代码，其中，在第一个代码更新轮次中，上述当前运行的代码为上述初始代码。

[0006] 根据本公开的另一方面，提供了一种任务处理方法，包括：响应于任务处理请求，解析上述任务处理请求包括的目标任务的任务文本，得到意图信息；基于上述意图信息，生成初始代码；在测试环境中，基于当前运行的代码的错误状态，对上述当前运行的代码进行迭代更新，以得到目标代码，其中，在第一个代码更新轮次中，上述当前运行的代码为上述初始代码；以及在生产环境中运行上述目标代码，以执行上述目标任务。

[0007] 根据本公开的另一方面，提供了一种代码解释器，包括：代码沙盒，被配置为提供测试环境，上述代码沙盒被配置为接收代码，在上述测试环境中运行上述代码，并生成上述代码的运行反馈信息，其中，上述代码是由大语言模型生成的；错误反思模块，被配置为接收上述运行反馈信息，响应于上述运行反馈信息表示上述代码运行出错，基于上述代码的错误状态，得到会话信息，并向上述大语言模型发送上述会话信息，其中，上述大语言模型被配置为基于上述会话信息和上述代码，生成更新后的代码。

[0008] 根据本公开的另一方面，提供了一种基于大模型的代码更新装置，包括：接收模块，用于接收来自测试环境的运行反馈信息，上述运行反馈信息用于表示代码在上述测试环境中的运行结果，上述代码是利用大语言模型，基于用户意图生成的；第一处理模块，用

于响应于上述运行反馈信息表示上述代码运行出错,基于上述代码的错误状态,得到会话信息;以及第一更新模块,用于利用上述会话信息对上述代码进行更新,得到更新后的代码。

[0009] 根据本公开的另一方面,提供了一种代码生成装置,包括:第一生成模块,用于响应于代码生成请求,基于上述代码生成请求包括的意图信息,生成初始代码;以及第二更新模块,用于在测试环境中,基于当前运行的代码的错误状态,对上述当前运行的代码进行迭代更新,以得到目标代码,其中,在第一个代码更新轮次中,上述当前运行的代码为上述初始代码。

[0010] 根据本公开的另一方面,提供了一种任务处理装置,包括:解析模块,用于响应于任务处理请求,解析上述任务处理请求包括的目标任务的任务文本,得到意图信息;第二生成模块,用于基于上述意图信息,生成初始代码;第三更新模块,用于在测试环境中,基于当前运行的代码的错误状态,对上述当前运行的代码进行迭代更新,以得到目标代码,其中,在第一个代码更新轮次中,上述当前运行的代码为上述初始代码;以及第二处理模块,用于在生产环境中运行上述目标代码,以执行上述目标任务。

[0011] 根据本公开的另一方面,提供了一种电子设备,包括:至少一个处理器;以及与所述至少一个处理器通信连接的存储器;其中,所述存储器存储有可被所述至少一个处理器执行的指令,所述指令被所述至少一个处理器执行,以使所述至少一个处理器能够执行如上所述的方法。

[0012] 根据本公开的另一方面,提供了一种存储有计算机指令的非瞬时计算机可读存储介质,其中,所述计算机指令用于使所述计算机执行如上所述的方法。

[0013] 根据本公开的另一方面,提供了一种计算机程序产品,包括计算机程序,所述计算机程序在被处理器执行时实现如上所述的方法。

[0014] 应当理解,本部分所描述的内容并非旨在标识本公开的实施例的关键或重要特征,也不用于限制本公开的范围。本公开的其它特征将通过以下的说明书而变得容易理解。

附图说明

[0015] 附图用于更好地理解本方案,不构成对本公开的限定。其中:

[0016] 图1示意性示出了根据本公开实施例的可以应用本公开实施例的方法及装置的示例性系统架构。

[0017] 图2示意性示出了根据本公开实施例的基于大模型的代码更新方法的流程图。

[0018] 图3示意性示出了根据本公开实施例的代码更新流程的示意图。

[0019] 图4示意性示出了根据本公开另一实施例的基于大模型的代码更新方法的流程图。

[0020] 图5示意性示出了根据本公开实施例的代码生成方法的流程图。

[0021] 图6示意性示出了根据本公开实施例的任务处理方法的流程图。

[0022] 图7A示意性示出了根据本公开实施例的代码解释器的架构图。

[0023] 图7B示意性示出了根据本公开另一实施例的代码解释器的架构图。

[0024] 图8示意性示出了根据本公开实施例的基于大模型的代码更新装置的框图。

[0025] 图9示意性示出了根据本公开实施例的代码生成装置的框图。

[0026] 图10示意性示出了根据本公开实施例的任务处理装置的框图。

[0027] 图11示出了可以用来实施本公开的实施例的示例电子设备的示意性框图。

具体实施方式

[0028] 以下结合附图对本公开的示范性实施例做出说明,其中包括本公开实施例的各种细节以助于理解,应当将它们认为仅仅是示范性的。因此,本领域普通技术人员应当认识到,可以对这里描述的实施例做出各种改变和修改,而不会背离本公开的范围和精神。同样,为了清楚和简明,以下的描述中省略了对公知功能和结构的描述。

[0029] 代码解释器可以是辅助大语言模型进行代码生成的工具。代码解释器主要是为了解决非专业程序员难以直接进行编程和数据处理的问题。它通过将自然语言需求翻译成可执行的代码,使得不具备深厚编程背景的用户也能实现复杂的编程任务。代码解释器可以有效降低编程的门槛,让更多的用户可以轻松地进行数据处理、代码编辑、图像转换等操作,从而提高了工作效率和创造力。代码解释器在数据分析和可视化、自动化和脚本编写、图像和视频处理、网站和应用开发、科学计算和模拟、技术支持和故障排除等场景得到了广泛应用。

[0030] 然而,当前的代码解释器存在着一定的缺陷,这些缺陷影响了代码生成的质量及效率。例如,在代码执行错误的处理上,缺乏有效的代码纠错机制,从而限制了开发效率的提升。再例如,在面对复杂任务时,如复杂的合并表头数量查询和分析任务时,代码解释器生成的代码指令参差不齐,无法满足高标准的应用需求。

[0031] 有鉴于此,本公开的实施例提供了一种基于大模型的代码更新方法、代码生成方法、任务处理方法及代码解释器,该基于大模型的代码更新方法包括:接收来自测试环境的运行反馈信息,运行反馈信息用于表示代码在测试环境中的运行结果,代码是利用大语言模型,基于用户意图生成的;响应于运行反馈信息表示代码运行出错,基于代码的错误状态,得到会话信息;以及利用会话信息对代码进行更新,得到更新后的代码。

[0032] 图1示意性示出了根据本公开实施例的可以应用本公开实施例的方法及装置的示例性系统架构。

[0033] 需要注意的是,图1所示仅为可以应用本公开实施例的系统架构的示例,以帮助本领域技术人员理解本公开的技术内容,但并不意味着本公开实施例不可以用于其他设备、系统、环境或场景。例如,在另一实施例中,可以应用本公开实施例的方法及装置的示例性系统架构可以包括终端设备,但终端设备可以无需与服务器进行交互,即可实现本公开实施例提供的方法及装置。

[0034] 如图1所示,根据该实施例的系统架构100可以包括终端设备101、102、103,网络104和服务器105。网络104用以在终端设备101、102、103和服务器105之间提供通信链路的介质。网络104可以包括各种连接类型,例如有线和/或无线通信链路等等。

[0035] 用户可以使用终端设备101、102、103通过网络104与服务器105交互,以接收或发送消息等。终端设备101、102、103上可以安装有各种通讯客户端应用,例如知识阅读类应用、网页浏览器应用、搜索类应用、即时通信工具、邮箱客户端和/或社交平台软件等(仅为示例)。

[0036] 终端设备101、102、103可以是具有显示屏并且支持网页浏览的各种电子设备,包

括但不限于智能手机、平板电脑、膝上型便携计算机和台式计算机等等。

[0037] 服务器105可以是提供各种服务的服务器,例如对用户利用终端设备101、102、103所浏览的内容提供支持的后台管理服务器(仅为示例)。后台管理服务器可以对接收到的用户请求等数据进行分析等处理,并将处理结果(例如根据用户请求获取或生成的网页、信息、或数据等)反馈给终端设备。

[0038] 需要说明的是,本公开实施例所提供的方法一般可以由终端设备101、102、或103执行。相应地,本公开实施例所提供的装置也可以设置于终端设备101、102、或103中。

[0039] 或者,本公开实施例所提供的方法一般也可以由服务器105执行。相应地,本公开实施例所提供的装置一般可以设置于服务器105中。本公开实施例所提供的方法也可以由不同于服务器105且能够与终端设备101、102、103和/或服务器105通信的服务器或服务器集群执行。相应地,本公开实施例所提供的装置也可以设置于不同于服务器105且能够与终端设备101、102、103和/或服务器105通信的服务器或服务器集群中。

[0040] 应该理解,图1中的终端设备、网络和服务器的数目仅仅是示意性的。根据实现需要,可以具有任意数目的终端设备、网络和服务器。

[0041] 在本公开的技术方案中,所涉及的用户个人信息的收集、存储、使用、加工、传输、提供、公开和应用等处理,均符合相关法律法规的规定,采取了必要保密措施,且不违背公序良俗。

[0042] 在本公开的技术方案中,在获取或采集用户个人信息之前,均获取了用户的授权或同意。

[0043] 图2示意性示出了根据本公开实施例的基于大模型的代码更新方法的流程图。

[0044] 如图2所示,该方法包括操作S210~S230。

[0045] 在操作S210,接收来自测试环境的运行反馈信息。

[0046] 在操作S220,响应于运行反馈信息表示代码运行出错,基于代码的错误状态,得到会话信息。

[0047] 在操作S230,利用会话信息对代码进行更新,得到更新后的代码。

[0048] 测试环境可以是对代码进行开发和测试用的环境。测试环境的选择及配置可以与该代码的需求相关,例如,该测试环境中可以装载有适于该代码运行的操作系统,该测试环境中可以配置有该操作系统下代码运行所需的环境组件等。测试环境可以由单独的测试用电子设备来提供,或者,测试环境也可以由虚拟机来提供,在此不作限定。

[0049] 测试环境中运行的代码可以指利用大语言模型,基于用户意图生成的代码。该用户意图可以包括代码运行时的操作系统、待生成代码的程序语言种类、需要该代码实现的功能、代码的输入接口信息、代码的输出接口信息等。可以将用户意图包括的各类信息分别写入到一个提示词模板中,得到提示词文本,并将该提示词文本输入到大语言模型中,得到大语言模型输出的代码。可选地,该测试环境中运行的代码也可以是开发人员基于其需求编译得到的代码,在此不作限定。

[0050] 运行反馈信息可以用于表示代码在测试环境中的运行结果,例如表示代码在测试环境中正常运行,表示代码在测试环境中运行出错等。可选地,运行反馈信息中还可以包括其他的用于表征代码运行状态的信息,如错误码等,在此不作限定。

[0051] 代码的错误状态可以用于表示代码面临的问题的复杂程度,该复杂程度的确定方

式在此不作限定。例如,可以根据运行反馈信息中包含的报错信息的数量来确定代码面临的问题的复杂程度,报错信息的数量越多,则表示代码面临的问题越复杂。再例如,可以根据代码的更新次数来确定代码面临的问题的复杂程度,每次得到的更新后的代码可以再次放入测试环境中进行测试,从代码在测试环境中运行出错到利用生成的会话信息对代码进行更新以得到更新后的代码的过程可以视为对代码的一次更新过程,代码的更新次数越多,则表示大语言模型更不容易解决代码面临的问题,即该代码面临的问题相对于大语言模型更为复杂。

[0052] 大语言模型可以从属于一个对话系统,该会话信息可以表示为直接输入到该对话系统的文本信息。在该对话系统中,可以将该会话信息与所需的上下文信息进行拼接,以得到实际应输入大语言模型的文本。可选地,可以基于代码的错误状态,模拟人工提问的方式,生成该会话信息。

[0053] 可以使用大语言模型基于该会话信息对代码进行更新,得到更新后的代码。该大语言模型可以指配置有代码生成插件或代码生成工具的大模型,该代码生成插件或代码生成工具的选择及配置在此不作限定。

[0054] 根据本公开的实施例,可以在测试环境中对利用大语言模型所生成的代码进行测试,并在确认代码运行出错的情况下,基于代码的不同错误状态,进行会话信息的构建,并利用大语言模型基于该会话信息进行代码的更新。在代码处于不同的错误状态时,可以构建不同文本规模的会话信息,从而可以兼顾代码生成的效率和准确性,有效提升代码生成的实用性。

[0055] 下面参考图3~图4,结合具体实施例对图2所示的方法做进一步说明。

[0056] 图3示意性示出了根据本公开实施例的代码更新流程的示意图。

[0057] 如图3所示,该代码更新流程可以在电子设备中执行,该电子设备中可以配置有虚拟机301,该虚拟机301可以用于提供测试环境。利用大语言模型302,基于用户意图可以生成代码303。之后,可以在迭代地对该代码303进行更新,直至得到可以正常运行的目标代码304。

[0058] 在每个代码更新轮次中,该代码303可以被送入虚拟机301中。虚拟机301可以运行该代码303,并将该代码303的运行结果作为运行反馈信息305进行输出。电子设备中的处理器可以对该运行反馈信息305进行解析,在确定该运行反馈信息305表示为代码运行正常的情况下,处理器可以将该代码303视为目标代码304进行输出。该目标代码304可以表示为能够正常运行且符合用户意图的代码。处理器在在确定该运行反馈信息305表示为代码运行出错的情况下,可以确定代码303的错误状态306。基于错误状态306表征的不同状态,可以生成不同的会话信息307。该会话信息307可以作为大语言模型302的至少一部分输入文本,输入到大语言模型302中,得到更新后的代码。该更新后的代码可以作为下一个代码更新轮次中的代码303,被送入到虚拟机301中进行运行测试。

[0059] 代码的错误状态可以基于代码更新流程的执行过程中所产生的各类参数来界定,在此不作限定。

[0060] 代码的错误状态的界定方式在此不作限定,例如,可以采用二分法来界定代码的错误状态,及代码的错误状态可以分为第一状态和第二状态。该第一状态可以表示代码面临的问题为简单问题,该第二状态可以表示代码面临的问题为复杂问题。

[0061] 例如,可以配置有一个计数器,该计数器可以用于对代码更新的轮数进行计量,该计数器的值即表示为代码的已更新次数。可以基于代码的已更新次数来界定代码的错误状态。对于当前的代码更新轮次的代码,在代码的已更新次数小于或等于预设值的情况下,可以确定代码的错误状态表示为第一状态。在代码的已更新次数大于预设值的情况下,确定代码的错误状态表示为第二状态。

[0062] 该预设值可以根据具体应用场景进行设置,例如可以设置为1、2等,在此不作限定。

[0063] 该计数器可以在每个代码更新轮次结束时,进行其计数值的更新,例如,可以响应于利用错误反思信息对代码进行更新,基于自增策略调整计数器的计数值,即该代码的已更新次数。该计数器可以使用计数寄存器来实现,或者,该计数器可以表示为一个预设的计数变量,在此不作限定。该自增策略例如可以表示为基于设定的步长对计数值进行递增处理,该设定的步长例如可以是1。

[0064] 再例如,测试环境中可以配置有调试工具,该调试工具可以展示代码运行过程中所产生的各类问题及各类问题的数量,相应地,该运行反馈信息中可以包含与该各类问题及各类问题的数量相关的信息。可以基于运行反馈信息中包含的问题种类信息及问题数量信息,界定代码的错误状态。在运行反馈信息包括的问题种类信息表示不存在复杂问题,且问题数量信息表示问题的数量小于预设阈值的情况下,可以确定代码的错误状态表示为第一状态。在运行反馈信息包括的问题种类信息表示存在复杂问题,以及/或者,问题数量信息表示问题的数量大于或等于预设阈值的情况下,可以确定代码的错误状态表示为第二状态。该预设阈值可以根据具体应用场景进行设置,在此不作限定。

[0065] 对于较为简单的问题,如变量类型的定义错误、方法或类的调用错误等,大语言模型一般可以通过有限次数的更迭即可生成可以能够较好地运行的代码。而对于较为复杂的问题,如类或方法的算法逻辑错误等,大语言模型解决该错误所需的信息及更迭次数一般较多。可选地,分别针对于代码的不同错误状态,可以创建不同文本长度的会话信息,以自适应地基于不同复杂程度的问题,采用不同的规模的文本进行代码的修正,从而可以兼顾代码生成的效率和准确性。

[0066] 根据本公开的实施例,在代码的错误状态表示为第一状态的情况下,基于代码的错误状态,得到会话信息,可以包括如下操作:

[0067] 基于运行反馈信息中包括的错误码,确定当前错误信息;以及,基于当前错误信息和工具调用信息,得到会话信息。

[0068] 测试环境中配置的调试工具可以将代码运行过程中的问题转译为错误码,每个错误码可以用于指示一个类型的问题及该问题在代码中的位置。例如,错误码可以表示为:“编码:001;位置:文件A中的第B行”。

[0069] 电子设备中可以配置有关系表,该关系表可以用于记录错误码中的编码与错误信息模板之间的联系,错误信息模板可以是用于对一个类型的问题进行描述的模板。例如,若编码001所表示的问题为变量调用错误,则与该编码对应的错误信息模板可以表示为“在文件X的第X行中存在变量调用错误的问题”。基于错误码中的编码,可以确定对应的错误信息模板,再将错误码中的位置信息填入到该错误信息模板中,即可得到当前错误信息。

[0070] 工具调用信息中可以包括与特定工具相关的声明文本,利用该声明文本,大语言

模型可以通过调用该特定工具,来实现相应功能。例如,该特定工具可以包括代码生成工具、用户意图解析工具等。大语言模型在获取该工具调用信息后,可以分别加载用户意图解析工具和代码生成工具,利用用户意图解析工具,对用户输入的文本信息进行解析,并重组得到模型的输入提示文本,大语言模型对该输入提示文本的输出特征可以经由代码生成工具处理为一串可执行的代码。通过在会话信息中添加工具调用信息,可以增加代码生成准确率。

[0071] 可以分别将当前错误信息和工具调用信息填入到一个模板中,以得到会话信息。

[0072] 根据本公开的实施例,在代码的错误状态表示为第二状态的情况下,基于代码的错误状态,得到会话信息,可以包括如下操作:

[0073] 基于运行反馈信息中包括的错误码,确定当前错误信息,当前错误信息表示当前代码更新轮次的错误信息;以及,基于当前错误信息、代码的历史错误信息、代码的历史会话信息和代码的元信息,得到会话信息。

[0074] 根据本公开的实施例,历史错误信息可以包括多个历史代码更新轮次各自的错误信息,历史会话信息可以包括多个历史代码更新轮次各自的会话信息,元信息包括代码生成任务信息、模型信息和工具调用信息中的至少一项。

[0075] 代码生成任务信息可以基于用户意图得到,即该代码生成任务信息可以包括代码运行时的操作系统、待生成代码的程序语言种类、需要该代码实现的功能、代码的输入接口信息、代码的输出接口信息等。

[0076] 模型信息可以表示代码生成使所使用的大语言模型的相关信息,包括大语言模型的版本号、需要加载的插件或工具的信息等。

[0077] 可选地,在元信息中包含工具调用信息的情况下,在记录历史更新轮次的会话信息时,可以将该会话信息中包含的工具调用信息删除,以减少会话信息的整体文本长度。

[0078] 根据本公开的实施例,通过在会话信息中添加如上的元信息,可以有效提升当前轮次的代码生成效果。

[0079] 可选地,在多个历史错误信息中,可以存在重要性较低的错误信息,这些错误信息无法为当前代码更新轮次中代码所面临的问题的解决提供帮助。对于多个历史错误信息可以有选择地进行筛选,以得到更为有效的错误信息,并基于该更为有效的错误信息进行会话信息的生成。

[0080] 根据本公开的实施例,基于当前错误信息、代码的历史错误信息、代码的历史会话信息和代码的元信息,得到会话信息,可以包括如下操作:

[0081] 基于信息选择策略,从代码的历史错误信息中选择得到目标错误信息;以及,将当前错误信息、历史错误信息、历史会话信息和元信息进行拼接,得到会话信息。

[0082] 信息选择策略可以表示为从多个历史错误信息中筛选到更为有效的错误信息的方式。可选地,该方式可以包括最相关选择、最近选择等,在此不作限定。

[0083] 最相关选择可以表示为从多个历史错误信息中选择与当前代码更新轮次中的错误信息相关性最强的错误信息,作为目标错误信息。例如,可以通过关键词匹配、向量特征匹配的方式,计算当前代码更新轮次中的错误信息与多个历史错误信息各自的相似度,并确定与最大的相似度对应的历史错误信息为目标错误信息。再例如,基于当前代码更新轮次中获取的错误码,从多个历史错误信息中选择与该错误码对应的错误信息作为目标错误

信息。

[0084] 最近选择可以表示为选择时间最接近的历史错误信息作为目标错误信息,即可以从多个历史代码更新轮次各自的错误信息中,选择与当前代码更新轮次最接近的历史代码更新轮次的错误信息作为目标错误信息。

[0085] 根据本公开的实施例,通过对历史错误信息进行筛选,可以减少噪声信息的干扰,提升生成代码的准确性。

[0086] 在得到会话信息后,可以借助大语言模型,利用该会话信息对代码进行更新。

[0087] 根据本公开的实施例,利用会话信息对代码进行更新,得到更新后的代码,可以包括如下操作:

[0088] 将会话信息和代码填入提示模板,得到提示文本;以及,将提示文本输入大语言模型,以便大语言模型以会话信息作为提示信息,基于代码进行新代码的生成,得到更新后的代码。

[0089] 例如,提示模版可以表示为:“当前有如下信息:会话信息:message;代码:code。请根据所提供的会话信息,对代码进行修正”。可以使用会话信息替换该提示模板中的message字段,使用代码替换该提示模板中的code字段,以得到提示文本。

[0090] 图4示意性示出了根据本公开另一实施例的基于大模型的代码更新方法的流程图。

[0091] 如图4所示,该方法包括操作S401~S410。

[0092] 在操作S401,在测试环境中运行代码。

[0093] 在操作S402,判断代码是否正常运行。在确定代码正常运行的情况下,执行操作S403。在确定代码运行出错的情况下,执行操作S404。

[0094] 在操作S403,确定该代码为符合用户意图的目标代码。

[0095] 在操作S404,从测试环境的反馈信息中获取当前错误信息。

[0096] 在操作S405,判断代码的已更新次数是否大于预设值。在确定代码的已更新次数大于预设值的情况下,执行操作S406。在确定代码的已更新次数小于或等于预设值的情况下,执行操作S408。

[0097] 在操作S406,从多个历史代码更新轮次各自的历史错误信息中确定目标错误信息。

[0098] 在操作S407,将当前错误信息、目标错误信息、历史会话信息和元信息进行拼接,得到会话信息。在完成操作S407后,可以执行操作S409。

[0099] 在操作S408,将当前错误信息与工具调用信息拼接,得到会话信息。

[0100] 在操作S409,利用会话信息对代码进行更新,得到更新后的代码。

[0101] 在操作S410,记录当前代码更新轮次的错误信息和会话信息,并对代码的已更新次数进行递增处理。在完成操作S410之后,可以将更新后的代码作为下一个代码更新轮次的代码,并执行操作S401。

[0102] 根据本公开的实施例,可以在测试环境中对利用大语言模型所生成的代码进行测试,并在确认代码运行出错的情况下,基于代码的不同错误状态,进行会话信息的构建,并利用大语言模型基于该会话信息进行代码的更新。在代码处于不同的错误状态时,可以构建不同文本规模的会话信息,从而可以兼顾代码生成的效率和准确性,有效提升代码生成

的实用性。

[0103] 图5示意性示出了根据本公开实施例的代码生成方法的流程图。

[0104] 如图5所示,该方法包括操作S510~S520。

[0105] 在操作S510,响应于代码生成请求,基于代码生成请求包括的意图信息,生成初始代码。

[0106] 在操作S510,在测试环境中,基于当前运行的代码的错误状态,对当前运行的代码进行迭代更新,以得到目标代码。

[0107] 根据本公开的实施例,当前运行的代码可以包括该初始代码,即在第一个代码更新轮次中,当前运行的代码为初始代码。

[0108] 根据本公开的实施例,对当前运行的代码进行迭代更新可以使用如上所述的基于大模型的代码更新方法来实现,在此不再赘述。

[0109] 图6示意性示出了根据本公开实施例的任务处理方法的流程图。

[0110] 如图6所示,该方法包括操作S610~S640。

[0111] 在操作S610,响应于任务处理请求,解析任务处理请求包括的目标任务的任务文本,得到意图信息。

[0112] 在操作S620,基于意图信息,生成初始代码。

[0113] 在操作S630,在测试环境中,基于当前运行的代码的错误状态,对当前运行的代码进行迭代更新,以得到目标代码。

[0114] 在操作S640,在生产环境中运行目标代码,以执行目标任务。

[0115] 根据本公开的实施例,当前运行的代码可以包括该初始代码,即在第一个代码更新轮次中,当前运行的代码为初始代码。

[0116] 根据本公开的实施例,对当前运行的代码进行迭代更新可以使用如上所述的基于大模型的代码更新方法来实现,在此不再赘述。

[0117] 如上所述的基于大模型的代码更新方法、代码生成方法及任务处理方法可以使用代码解释器来实现。以下以利用代码解释器实现基于大模型的代码更新方法为例,对代码解释器的架构进行说明。

[0118] 图7A示意性示出了根据本公开实施例的代码解释器的架构图。

[0119] 如图7A所示,代码解释器的架构自上而下可以分为三层。其中,第一层可以为代码沙盒710,第二层可以为错误反思模块720,第三层可以为大语言模型730。

[0120] 根据本公开的实施例,代码沙盒710可以被配置为提供测试环境。代码沙盒710可以被配置为接收代码,在测试环境中运行代码,并生成代码的运行反馈信息。在代码沙盒710中运行的代码可以是由大语言模型730生成的。

[0121] 根据本公开的实施例,错误反思模块720可以被配置为接收运行反馈信息,响应于运行反馈信息表示代码运行出错,基于代码的错误状态,得到会话信息,并向大语言模型730发送会话信息。大语言模型730可以被配置为基于会话信息和代码,生成更新后的代码。该更新后的代码可以再次被送入代码沙盒710中进行测试。

[0122] 可选地,代码解释器还可以包括其他功能模块,以使得代码解释器可以自适应地调节所生成的会话信息的规模。

[0123] 图7B示意性示出了根据本公开另一实施例的代码解释器的架构图。

[0124] 如图7B所示,代码解释器的第一层中还可以包括错误管理模块740,第二层中还可以包括记忆管理模块750。

[0125] 根据本公开的实施例,错误管理模块740可以被配置为记录多个错误码,和与错误码对应的错误信息。

[0126] 根据本公开的实施例,错误反思模块720可以被配置为在代码的错误状态表示为第一状态的情况下,通过错误管理模块740,获取与运行反馈信息中包括的错误码对应的当前错误信息,并基于当前错误信息和工具调用信息,得到会话信息。该第一状态可以表示代码的已更新次数小于或等于预设值。

[0127] 根据本公开的实施例,记忆管理模块750可以被配置为记录代码的元信息,并记录历史代码更新轮次中的错误信息和会话信息,得到历史错误信息和历史会话信息。

[0128] 根据本公开的实施例,错误反思模块720可以被配置为在代码的错误状态表示为第二状态的情况下,通过错误管理模块740,获取与运行反馈信息中包括的错误码对应的当前错误信息,通过记忆管理模块750,获取元信息、历史错误信息和历史会话信息,并基于当前错误信息、元信息、历史错误信息和历史会话信息,得到会话信息。该第二状态表示代码的已更新次数大于预设值。

[0129] 根据本公开的实施例,利用该代码解释器,在执行代码生成、代码修正等任务时,可以有效提升任务处理的效率。

[0130] 图8示意性示出了根据本公开实施例的基于大模型的代码更新装置的框图。

[0131] 如图8所示,基于大模型的代码更新装置800包括接收模块810、第一处理模块820和第一更新模块830。

[0132] 接收模块810,用于接收来自测试环境的运行反馈信息,运行反馈信息用于表示代码在测试环境中的运行结果,代码是利用大语言模型,基于用户意图生成的。

[0133] 第一处理模块820,用于响应于运行反馈信息表示代码运行出错,基于代码的错误状态,得到会话信息。

[0134] 第一更新模块830,用于利用会话信息对代码进行更新,得到更新后的代码。

[0135] 根据本公开的实施例,代码的错误状态是基于代码的已更新次数来确定的。

[0136] 根据本公开的实施例,基于大模型的代码更新装置800还包括第一确定模块和第二确定模块。

[0137] 第一确定模块,用于在代码的已更新次数小于或等于预设值的情况下,确定代码的错误状态表示为第一状态。

[0138] 第二确定模块,用于在代码的已更新次数大于预设值的情况下,确定代码的错误状态表示为第二状态。

[0139] 根据本公开的实施例,第一处理模块820包括第一处理子模块和第二处理子模块。

[0140] 第一处理子模块,用于在代码的错误状态表示为第一状态的情况下,基于运行反馈信息中包括的错误码,确定当前错误信息。

[0141] 第二处理子模块,用于基于当前错误信息和工具调用信息,得到会话信息。

[0142] 根据本公开的实施例,第一处理模块820包括第三处理子模块和第四处理子模块。

[0143] 第三处理子模块,用于在代码的错误状态表示为第二状态的情况下,基于运行反馈信息中包括的错误码,确定当前错误信息,当前错误信息表示当前代码更新轮次的错误

信息。

[0144] 第四处理子模块,用于基于当前错误信息、代码的历史错误信息、代码的历史会话信息和代码的元信息,得到会话信息,其中,历史错误信息包括多个历史代码更新轮次各自的错误信息,历史会话信息包括多个历史代码更新轮次各自的会话信息,元信息包括代码生成任务信息、模型信息和工具调用信息中的至少一项。

[0145] 根据本公开的实施例,第四处理子模块包括第一处理单元和第二处理单元。

[0146] 第一处理单元,用于基于信息选择策略,从代码的历史错误信息中选择得到目标错误信息。

[0147] 第二处理单元,用于将当前错误信息、目标错误信息、历史会话信息和元信息进行拼接,得到会话信息。

[0148] 根据本公开的实施例,第一处理单元包括第一处理子单元和第二处理子单元。

[0149] 第一处理子单元,用于从多个历史代码更新轮次各自的错误信息中,选择与当前代码更新轮次最接近的历史代码更新轮次的错误信息作为目标错误信息。

[0150] 第二处理子单元,用于从多个历史代码更新轮次各自的错误信息中,选择与错误码对应的错误信息作为目标错误信息。

[0151] 根据本公开的实施例,基于大模型的代码更新装置800还包括调整模块。

[0152] 调整模块,用于响应于利用错误反思信息对代码进行更新,基于自增策略调整代码的已更新次数。

[0153] 根据本公开的实施例,第一更新模块830包括第一更新子模块和第二更新子模块。

[0154] 第一更新子模块,用于将会话信息和代码填入提示模板,得到提示文本。

[0155] 第二更新子模块,用于将提示文本输入大语言模型,以便大语言模型以会话信息作为提示信息,基于代码进行新代码的生成,得到更新后的代码。

[0156] 根据本公开的实施例,基于大模型的代码更新装置800还包括第三确定模块。

[0157] 第三确定模块,用于响应于运行反馈信息表示代码运行正常,确定代码为符合用户意图的目标代码。

[0158] 图9示意性示出了根据本公开实施例的代码生成装置的框图。

[0159] 如图9所示,代码生成装置900包括第一生成模块910和第二更新模块920。

[0160] 第一生成模块910,用于响应于代码生成请求,基于代码生成请求包括的意图信息,生成初始代码。

[0161] 第二更新模块920,用于在测试环境中,基于当前运行的代码的错误状态,对当前运行的代码进行迭代更新,以得到目标代码,其中,在第一个代码更新轮次中,当前运行的代码为初始代码。

[0162] 根据本公开的实施例,对当前运行的代码进行迭代更新可以使用如上所述的基于大模型的代码更新方法来实现,在此不再赘述。

[0163] 图10示意性示出了根据本公开实施例的任务处理装置的框图。

[0164] 如图10所示,任务处理装置1000可以包括解析模块1010、第二生成模块1020、第三更新模块1030和第二处理模块1040。

[0165] 解析模块1010,用于响应于任务处理请求,解析任务处理请求包括的目标任务的任务文本,得到意图信息。

[0166] 第二生成模块1020,用于基于意图信息,生成初始代码。

[0167] 第三更新模块1030,用于在测试环境中,基于当前运行的代码的错误状态,对当前运行的代码进行迭代更新,以得到目标代码,其中,在第一个代码更新轮次中,当前运行的代码为初始代码。

[0168] 第二处理模块1040,用于在生产环境中运行目标代码,以执行目标任务。

[0169] 根据本公开的实施例,对当前运行的代码进行迭代更新可以使用如上所述的基于大模型的代码更新方法来实现,在此不再赘述。

[0170] 根据本公开的实施例,本公开还提供了一种电子设备、一种可读存储介质和一种计算机程序产品。

[0171] 根据本公开的实施例,一种电子设备,包括:至少一个处理器;以及与至少一个处理器通信连接的存储器;其中,存储器存储有可被至少一个处理器执行的指令,指令被至少一个处理器执行,以使至少一个处理器能够执行如上所述的方法。

[0172] 根据本公开的实施例,一种存储有计算机指令的非瞬时计算机可读存储介质,其中,计算机指令用于使计算机执行如上所述的方法。

[0173] 根据本公开的实施例,一种计算机程序产品,包括计算机程序,计算机程序在被处理器执行时实现如上所述的方法。

[0174] 图11示出了可以用来实施本公开的实施例的示例电子设备的示意性框图。电子设备旨在表示各种形式的数字计算机,诸如,膝上型计算机、台式计算机、工作台、个人数字助理、服务器、刀片式服务器、大型计算机、和其它适合的计算机。电子设备还可以表示各种形式的移动装置,诸如,个人数字处理、蜂窝电话、智能电话、可穿戴设备和其它类似的计算装置。本文所示的部件、它们的连接和关系、以及它们的功能仅仅作为示例,并且不意在限制本文中描述的和/或者要求的本公开的实现。

[0175] 如图11所示,电子设备1100包括计算单元1101,其可以根据存储在只读存储器(ROM)1102中的计算机程序或者从存储单元1108加载到随机访问存储器(RAM)1103中的计算机程序,来执行各种适当的动作和处理。在RAM 1103中,还可存储设备1100操作所需的各种程序和数据。计算单元1101、ROM 1102以及RAM 1103通过总线1104彼此相连。输入/输出(I/O)接口1105也连接至总线1104。

[0176] 设备1100中的多个部件连接至输入/输出(I/O)接口1105,包括:输入单元1106,例如键盘、鼠标等;输出单元1107,例如各种类型的显示器、扬声器等;存储单元1108,例如磁盘、光盘等;以及通信单元1109,例如网卡、调制解调器、无线通信收发机等。通信单元1109允许设备1100通过诸如因特网的计算机网络和/或各种电信网络与其他设备交换信息/数据。

[0177] 计算单元1101可以是各种具有处理和计算能力的通用和/或专用处理组件。计算单元1101的一些示例包括但不限于中央处理单元(CPU)、图形处理单元(GPU)、各种专用的人工智能(AI)计算芯片、各种运行机器学习模型算法的计算单元、数字信号处理器(DSP)、以及任何适当的处理器、控制器、微控制器等。计算单元1101执行上文所描述的各个方法和处理,例如代码更新方法、代码生成方法和任务处理方法。例如,在一些实施例中,代码更新方法、代码生成方法和任务处理方法可被实现为计算机软件程序,其被有形地包含于机器可读介质,例如存储单元1108。在一些实施例中,计算机程序的部分或者全部可以经由ROM

1102和/或通信单元1109而被载入和/或安装到设备1100上。当计算机程序加载到RAM 1103并由计算单元1101执行时,可以执行上文描述的代码更新方法、代码生成方法和任务处理方法的一个或多个步骤。备选地,在其他实施例中,计算单元1101可以通过其他任何适当的方式(例如,借助于固件)而被配置为执行代码更新方法、代码生成方法和任务处理方法。

[0178] 本文中以上描述的系统和技术的各种实施方式可以在数字电子电路系统、集成电路系统、场可编程门阵列(FPGA)、专用集成电路(ASIC)、专用标准产品(ASSP)、芯片上系统的系统(SOC)、复杂可编程逻辑设备(CPLD)、计算机硬件、固件、软件、和/或它们的组合中实现。这些各种实施方式可以包括:实施在一个或者多个计算机程序中,该一个或者多个计算机程序可在包括至少一个可编程处理器的可编程系统上执行和/或解释,该可编程处理器可以是专用或者通用可编程处理器,可以从存储系统、至少一个输入装置、和至少一个输出装置接收数据和指令,并且将数据和指令传输至该存储系统、该至少一个输入装置、和该至少一个输出装置。

[0179] 用于实施本公开的方法的程序代码可以采用一个或多个编程语言的任何组合来编写。这些程序代码可以提供给通用计算机、专用计算机或其他可编程数据处理装置的处理器或控制器,使得程序代码当由处理器或控制器执行时使流程图和/或框图中所规定的功能/操作被实施。程序代码可以完全在机器上执行、部分地在机器上执行,作为独立软件包部分地在机器上执行且部分地在远程机器上执行或完全在远程机器或服务器上执行。

[0180] 在本公开的上下文中,机器可读介质可以是有形的介质,其可以包含或存储以供指令执行系统、装置或设备使用或与指令执行系统、装置或设备结合地使用的程序。机器可读介质可以是机器可读信号介质或机器可读储存介质。机器可读介质可以包括但不限于电子的、磁性的、光学的、电磁的、红外的、或半导体系统、装置或设备,或者上述内容的任何合适组合。机器可读存储介质的更具体示例会包括基于一个或多个线的电气连接、便携式计算机盘、硬盘、随机存取存储器(RAM)、只读存储器(ROM)、可擦除可编程只读存储器(EPROM或快闪存储器)、光纤、便捷式紧凑盘只读存储器(CD-ROM)、光学储存设备、磁储存设备、或上述内容的任何合适组合。

[0181] 为了提供与用户的交互,可以在计算机上实施此处描述的系统和技术,该计算机具有:用于向用户显示信息的显示装置(例如,CRT(阴极射线管)或者LCD(液晶显示器)监视器);以及键盘和指向装置(例如,鼠标或者轨迹球),用户可以通过该键盘和该指向装置来将输入提供给计算机。其它种类的装置还可以用于提供与用户的交互;例如,提供给用户的反馈可以是任何形式的传感反馈(例如,视觉反馈、听觉反馈、或者触觉反馈);并且可以用任何形式(包括声输入、语音输入或者、触觉输入)来接收来自用户的输入。

[0182] 可以将此处描述的系统和技术实施在包括后台部件的计算系统(例如,作为数据服务器)、或者包括中间件部件的计算系统(例如,应用服务器)、或者包括前端部件的计算系统(例如,具有图形用户界面或者网络浏览器的用户计算机,用户可以通过该图形用户界面或者该网络浏览器来与此处描述的系统和技术的实施方式交互)、或者包括这种后台部件、中间件部件、或者前端部件的任何组合的计算系统中。可以通过任何形式或者介质的数字数据通信(例如,通信网络)来将系统的部件相互连接。通信网络的示例包括:局域网(LAN)、广域网(WAN)和互联网。

[0183] 计算机系统可以包括客户端和服务器。客户端和服务器一般远离彼此并且通常通

过通信网络进行交互。通过在相应的计算机上运行并且彼此具有客户端-服务器关系的计算机程序来产生客户端和服务器的关系。服务器可以是云服务器,也可以是分布式系统的服务器,或者是结合了区块链的服务器。

[0184] 应该理解,可以使用上面所示的各种形式的流程,重新排序、增加或删除步骤。例如,本发公开中记载的各步骤可以并行地执行也可以顺序地执行也可以不同的次序执行,只要能够实现本公开公开的技术方案所期望的结果,本文在此不进行限制。

[0185] 上述具体实施方式,并不构成对本公开保护范围的限制。本领域技术人员应该明白的是,根据设计要求和其他因素,可以进行各种修改、组合、子组合和替代。任何在本公开的精神和原则之内所作的修改、等同替换和改进等,均应包含在本公开保护范围之内。

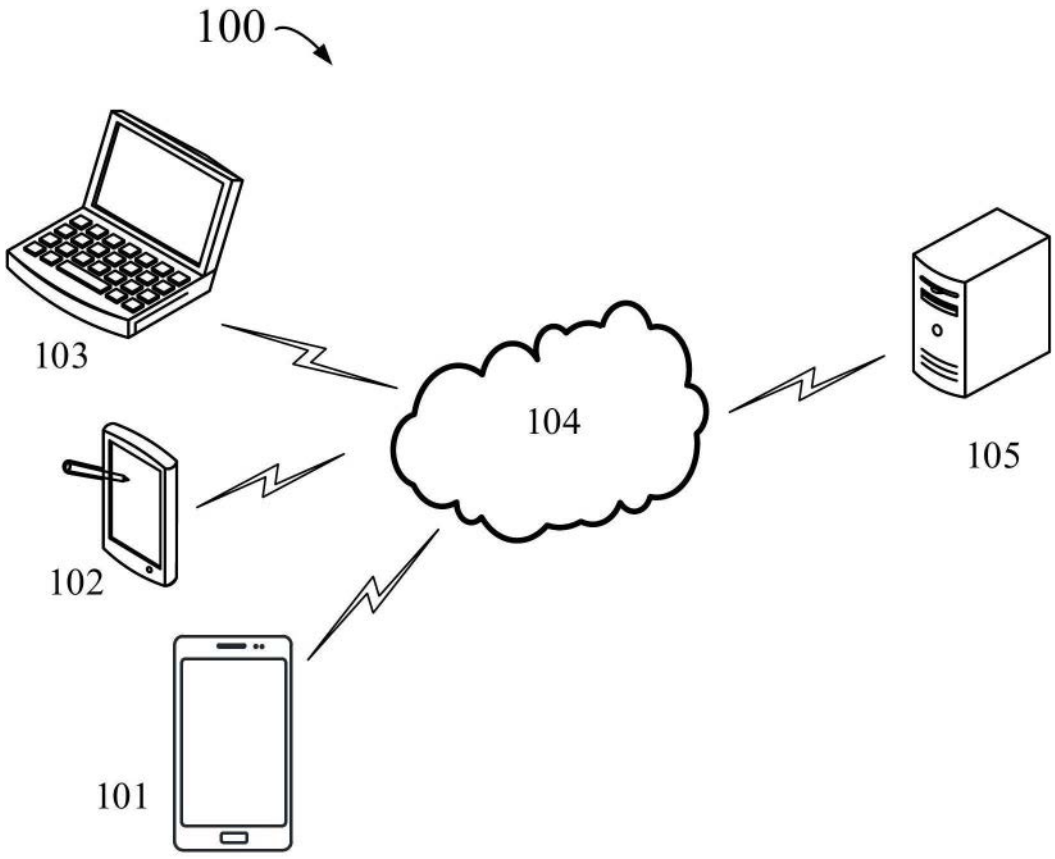

100
103
104
105
102
101

图1

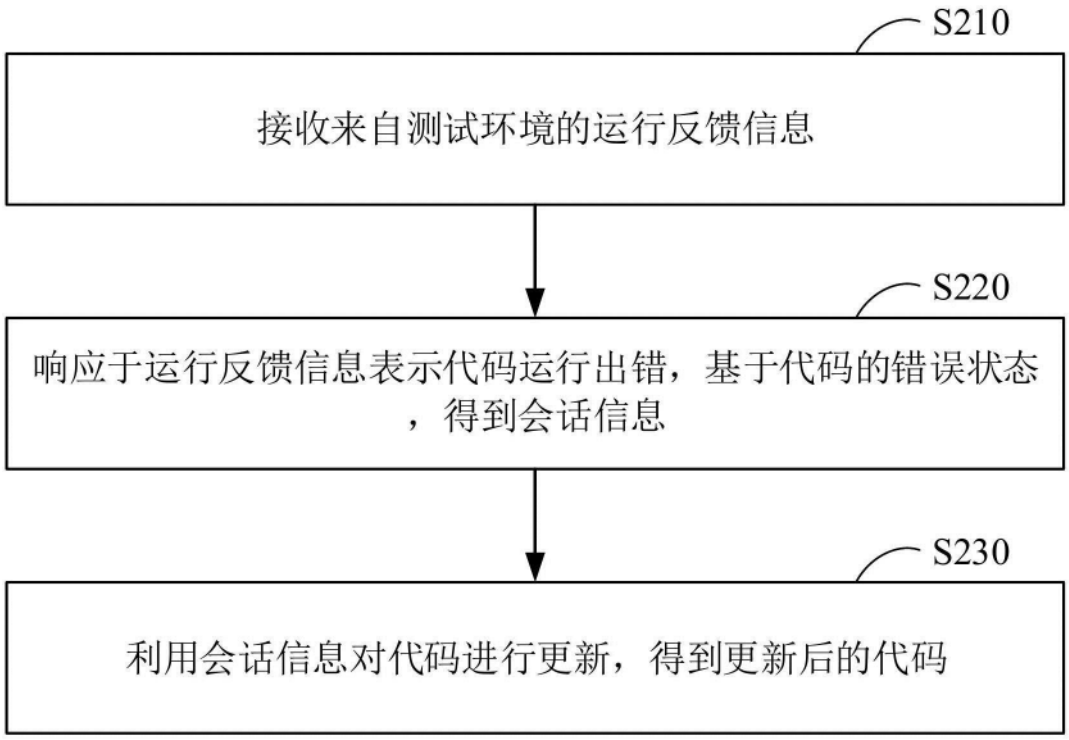

S210
接收来自测试环境的运行反馈信息
S220
响应于运行反馈信息表示代码运行出错，基于代码的错误状态，得到会话信息
S230
利用会话信息对代码进行更新，得到更新后的代码

图2

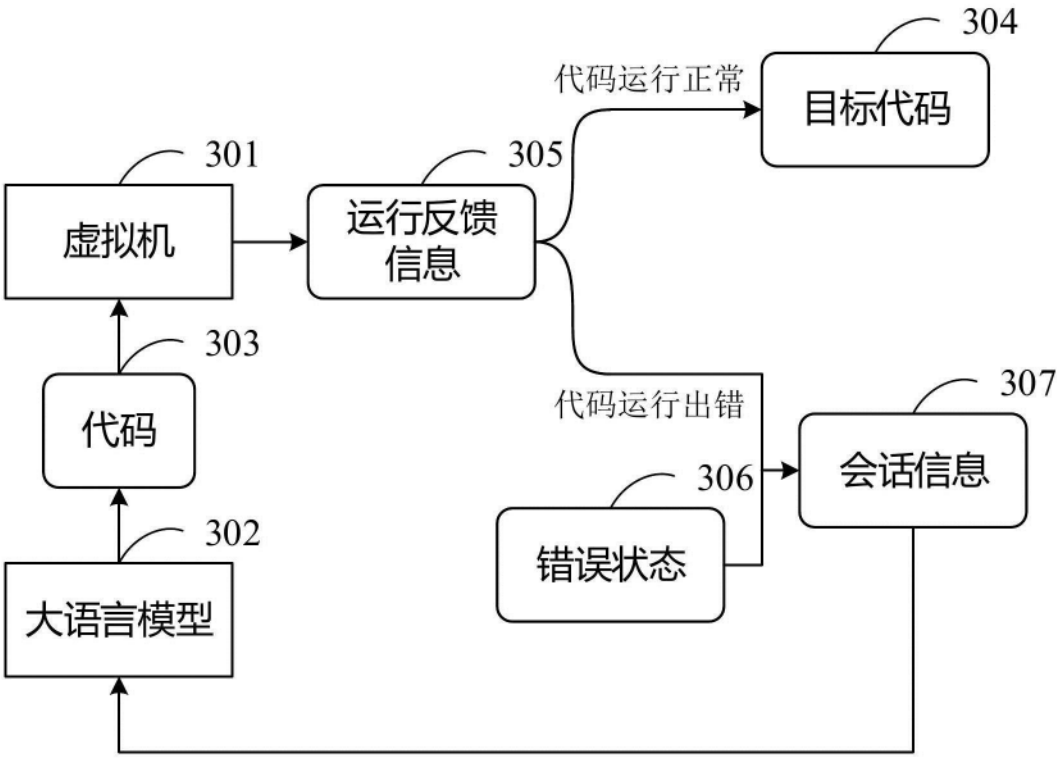

304
代码运行正常
目标代码
301
305
虚拟机
运行反馈信息
303
代码
代码运行出错
307
306
会话信息
错误状态
302
大语言模型

图3

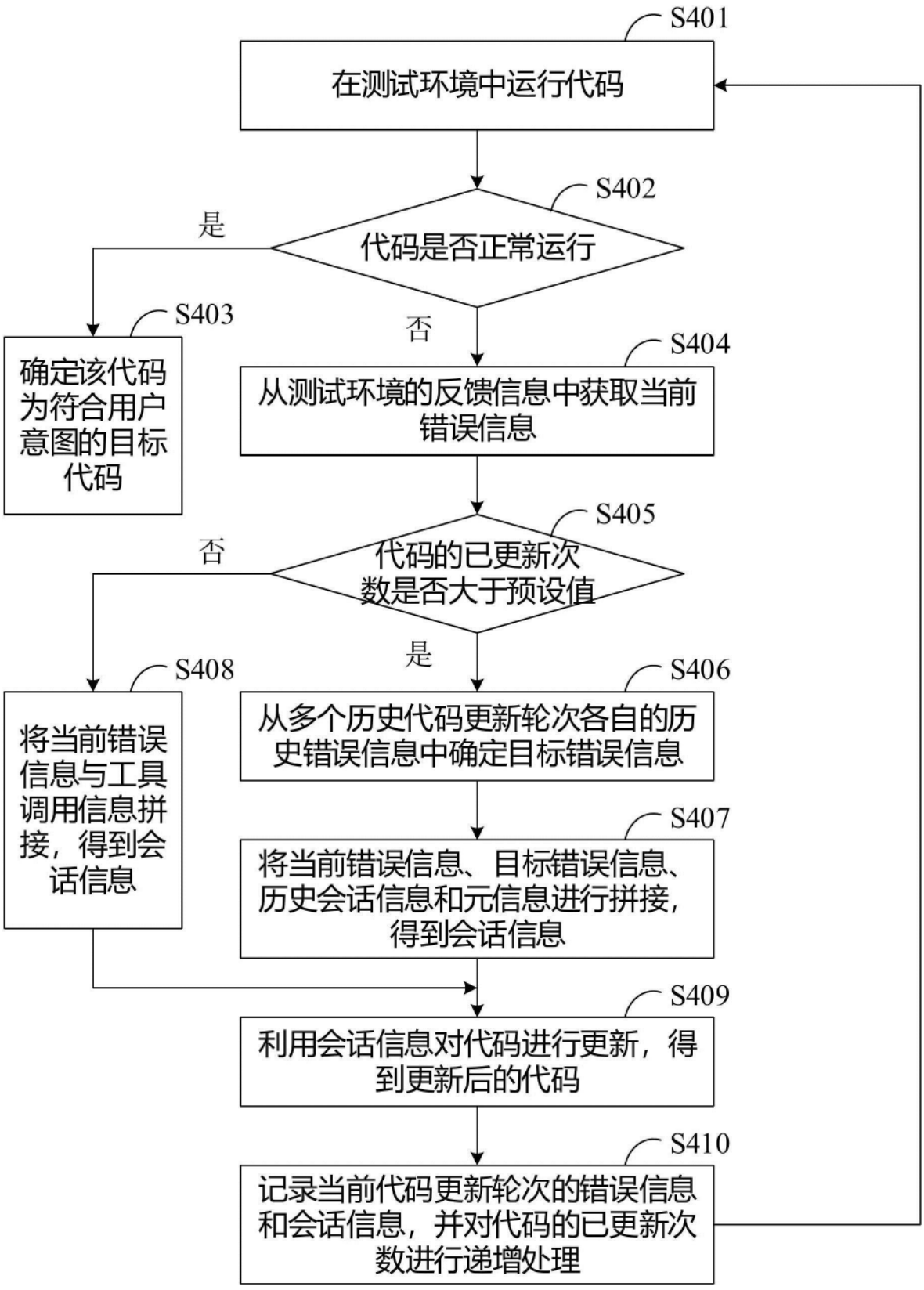

S401
在测试环境中运行代码
S402
代码是否正常运行
是
否
S403
确定该代码为符合用户意图的目标代码
S404
从测试环境的反馈信息中获取当前错误信息
S405
代码的已更新次数是否大于预设值
否
是
S408
将当前错误信息与工具调用信息拼接，得到会话信息
S406
从多个历史代码更新轮次各自的历史错误信息中确定目标错误信息
S407
将当前错误信息、目标错误信息、历史会话信息和元信息进行拼接，得到会话信息
S409
利用会话信息对代码进行更新，得到更新后的代码
S410
记录当前代码更新轮次的错误信息和会话信息，并对代码的已更新次数进行递增处理

图4

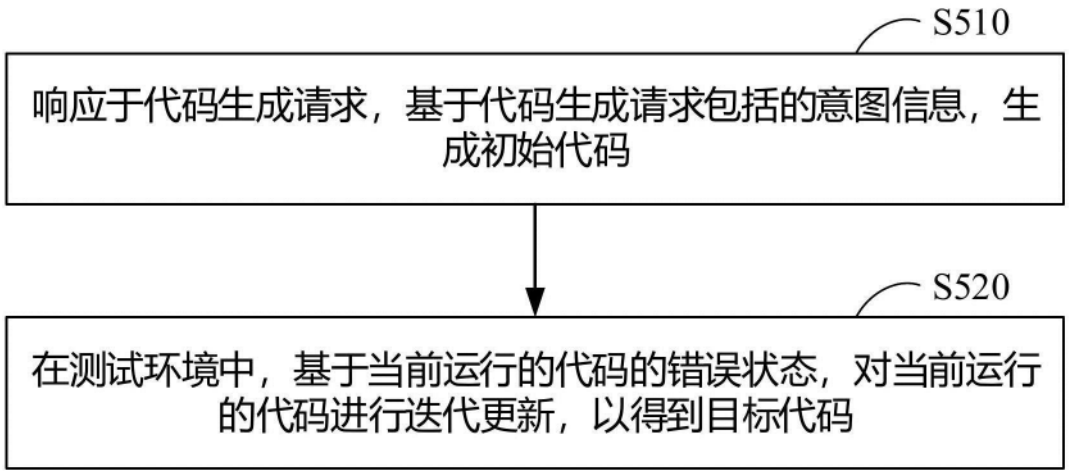

S510
响应于代码生成请求，基于代码生成请求包括的意图信息，生成初始代码
S520
在测试环境中，基于当前运行的代码的错误状态，对当前运行的代码进行迭代更新，以得到目标代码

图5

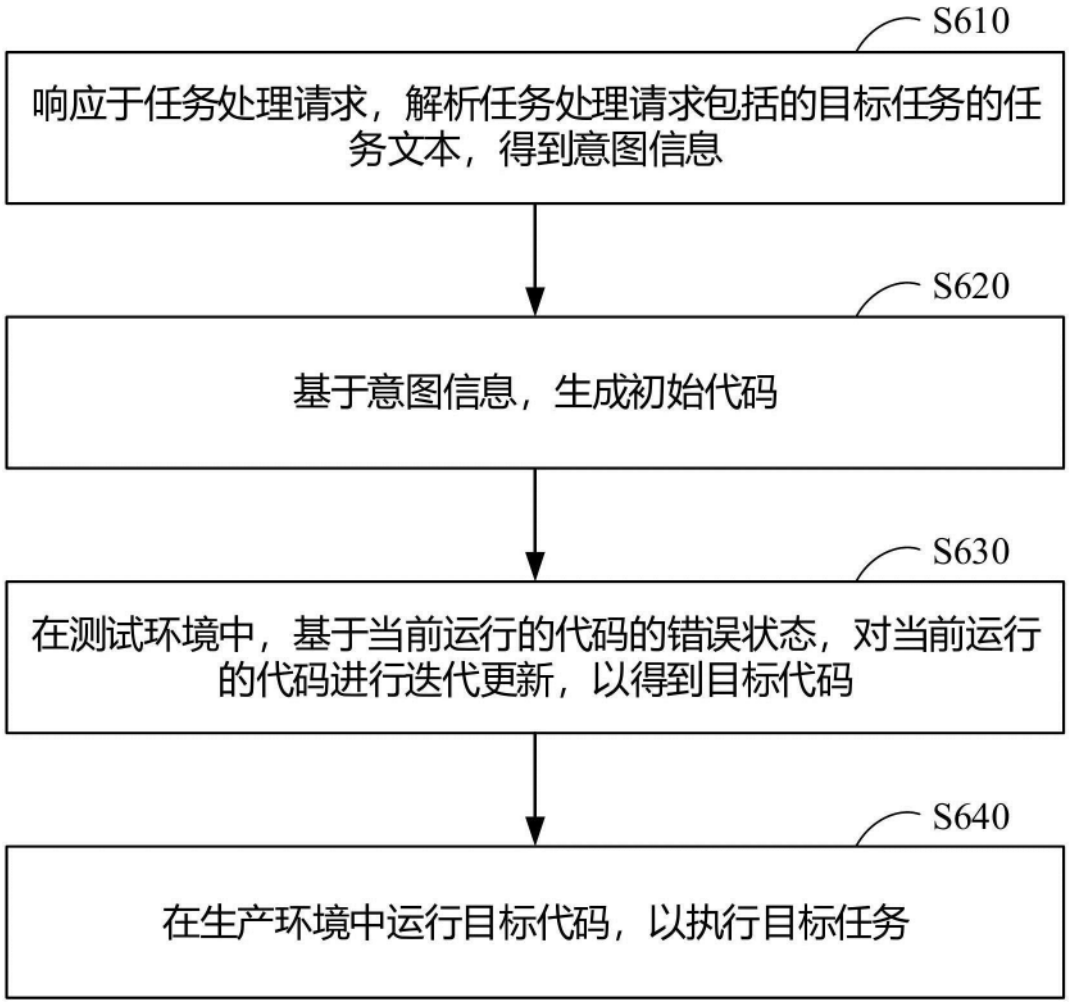

S610
响应于任务处理请求，解析任务处理请求包括的目标任务的任务文本，得到意图信息
S620
基于意图信息，生成初始代码
S630
在测试环境中，基于当前运行的代码的错误状态，对当前运行的代码进行迭代更新，以得到目标代码
S640
在生产环境中运行目标代码，以执行目标任务

图6

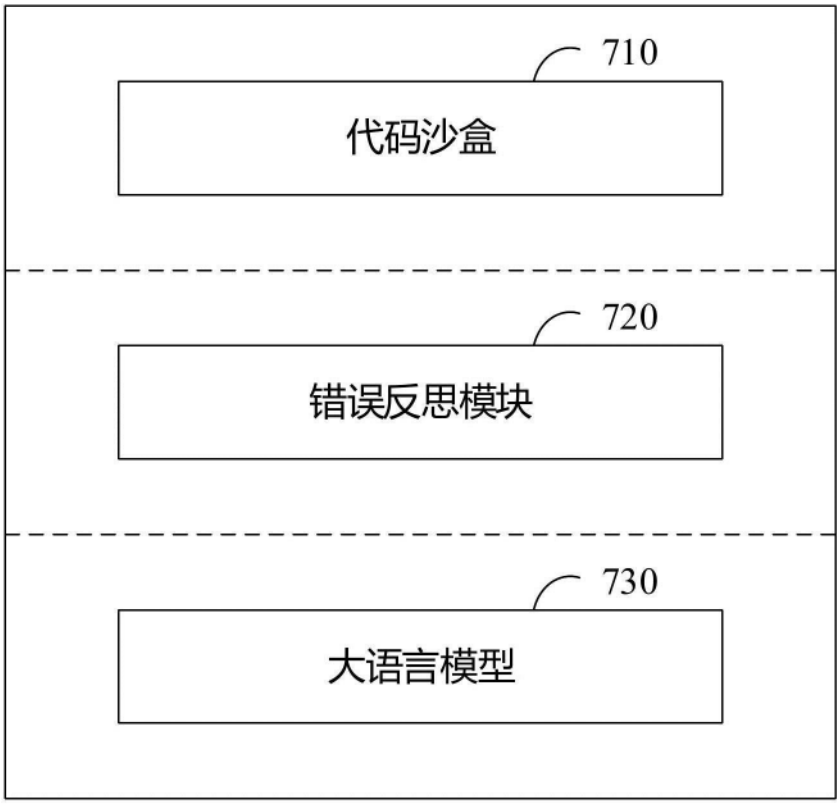

710
代码沙盒
720
错误反思模块
730
大语言模型

图7A

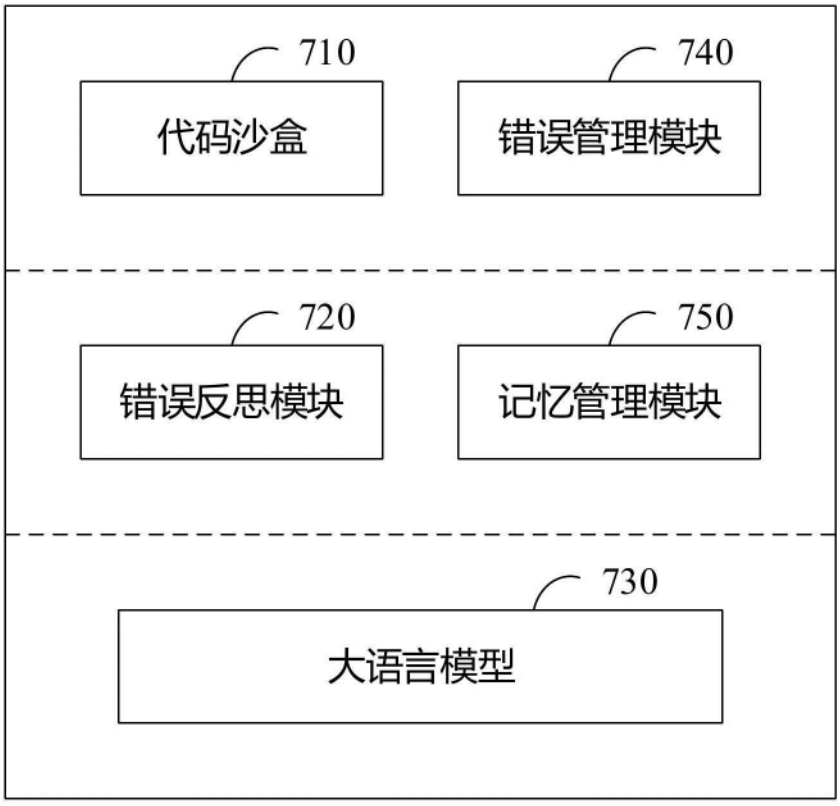

710
代码沙盒
740
错误管理模块
720
错误反思模块
750
记忆管理模块
730
大语言模型

图7B

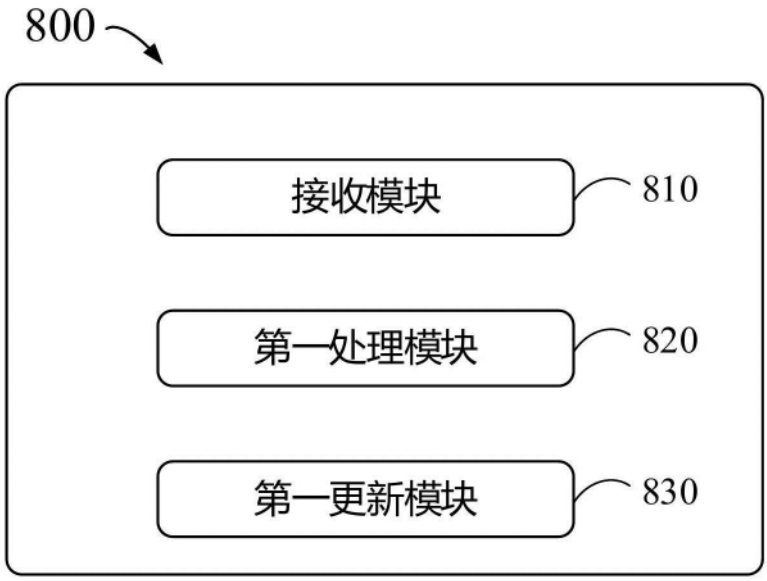


图8

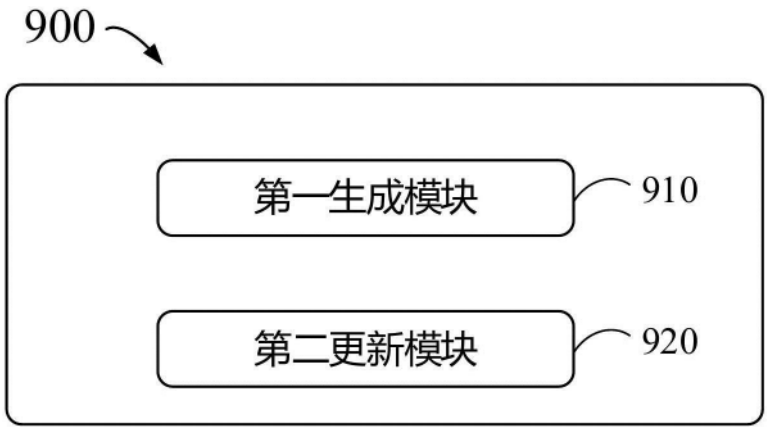


图9

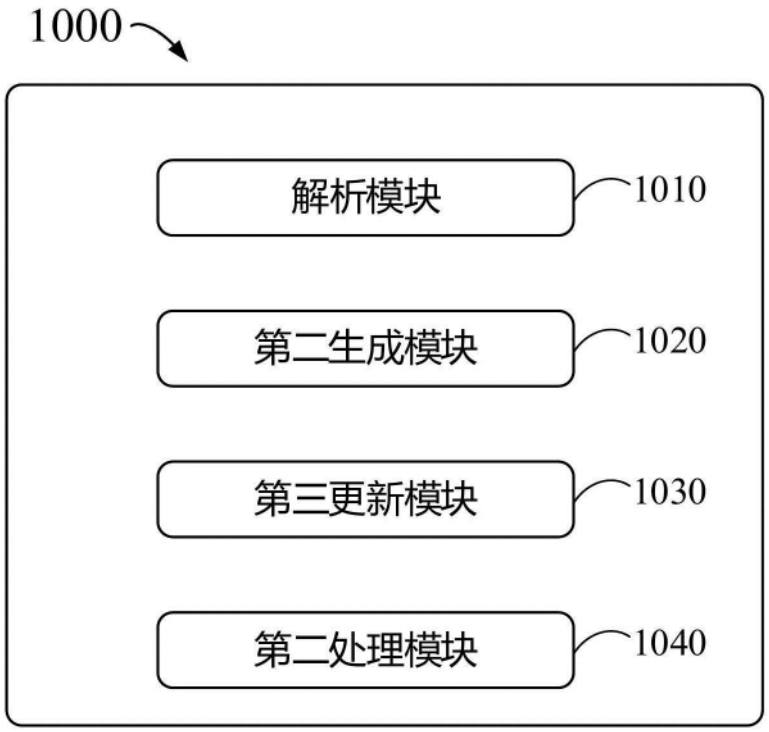

1000
解析模块
1010
第二生成模块
1020
第三更新模块
1030
第二处理模块
1040

图10

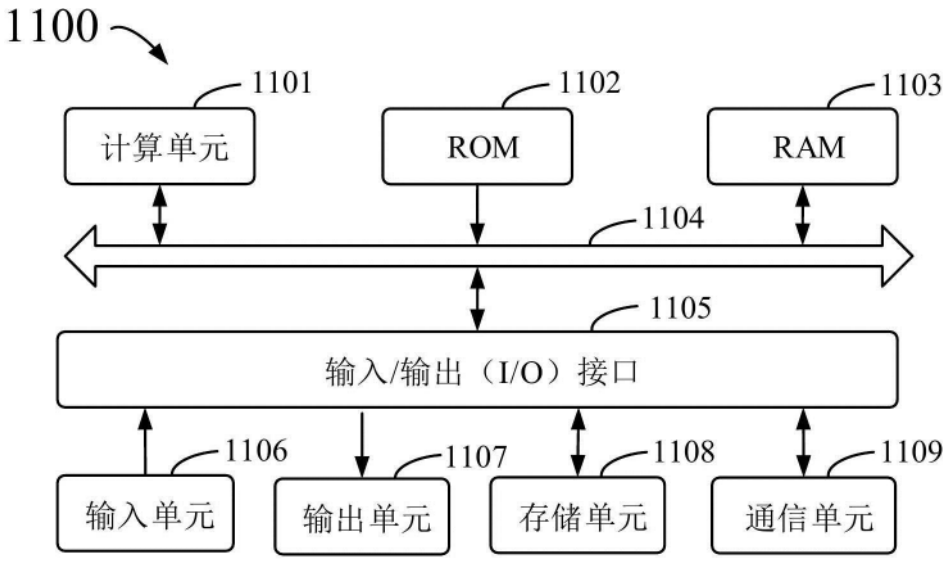

1100
1101
计算单元
1102
ROM
1103
RAM
1104
1105
输入/输出（I/O）接口
1106
输入单元
1107
输出单元
1108
存储单元
1109
通信单元

图11