(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 117909243 A
(43)申请公布日 2024.04.19

(21)申请号 202410120458.4

(22)申请日 2024.01.29

(71)申请人 南京大学
地址 210023 江苏省南京市栖霞区仙林大道163号南京大学

(72)发明人 黄书剑 张我豪 戴新宇 陈家骏

(74)专利代理机构 江苏圣典律师事务所 32237
专利代理师 于瀚文 胡建华

(51)Int.Cl.
G06F 11/36(2006.01)
G06N 5/022(2023.01)
G06N 3/126(2023.01)
G06F 40/205(2020.01)

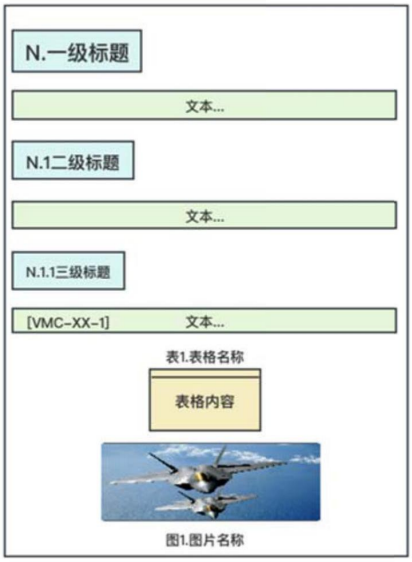
权利要求书4页 说明书16页 附图4页

(54)发明名称

大模型智能体驱动的航空文档分析与测试用例生成系统

(57)摘要

本发明提供了大模型智能体驱动的航空文档分析与测试用例生成系统,包括原生文档知识化处理模块、指令分解模块、指令重用模块、记忆模块、规划模块、校验模块、执行模块、探索模块、本地知识库;所述原生文档知识化处理模块用于将含异质数据的航空数据原生文档进行知识化处理;所述指令分解模块用于,通过指令分解器,将用户输入的初始指令分解为一系列抽象的高层规划步骤或子目标,指导整个系统的运作流程;本发明基于智能决策框架和形式化模型的构建,能够更精确地符合实际需求和条件限制。这种方法与传统的静态决策过程相比,能够更灵活地适应不同设备、不同场景的测试用例生成任务。



CN 117909243 A

1.大模型智能体驱动的航空文档分析与测试用例生成系统,其特征在于,包括原生文档知识化处理模块、指令分解模块、指令重用模块、记忆模块、规划模块、校验模块、执行模块、探索模块、本地知识库;

所述原生文档知识化处理模块用于将含异质数据的航空数据原生文档进行知识化处理;

所述指令分解模块用于,通过指令分解器,将用户输入的初始指令分解为一系列抽象的高层规划步骤或子目标,指导整个系统的运作流程;

所述指令重用模块用于收集经验、从经验中提取宏观规律、推理评估;

所述规划模块用于产生低层计划,负责具体执行时的规划;

所述校验模块用于调整高层规划步骤,反馈错误原因给规划模块,帮助规划模块重新执行当前步骤;

所述执行模块用于提供工具树;

所述记忆模块用于收集每个子智能体类型工具的交互过程作为历史交互记录,为长窗口对话提供条件;还用于指令复用,收集成功执行的工具链交互轨迹,当面临类似的新场景时,将成功执行的工具链交互轨迹作为示例或者根据实际上下文进行调整;

所述记忆模块还用于提供缓存机制,缓存每个工具出的中间结果;使用当前工具位置所在的工具链路径涉及的各个工具名组成的字符串作为一层键,当前工具的输入的哈希值作为二层键,将工具输入作为识别同一工具的不同运行;

所述记忆模块还用于测试用例维护与管理:当生成测试用例时,就将测试用例对应依据、生成过程、组成成分、最终测试用例联系在一起;

所述探索模块用于,根据飞行测试仪器的反馈,如果输入的测试用例使得飞行测试仪器显式本次测试不通过,且生成的测试用例不存在异常。

2.根据权利要求1所述的系统,其特征在于,所述原生文档知识化处理模块用于将航空数据原生文档进行知识化处理,具体包括:

将使用python-docx库对Docx文档进行初次解析并遍历:

对于标题,维护一个元素为各级标题的栈结构,将格式为<文件名>-<一级标题>-<二级标题>..-<N级标题>的元信息关联到文本、图片和表格;

对于文本,按特殊标题为单位收集段落,同一最近邻标题下的各个段落共享相同的元数据,合并为更大的文本块;将文本块按特定的阈值尺寸,以滑动窗口的方式进行切割,使文本块同时满足适合尺寸和语义完整性;使用中文预训练句子编码器将文本块向量化,与元数据一起存入本地知识库;

对于图片,同一最近邻标题下的各个图片也共享相同的元数据,如果是Visio嵌入格式的图片,解压Docx文档获得图片数据,并解析Docx底层XML文件获得图片锚点信息,与最近邻标题进行对齐;选择预训练多模态图文编码器进行编码,预训练多模态图文编码器将图片和文本编码到同一向量空间;或者直接存储图片路径,依靠元数据进行语义匹配或过滤;

对于表格,按照模版将表格序列化为顺序文本。

3.根据权利要求2所述的系统,其特征在于,在本地知识库需要检索时,将查询编码到同一文本向量空间,计算查询向量与候选文本块向量之间的相似性度量,取前K个候选文本块进行召回:进行混合检索,将向量查询和基于关键词的查询进行结合,计算综合得分;通

过元数据过滤限制搜索范围;进行重排校验。

4.根据权利要求3所述的系统,其特征在于,所述指令分解模块将指令分解器的运行过程描述为将初始指令I_0分解为高层计划$I_h=[g_0,g_1,\cdots,g_T]$,其中每个子目标g_i是指通过高层动作得到某种资源或数据集合,i取值为0~T,T是获取所有资源或数据集合的总时间步数;其中高层动作是由下文中介绍的低层规划步骤的集合。

5.根据权利要求4所述的系统,其特征在于,当用户输入查询指令时,所述指令分解模块从航空行业测试者的角度将所述查询指令分解为生成测试用例的宏观步骤,包括:获取待测试通道、测试规则作为初始测试限制信息;使用组合测试工具生成测试用例集;在测试用例集元素上执行模拟连锁故障、故障复位、测试用例组装;提示测试用例生成任务结束。

6.根据权利要求5所述的系统,其特征在于,所述规划模块利用大型语言模型产生底层计划,将每个子目标映射为一系列更为细粒度的计划$I_l=[a_0,a_1,\cdots,a_T]$,其中I_l表示底层指令,a_T表示某时间步下执行的指令动作。

7.根据权利要求6所述的系统,其特征在于,所述工具树的节点包括四种类型:子智能体Sub Agent、管道Pipeline、组件Component、可调用函数Callable Function;

所述工具树的每个节点都设置了缓存记忆,所述执行模块还为指令重用模块提供记忆的成功或失败执行轨迹;

所述工具树同级节点之间存在前后调用顺序限制关系;

当某个子智能体节点输出响应,规划出下一步将要执行的步骤,会在后台解析响应,获取工具名和工具输入,然后运行工具得到运行结果;根据运行结果的性质,采用不同的策略:如果运行结果涉及多跳推理,将运行结果反馈到智能体的上下文窗口中,参与后续规划步骤;如果运行结果不涉及后续推理,或者含有N个以上tokens,将运行结果卸载到记忆模块缓存。

8.根据权利要求7所述的系统,其特征在于,所述本地知识库中存储有将航空系统测试的历史数据和专家知识,所述系统使用航空系统测试的历史数据和专家知识来指导遗传算法。

9.根据权利要求8所述的系统,其特征在于,系统实时监控仿真设备的反馈验证信息,其中仿真设备是用于模拟航空真实运行环境的软硬件设施,具体包括:

初始化遗传算法:生成初步的测试用例集,涵盖所有参数和故障类型的组合;

根据组合测试原则,将已采样的变量参数和待模拟的故障列表进行组合,得到测试用例组成参数,后续经过所述系统的智能形式化建模,得到测试用例组成成分;

执行测试并收集数据:运行测试用例,系统收集飞行测试仪器的反馈以及测试过程中的异常;

分析测试结果:智能体分析测试数据,确定最常被检测到的故障类型,以及更容易被忽略的故障类型、最常导致故障的参数组合;根据飞行测试仪器的反馈,如果输入的测试用例使得飞行测试仪器显式本次测试不通过,且测试用例不存在异常;

调整算法参数:基于智能体的分析,调整遗传算法的适应度函数和算子权重;

迭代和优化:重复测试和分析过程,持续优化遗传算法,直至达符合要求的故障检测率和测试覆盖率。

10.根据权利要求9所述的系统,其特征在于,所述指令重用模块用于收集经验、从经验

中学习并提取宏观经验、推理生成系统未见过的测试用例并评估,具体包括:

步骤a1,收集经验:在收集经验阶段,通过记忆模块收集每个节点的交互过程、中间结果和外部反馈,如果节点是子智能体,还收集节点的上层规划:在最多Z次的时间内不断重试训练任务,首先定义以下符号:

$\tau_{n,z}$:表示第n个任务的第z次尝试的执行轨迹;

F_{examp}:与$\tau_{n,z}$相关的Few-shot示例;

$V_{n,z}$:表示第n个任务的第z次及之前所有尝试获取的所有反馈;

在接下来的过程中,智能体会经历如下状态:

初始状态:智能体尝试使用与第n个任务初次尝试的执行轨迹$\tau_{n,0}$相关的Few-shot示例作为上下文来执行任务,使用ReAct作为大型语言模型的基本规划算法,将执行轨迹$\tau_{n,0}$、轨迹示例F_{examp}、初始反馈$V_{n,0}$进行拼接,作为模型的输入:

$LLMReAct(\cdot|\tau_{n,0},F_{examp},V_{n,0})$

在第z次实验中,当智能体完成任务或达到最大步数时,记忆模块收集轨迹$\tau_{n,z}$;如果智能体成功完成任务,则执行下一个任务;z的取值范围是0到Z;

如果智能体失败,让智能体回顾失败的轨迹并进行自我反思,产生当前步的反馈,并与之前的反馈拼接进行结合:

$V_{n,z+1}=concat(V_{n,z},LLMreflect(\tau_{n,z}))$

其中concat表示拼接;

在下一次重试任务时,智能体会使用反馈$V_{n,z+1}$来增强上下文,此时上下文是:

$LLMReAct(\cdot|\tau_{n,z+1},F_{examp},V_{n,z+1})$

步骤a2,从经验中学习并提取宏观经验:

从经验中学习并提取宏观经验类似非策略学习,其中智能体从行为策略的经验中学习;

给大型语言模型智能体的指令I分解为任务说明和Few-shot示例:

类似的经验作为示例:在这个过程中,系统将从记忆模块中提取成功的案例轨迹作为示例;

从成功和失败中学习:系统通过不断从成功或失败轨迹中总结出航空领域中普遍适用的、具有泛化性质的抽象规律和原则,构建宏观经验集合;

具体操作:为模型提供操作符,维护一个宏观经验集合,宏观经验集合初始为空集,迭代地为模型提供失败和成功轨迹对,或者提供从记忆模块中检索的L个成功轨迹;

LLM执行的操作是:ADD添加一个新的见解,EDIT编辑现有见解的内容,DOWNVOTE否决不同意现有见解;UPVOTE赞成票与现有见解一致;

新添加的见解初始将被赋予两个重要性计数,并且如果随后应用了UPVOTE或EDIT操作,计数将会增加;如果应用了DOWNVOTE操作,计数则会减少;如果某个见解的重要性计数降到零,将会被移除;

步骤a3,在推理评估阶段,所述系统尝试生成一系列未曾见过的测试用例:

提取步骤a1和a2收集的见解和成功轨迹,并建立成功轨迹的向量知识库,对于每项任务,任务描述由提取的完整见解列表串联来扩充:$l\hat{}=concat(l_1,l_2..l_n)$,其中$l\hat{}$是串联后的完整见解列表,$l_n$为第n个见解;然后检索具有最高任务相似性的前k个轨迹作为Few-

shot上下文示例F_{retrie_examp};最后将完整任务见解1ˆ和上下文示例F_{retrie_examp}填入模板,输入大型语言模型进行任务尝试。

大模型智能体驱动的航空文档分析与测试用例生成系统

技术领域

[0001] 本发明属于人工智能与数据处理技术领域,尤其涉及大模型智能体驱动的航空文档分析与测试用例生成系统。

背景技术

[0002] 根据资料显示,目前国内外很多研究机构对航空软件测试标准都有专门的研究。2004年我国发布了GJB/Z-2004《军用软件测试指南》,对国内有关航空软件测试提出了一套正式的官方标准。国外在1982年发布了一套标准,在修改后于1992年发布了DO-178B标准,名为《机载系统和设备合格审定中的软件考虑》。DO-178标准为航空软件的设计、开发、测试以及配置提供指导。在该标准中,与测试相关的理论主要是用验证定义软件测试,即验证包括评审、分析和测试三个部分。

[0003] 评审主要分析软件系统是否满足设计需求说明书所规定的各项功能指标,提供定性的分析;分析是指对一个软件系统的功能、性能和安全性影响等内容进行分析,提供正确的可重复证据;测试一般就是指通过构造测试用例后运行被测软件程序以发现故障缺陷,除此之外,需要补充的有两点:第一点是证明软件满足它的需求,第二点是证明系统安全评估时审定的那些可能导致严重问题的缺陷已经被移除。

[0004] 目前同类测试用例自动化技术主要面临的挑战是:一是软件需求规格说明复杂,形式化需求困难;二是要求测试用例对于软件的覆盖度尽可能高,去除冗余测试用例,保证测试用例的类型和数量;三是尽可能降低人力成本,提高测试用例生成方法的自动化和泛用性。最终目的是通过自动化测试发现航空软件系统中存在的缺陷。

[0005] 关于形式化建模,最近工作使用XYZ/E这种语言将边界值产生的测试数据进行时序逻辑的描述用以生成测试用例,将软件规范和语言语义有效结合起来,但在实际使用中,XYZ/E这种语言不仅鲜有人用,而且难以在短时间内理解。

[0006] 曹爽使用基于Kripke结构的时序模型的有限状态机来建模,同样存在需要人工分析软件需求规格说明书或仿真设备技术文档,然后手动建模,存在人力成本较高、迁移性不强的问题。张伟提出的自动生成测试用例框架在对输入参数添加约束和对生成的测试用例添加期望时,都需要手工去操作;并且对软件需求文档进行自动化分析时使用的是正则表达式,这限制了文档处理的灵活性,适用性不强。王文轩在论文中提出了“四变量理论模型”的形式化方法,构建包含事件、条件、模式、环境相互作用的环境,需要人工构建整个过程,自动化程度不高。

[0007] 关于测试用例算法改进,主要目的是去除冗余测试用例,保证测试用例的类型和数量。郑超群针对其局部收敛和早熟现象,优化了适应度函数和交叉变异算法。这种方式在进化后期仍能保留一定的交叉与变异能力,与标准遗传算法相比收敛速度更快。高雪笛采用模拟退火的方法对选择、变异、遗传等算法进行优化改进。新个体的接受概率由Metropolis准则函数得出,这样就能达到在搜索后期仍具有全局最优性。他们的工作都主要集中在改进测试用例算法本身,并没有关注测试用例执行结果带来的持续反馈,没有强

化测试用例算法的针对性搜索功能。

发明内容

[0008] 发明目的:针对软件需求规格或仿真设备技术文档说明含异质数据、说明复杂导致需要耗费大量人力和时间进行形式化建模的问题,本发明提供了一种大模型智能体驱动的文档分析与测试用例生成系统,这里的智能体是指大模型在特定指令下,模仿人类的决策过程来充当核心协调者,它使用自然语言处理、检索、多模态问答等技术提高了测试用例生成的自动化程度;针对以往测试用例建模场景难以泛化的问题,本发明提供了一种将新测试用例任务的自然语言指令,借助已执行轨迹作为测试用例生成示例并提取通用测试经验,自主转换为新的高低层规划的方法,该方法对节点集合进行排列组合,快速变换到不同测试场景的有限状态图;针对传统测试用例生成算法没有强化针对性搜索的问题,本发明提供了在遗传算法生成过程中根据反馈实时调整算子权重的方法。

[0009] 本发明系统包括原生文档知识化处理模块、指令分解模块、指令重用模块、记忆模块、规划模块、校验模块、执行模块、探索模块、本地知识库;

[0010] 所述原生文档知识化处理模块用于将含异质数据(表格、文本、图片)的航空数据原生文档进行知识化处理;

[0011] 所述指令分解模块用于,通过指令分解器,将用户输入的初始指令分解为一系列抽象的高层规划步骤或子目标,指导整个系统的运作流程。这些高层步骤与常识推理密切相关,例如在航空测试用例场景涉及的常识是:“发送数据”之前通常需要“读取数据”,然后是“处理数据”;“选择表格中的行列单元格”之前需要需要“检索到相关表格”。目前大型语言模型已经被证明掌握了大量的常识性知识,且已有工作利用大型语言模型作为零样本规划器,这意味着它们可能具备自主规划的能力,能够在最小的人为干预下,有效地担任智能体系统中的高级规划者。指令分解模块利用思维链、上下文学习、实时反馈等技巧来有效地激活和利用这种自主规划能力。

[0012] 所述指令重用模块用于收集经验、从经验中提取宏观规律、推理评估;

[0013] 所述规划模块用于产生低层计划,负责具体执行时的规划;

[0014] 所述校验模块用于调整高层规划步骤,反馈错误原因给规划模块,帮助规划模块重新执行当前步骤;

[0015] 所述执行模块用于提供工具树;

[0016] 所述记忆模块用于收集每个子智能体(Sub Agent)类型工具的交互过程作为历史交互记录,为长窗口对话提供条件;还用于指令复用,收集成功执行的工具链交互轨迹,当面临类似的新场景时,将成功执行的工具链交互轨迹作为示例或者根据实际上下文进行调整;所述记忆模块收集的交互过程也是实际的运行场景,是满足Agent tuning训练数据格式的天然标记数据;

[0017] 所述记忆模块还用于提供缓存机制,缓存每个工具出的中间结果;使用当前工具位置所在的工具链路径涉及的各个工具名组成的字符串作为一层键,当前工具的输入的哈希值作为二层键,将工具输入作为识别同一工具的不同运行;

[0018] 所述记忆模块还用于测试用例维护与管理:当生成测试用例时,就将测试用例对应依据、生成过程、组成成分、最终测试用例联系在一起;

[0019] 所述探索模块用于,根据飞行测试仪器的反馈,如果输入的某测试用例使得飞行测试仪器显式本次测试不通过,且本发明生成的测试用例不存在异常,那么本测试用例中的组成成分,例如已采样的变量参数和模拟的故障列表是待搜索的重点方向,通过在遗传算法等现有搜索算法中赋予这些变量参数和故障列表更大的权重,这样的实时反馈改进了遗传算法容易陷入局部最优,从而缓解了遗传算法容易忽略实际故障的问题。

[0020] 所述原生文档知识化处理模块用于将航空数据原生文档进行知识化处理,具体包括:

[0021] 将使用python-docx库对Docx文档进行初次解析并遍历:

[0022] 对于标题,维护一个元素为各级标题的栈结构,将格式为<文件名>-<一级标题>-<二级标题>..-<N级标题>的元信息关联到文本、图片和表格;

[0023] 对于文本,按特殊标题为单位收集段落,同一最近邻标题下的各个段落共享相同的元数据,合并为更大的文本块;将文本块按特定的阈值尺寸(例如512,通常按照模型上下文窗口大小以及实际效果为准),以滑动窗口的方式进行切割,使文本块同时满足适合尺寸和语义完整性;使用中文预训练句子编码器将文本块向量化,与元数据一起存入本地知识库;

[0024] 对于图片,同一最近邻标题下的各个图片也共享相同的元数据,如果是Visio嵌入格式的图片,解压Docx文档获得图片数据,并解析Docx底层XML文件获得图片锚点信息,与最近邻标题进行对齐;选择预训练多模态图文编码器进行编码,预训练多模态图文编码器将图片和文本编码到同一向量空间;或者直接存储图片路径,依靠元数据进行语义匹配或过滤;

[0025] 对于表格,按照模版将表格序列化为顺序文本。

[0026] 在本地知识库需要检索时,将查询编码到同一文本向量空间,计算查询向量与候选文本块向量之间的相似性度量,取前K(例如2、4、8…)个候选文本块进行召回:进行混合检索,将向量查询和基于关键词的查询进行结合,计算综合得分;通过元数据过滤限制搜索范围;进行重排校验。

[0027] 所述指令分解模块将指令分解器的运行过程描述为将初始指令I_0分解为高层计划$I_h=[g_0,g_1,\cdots,g_T]$,其中每个子目标g_i是指通过高层动作得到某种资源或数据集合,i取值为0~T,T是获取所有资源或数据集合的总时间步数;其中高层动作是由下文中介绍的低层规划步骤的集合。

[0028] 当用户输入查询指令时,所述指令分解模块从航空行业测试者的角度将所述查询指令分解为生成测试用例的宏观步骤,包括:获取待测试通道、测试规则作为初始测试限制信息;使用组合测试工具生成测试用例集;在测试用例集元素上执行模拟连锁故障、故障复位、测试用例组装;提示测试用例生成任务结束。

[0029] 所述规划模块利用大型语言模型产生底层计划,将每个子目标映射为一系列更为细粒度的计划$I_l=[a_0,a_1,\cdots,a_T]$,其中I_l表示底层指令(lower instruction),a_T表示某时间步下执行的指令动作。这些计划不一定都是原子的,在下文的工具树部分,会介绍每个节点都会有产生高层计划的指令分解模块和产生低层规划的规划模块,经过层层调用,最终到达原子操作。

[0030] 所述工具树的节点包括四种类型:子智能体Sub Agent、管道Pipeline、组件

Component、可调用函数Callable Function;工具树中的叶结点一般为组件或可调用函数,是不可再分的元工具,通常对应了待组装的测试用例组成成分;工具树中的中间节点一般为子智能体或管道,其中子智能体能够够根据反馈自动修改或人工修改节点的高层规划,使其重新选择、实例化和排列组合其成员工具,此时工具树中一部分是原有节点,一部分则是信实例化的新节点。管道是包装常用的流水线过程,也能够实时排列组合其成员组件,但一般不具备修改规划的能力。四种类型的节点使得整个系统能够快速变换模拟对象或场景;

[0031] 所述工具树的每个节点(上述四种类型的节点)都设置了缓存记忆,这除了能够减少重复操作消耗的时间和计算资源外,所述执行模块还为指令重用模块提供记忆的成功或失败执行轨迹;

[0032] 所述工具树同级节点之间存在前后调用顺序限制关系;

[0033] 当某个子智能体节点输出响应,规划出下一步将要执行的步骤,会在后台解析响应,获取工具名和工具输入,然后运行工具得到运行结果;根据运行结果的性质,采用不同的策略:如果运行结果涉及多跳推理,将运行结果反馈到智能体的上下文窗口中,参与后续规划步骤;如果运行结果不涉及后续推理,或者含有N(一般取值为40)个以上tokens,将运行结果卸载到记忆模块缓存。

[0034] 所述本地知识库中存储有将航空系统测试的历史数据和专家知识,所述系统使用航空系统测试的历史数据和专家知识来指导遗传算法。

[0035] 系统实时监控仿真设备的反馈验证信息,其中仿真设备是用于模拟航空真实运行环境的软硬件设施,具体包括:

[0036] 仿真软件解释:仿真软件实现的算法模型是用于模拟现实世界事件、系统或过程的数学模型和计算方法的集合。这些模型的目的是在计算机环境中重现或预测复杂现象的行为。仿真设备解释:物理设备或硬件系统,用于实现仿真过程。仿真设备可以包括计算机硬件、专用仿真机器(如飞行模拟器)、传感器、执行器等。

[0037] 两者关系:仿真算法模型通常需要运行在仿真设备上。设备提供了必要的计算能力和用户界面,让模型能够被有效地执行和交互。仿真设备为算法模型提供输入(如传感器数据)并实现输出的展示(如视觉显示、物理反馈)。同时,算法模型指导设备的操作,模拟出现实世界中的行为和现象。

[0038] 所述系统主要是黑盒测试,不涉及仿真设备内部的信息。所述系统主要通过形式化技术文档,并根据文档信息建模数值之间的依赖和限制关系,输出一些特定信息,用于组成测试用例,然后输入到仿真设备中。

[0039] 初始化遗传算法:生成初步的测试用例集,涵盖所有参数和故障类型的组合;

[0040] 测试用例的组成成分中有物理量、预期值等参数,这些参数可能是整型变量、浮点型变量、或数字型变量,需要按照一定的算法(例如上面介绍的遗传算法)来对这些变量进行采样。故障类型是指待模拟的设备可能出现的故障种类,例如通信心跳故障,根据航空测试技术文档,心跳故障会引发设备故障、通信故障和信号故障。

[0041] 根据组合测试(常用的测试方法)原则,将已采样的变量参数和待模拟的故障列表进行组合,得到测试用例组成参数,后续经过所述系统的智能形式化建模,得到测试用例组成成分。

[0042] 执行测试并收集数据:运行测试用例,系统收集飞行测试仪器的反馈以及测试过程中的异常;(在航空仿真设备中,被测部件向飞行管理计算机发送模拟信号,然后飞行管理计算机发送预期接收的信号给飞行测试仪器,由飞行测试仪器判断本次测试是否通过);

[0043] 分析测试结果:智能体分析测试数据,确定最常被检测到的故障类型,以及更容易被忽略的故障类型、最常导致故障的参数组合等;根据飞行测试仪器的反馈,如果输入的某测试用例使得飞行测试仪器显式本次测试不通过,且本发明的测试用例不存在异常,那么本测试用例中的组成成分,例如已采样的变量参数和模拟的故障列表是待搜索的重点方向,通过在遗传算法等现有搜索算法中赋予这些变量参数和故障列表更大的权重,这样的实时反馈改进了遗传算法容易陷入局部最优,从而缓解了遗传算法容易忽略实际故障的问题。

[0044] 调整算法参数:基于智能体的分析,调整遗传算法的适应度函数和算子权重;

[0045] 迭代和优化:重复测试和分析过程,持续优化遗传算法,直至达符合要求的故障检测率和测试覆盖率。

[0046] 所述指令重用模块用于收集经验、从经验中学习并提取宏观经验、推理生成系统未见过的测试用例并评估,具体包括:

[0047] 步骤a1,收集经验:在收集经验阶段,通过记忆模块收集每个节点的交互过程、中间结果和外部反馈,如果节点是子智能体,还收集节点的上层规划:在最多Z次(Z取值为8~16)的时间内不断重试“训练”任务,首先定义以下符号:

[0048] $\tau_{n,z}$:表示第n个任务的第z次尝试的执行轨迹;

[0049] F_{examp}:与$\tau_{n,z}$相关的Few-shot示例;

[0050] $V_{n,z}$:表示第n个任务的第z次及之前所有尝试获取的所有反馈(最初$V_{n,0}$是空字符串);

[0051] 在接下来的过程中,智能体会经历如下状态:

[0052] 初始状态:智能体尝试使用与第n个任务初次尝试的执行轨迹$\tau_{n,0}$相关的Few-shot示例作为上下文来执行任务,使用ReAct作为大型语言模型(LLMs)的基本规划算法,将执行轨迹$\tau_{n,0}$、轨迹示例F_{examp}、初始反馈$V_{n,0}$进行拼接,作为模型的输入:

[0053] $LLMReAct(\bullet|\tau_{n,0},F_{examp},V_{n,0})$

[0054] 在第z次实验中,当智能体完成任务或达到最大步数时,记忆模块收集轨迹$\tau_{n,z}$;如果智能体成功完成任务,则执行下一个任务;z的取值范围是0到Z;

[0055] 如果智能体失败,让智能体回顾失败的轨迹并进行自我反思,产生当前步的反馈,并与之前的反馈拼接进行结合:

[0056] $V_{n,z+1}=concat(V_{n,z},LLMreflect(\tau_{n,z}))$

[0057] 其中concat表示拼接;

[0058] 在下一次重试任务时,智能体会使用反馈$V_{n,z+1}$来增强上下文,此时上下文是:

[0059] $LLMReAct(\bullet|\tau_{n,z+1},F_{examp},V_{n,z+1})$

[0060] 步骤a2,从经验中学习并提取宏观经验:

[0061] 从经验中学习并提取宏观经验类似非策略学习(off policy learning,Q-learning),其中智能体可从行为策略的经验中学习。

[0062] 给大型语言模型(Large Language Model)智能体的指令I分解为任务说明和Few-

shot示例：

[0063] 类似的经验作为示例：在这个过程中，系统将从记忆模块中提取成功的案例轨迹作为示例。具体来说，它会根据文本相似度，挑选出与待评估任务最为匹配的前k个成功轨迹，其中文本相似度是通过计算内积得出的。

[0064] 从成功和失败中学习：系统通过不断从成功或失败轨迹中总结出航空领域中普遍适用的、具有泛化性质的抽象规律和原则，构建“宏观经验”集合。该集合还为使用者提供了一个接口，可以将人脑记忆中的测试经验(通常不会记录在书面文档)注入到系统智能体的执行上下文中，实时提供一些敏感而保密的信息，以帮助生成测试用例，同时确保这些信息不会泄露；

[0065] 具体操作：为模型提供几个操作符，维护一个“宏观经验”集合，“宏观经验”集合初始为空集。迭代地为模型提供失败和成功轨迹对，或者提供从记忆模块中检索的L个成功轨迹(L的取值范围是2到8)。这里的意思是：要么提供成功和失败轨迹对，要么全部都是成功轨迹；

[0066] LLM可以执行的操作是：ADD(添加)一个新的见解，EDIT(编辑)现有见解的内容，DOWNVOTE(否决)不同意现有见解；UPVOTE(赞成票)与现有见解一致。

[0067] 新添加的见解初始将被赋予两个重要性计数，并且如果随后应用了UPVOTE或EDIT操作，这个计数将会增加；如果应用了DOWNVOTE操作，计数则会减少。如果某个见解的重要性计数降到零，它将会被移除；

[0068] 步骤a3，在推理评估阶段，所述系统尝试生成一系列未曾见过的测试用例。这些测试用例专门针对相同设备在不同测试场景下的应用，如多余度通信和多余度投票等场景。此外，所述系统还包括针对不同型号设备生成测试用例的功能。

[0069] 提取步骤a1和a2收集的见解和成功轨迹，并建立成功轨迹的向量知识库，对于每项任务，任务描述由提取的完整见解列表串联来扩充：$l\hat{}=\mathrm{concat}(l_1,l_2..l_n)$，其中$l\hat{}$是串联后的完整见解列表，$l_n$为第n个见解；当见解过多时，通过检索技术缓解上下文窗口压力。然后检索具有最高任务相似性的前k个轨迹作为Few-shot上下文示例F_{retrie_examp}；最后将完整任务见解$l\hat{}$和上下文示例F_{retrie_examp}填入模板，输入大型语言模型进行任务尝试；

[0070] 从某一型号部件的有限测试场景中所生成的测试用例，推广到该型号其他测试场景、不同型号的同类部件，甚至是不同的部件。使用从源测试场景中提取的宏观见解或使用人员实时注入的宏观见解和从目标测试场景提取较少(例如1到4条)执行轨迹示例来适配宏观见解。

[0071] 本发明具有如下有益效果：

[0072] 技术层面优点：

[0073] 1.航空数据原生文档知识化：通过自适应解析含异质数据的航空数据原生文档，如表格、文本、图片，并建立索引知识库，此技术能够提高数据处理的准确性和效率。传统测试方法通常无法高效地处理含混合数据类型的需求/技术文档，而这种方法则为复杂数据的分析和利用提供了新的途径。

[0074] 2.智能决策与形式化建模：基于智能决策框架和形式化模型的构建，能够更精确地符合实际需求和条件限制。这种方法与传统的静态决策过程相比，能够更灵活地适应不同设备、不同场景的测试用例生成任务。

[0075] 3.测试用例生成算法的自适应:根据反馈实时调整算法算子权重和测试路径,不仅提高了测试用例的质量和覆盖率,还增强了测试过程的灵活性和效率。

[0076] 4.持续进化:通过智能体模块之间的互动以及与人类的交互,系统能够持续学习成功或失败执行轨迹,并提取通用测试经验,提升其长期的适应性和性能。

[0077] 应用层面效果:

[0078] 1.提升测试效率和质量:本发明的智能测试用例系统可以快速生成和适应不同的测试场景,显著提高软件测试的效率和覆盖率,从而确保软件质量。

[0079] 2.降低人工成本和错误率:本发明的智能测试用例系统相比以往的系统,有着更高自动化和智能化程度,减少了对专业人员的依赖,降低了由于人为因素造成的错误。同时大模型通过自主调用中间工具,极大解决了大模型容易出现幻觉、输出片段化的问题,使得测试用例生成过程更加严谨。同时该系统具备长期学习能力,不会因知识过期导致无法使用。

[0080] 3.操作门槛低:以往的自动化程序工具大多都具有复杂的操作过程且不易理解,而本发明的方法通过自然语言指令来控制系统的运作过程,适合大多数操作者。

[0081] 综上所述,本发明在技术创新和应用实践上都展现出显著的优势,为航空软件系统的自动测试领域带来了新的发展机遇。

附图说明

[0082] 下面结合附图和具体实施方式对本发明做更进一步的具体说明,本发明的上述和/或其他方面的优点将会变得更加清楚。

[0083] 图1是Docx文档示意图。

[0084] 图2是航空数据原生文档知识化过程示意图。

[0085] 图3是智能测试用例生成系统架构图。

[0086] 图4是获取初始指令示意图。

[0087] 图5是指令分解示意图。

[0088] 图6是工具/子智能体树示意图。

[0089] 图7是测试用例生成任务提示模板示意图。

[0090] 图8是传统测试用例自动生成框图。

具体实施方式

[0091] 本发明提供了大模型智能体驱动的航空文档分析与测试用例生成系统,包括:

[0092] 航空数据原生文档知识化

[0093] 在本发明中,进行智能测试用例生成之前需要进行航空数据原生文档(与沈阳飞机设计研究所合作,获取航空数据原生文档)知识化处理,方便系统提取测试用例生成的相关信息。

[0094] 本发明在这部分使用的方法主要涉及文档自适应解析、自然语言向量化、检索技术和多模态图文编码技术。本发明参考了Langchain关于文本的切割方法,且与Langchain更偏向于处理纯文本(例如HTML、Markdwon、代码等)相比,本发明的处理方法更偏向于含异质数据的文档,因此在这里以富含表格、文本、图片等异质数据的Docx文档为例。如图1所

示。

[0095] 展示航空数据原生文档知识化的过程,如图2所示。

[0096] 首先,将使用python-docx库对Docx文档进行初次解析并遍历,在这过程中,需要注意四种对象:标题、文本、图片、表格。

[0097] 对于标题,会维护一个元素为各级标题的栈结构,仿照人类阅读习惯,将<文件名>-<一级标题>-<二级标题>..-<N级标题>这样的元信息关联到文本、图片和表格。在航空技术文档中可能出现特殊标题,这对应了一条测试规则,因此还可以通过提供类似模式的字段进行正则表达式编译,将特殊标题也加入栈进行维护。这样做的目的有:在检索时通过元数据过滤可以精准地限制搜索范围;在创建知识库时,可以将元数据拼接到文本开头,丰富上下文语义,提高搜索准确率;能够溯源生成测试用例组件所使用的文本材料证据支持。通过对元数据进行向量语义匹配,也可以做到在交互中灵活访问海量文档中特定章节的文本、表格、图片。

[0098] 对于文本,为了保证技术文档中的规则描述文本的完整性,按特殊标题为单位收集段落,同一最近邻标题下的各个段落共享相同的元数据,可合并为更大的文本块。为了能够发挥更好的检索性能,将文本块按特定的阈值尺寸以滑动窗口的方式进行切割,使文本块同时满足适合尺寸和语义完整性。之后使用中文预训练句子编码器将文本块向量化,与元数据一起存入知识库。

[0099] 对于图片,同一最近邻标题下的各个图片也共享相同的元数据。值得注意的是,如果是Visio嵌入格式的图片,需要解压Docx文档获得图片数据,并解析Docx底层XML文件获得图片锚点信息,与最近邻标题进行对齐。之后选择预训练多模态图文编码器进行编码,该编码器可将图片和文本编码到同一向量空间。还可以直接存储图片路径,依靠元数据进行语义匹配或过滤。

[0100] 对于表格,会按照某一模版将表格序列化为顺序文本,尽可能保证表格内容完整和结构不变。之后的处理流程和普通段落文本类似。

[0101] 在知识库检索阶段,将查询编码到同一文本向量空间,计算查询向量与候选文本块向量之间的相似性度量,取前K个候选文本块进行召回。为了提高检索准确率,有三种选择:一是可以进行混合检索,将向量查询和基于关键词的查询进行结合,计算综合得分;二是通过元数据过滤可以精准地限制搜索范围;三是进行重排校验。

[0102] 智能测试用例生成系统框架

[0103] 如果将大模型权重比作CPU,是硬件,那么一个设计良好的智能体框架就是操作系统,是软件,它能够帮助大模型激发出超常的能力,指引大模型完成未见过的任务,并帮助大模型完成长期记忆等工作,甚至可能养成熟能生巧的进化能力。因此通过自然语言指令及相关编程实现智能体框架的过程也被称为新型的编程。下面将介绍本发明设计的智能测试用例生成系统。

[0104] 智能测试用例生成系统中的控制流和测试用例相关源头、中间信息都是自然语言形式的,如图3所示。用户输入查询“请生成xxx部件xx故障测试用例”开始,系统最终输出测试用例到记忆模块进行测试用例管理,其中涉及了几个关键模块:指令分解模块、指令重用模块、记忆模块、规划模块、校验模块、执行模块、探索模块。

[0105] 如图3所示,首先,当用户输入查询“请生成xxx部件xx故障测试用例”时,将会转换

为初始指令,如图4所示,该指令从一个行业测试者的角度提出了生成测试用例的宏观步骤,包括:获取待测试通道、测试规则作为初始测试数据;使用组合测试工具生成测试用例集;在测试用例集元素上执行模拟连锁故障、故障复位、测试用例组装;提示测试用例生成任务结束。在系统中设置了默认几种测试用例的初始指令,但本发明如同Robert Feldt等人那样鼓励用户自己编写宏观指令,他们在论文中提出会话测试代理中,由人类、工具中间件、增强了记忆和规划能力的大模型共同驱动软件测试系统,其中人类则只需要提供高级指令,而大模型则启动整个测试过程。从这个角度来看,这使得人类的测试压力大大减少。

[0106] 交互方式

[0107] 从图4可以看到,本发明设计的智能测试生成系统是由聊天的方式进行的,这是出于对实验结果的观察,发现当在ReAct模板(即每个步骤包含“思考”、“工具”、“工具输入”,等待工具返回结果)加入含有多个步骤的完整示例时,不同的模型都可能呈现出不稳定的现象:正常情况模型一次交互必须仅输出一个步骤来表示下一步计划,等待外部工具调用结果反馈后再响应;而模型有时会在一次交互中输出多个步骤以及示例中的外部工具调用结果反馈,导致上下文混乱,无法在真实测试环境中取得外部工具调用结果反馈。猜测这可能是ReAct模板并没有标注用户和AI助手的角色,导致解决复杂任务的多步骤示例看起来像是具有某种格式的普通文本,而非交互步骤,以至于让模型执行了类似续写的功能,而不是交互式助手。另外,当前发布的具有指令遵循能力的大模型均是chat版本的模型,更适合聊天交互式的智能体交互方式。复杂交互任务定义:在每个时间步长$t\in\{0,\cdots,T\}$(其中T为总时间步长),智能体接收到一个观测$o\in O$(其中O为可观测信息集合),并且根据其观测历史H_t,决定执行动作$a\in A$(其中A为可采取的动作集合)。智能体的目标是实现一些目标$g\in G$(其中G为实现最终目标所必须实现的子目标集合)。

[0108] 指令分解模块

[0109] 当得到偏口语化的初始指令后,会通过指令分解器将其分解为一系列可操作的高层规划步骤,指导整个系统的运作流程,如图5所示。

[0110] 直觉上,这些高层步骤与常识推理密切相关,例如在本发明的场景中,涉及的常识是:“发送数据”之前通常需要“读取数据”,然后是“处理数据”;“选择表格中的行列单元格”之前需要需要“检索到相关表格”。目前大型语言模型已经被证明掌握了大量的常识性知识,那么其中就有可能包括在本发明场景下所需的常识。而之前的一些工作也已经表明具有指令遵循能力的大型语言模型,例如InstructGPT和Codex等模型被用作零样本规划器来为各种各样的任务生成子目标序列,这意味着它们可能具备自主规划的能力,能够在最小的人为干预下,有效地担任智能体系统中的高级规划者。因此,本发明的任务是利用思维链、上下文学习、实时反馈等技巧来有效地激活和利用这种自主规划能力。

[0111] 形式上,本发明将指令分解器的运行过程描述为将初始指令I_0分解为高层计划$I_h=[g_0,g_1,\cdots,g_T]$,其中每个子目标g_i是指通过高层动作得到某种资源或数据。其中高层动作是由下文中介绍的低层规划步骤的集合。

[0112] 在高层规划步骤的设计上,本发明使用相对一致的格式使用思维链来调用工具、组织整合待输入工具的参数(例如文件、标题、上一步步骤处理结果等),其中分别使用不同的特殊符号来表示实参和形参,其中统一的格式可充分发挥大模型的逐步骤规划能力。之前的工作表明,在上下文中使用与当前手头任务相似的示例可以获得更好的性能。此外,当

涉及到一个新的情况时,人类也会从记忆中回忆起他们在尝试解决任务时解决的类似任务作为参考。受这些观察的启发,本发明通常会以(初始指令,高层规划步骤集合)的配对格式,检索训练集中一些具有代表性和多样性的示例,在few-shot设置指导大模型自主规划,以增强大型语言模型生成高层规划的健壮性和准确性。

[0113] 但值得注意的是,这里的高层规划步骤并不是一成不变的,通过后面将要介绍的校验反馈模块来实时调整它们,也就是说,当校验模块反馈了调整错误步骤的指令给低级规划模块,指导其重新执行当前步骤,期望能够在加入新的提示上下文后能够避开之前的误区从而生成正确答案。但如果该低级规划步骤在若干轮反馈后仍无法得到合法的工具输出,可能是高层计划出现了失误,因为需要根据反馈来实时修改高层计划,并重新组织并实例化工具树中的部分节点。在仍然无法正常工作时,可能终结整个高层计划,并向用户反馈可能需要调整初始指令。

[0114] 规划模块

[0115] 上文中指令分解模块产生的是高层计划,而规划模块产生的是低层计划。

[0116] 形式上,低层计划通常将每个子目标映射为一系列更为细粒度的计划$I_l=[a_0,a_1,\cdots,a_T]$,其中I_l表示底层指令(lower instruction),a_T表示某时间步下执行的指令动作。这些计划不一定都是原子的,在下文的工具树部分,会介绍每个智能体类别的节点都会有产生高层计划的指令分解模块和产生低层规划的规划模块,经过层层调用,最终到达原子操作,这对应了测试用例的组成成分。

[0117] 在实现上,本发明通过构造逻辑连贯且一致的思维链引导模型遵循当前的高层规划,并出于与指令分解器类似的动机,在上下文中加入最相关且具有多样性的few-shot示例,令模型提取上下文信息并自主决策生成下一步骤适合调用的工具以及所需的参数。

[0118] 高层计划和底层计划两者的区别在于前者是指示系统运作的整体流程,并不涉及具体执行时的规划,是宏观层面的;而后者是负责具体执行时的规划,解决的是下一步应该做什么的问题,是具体层面的。将这两者进行分离开的好处有二:一是可以避免上下文语义的混乱。当具体逐步骤解决问题时,会产生一系列相互依赖的中间变量,类似有向无环图,这对大模型的理解能力是压力和挑战,极易造成大模型混淆步骤或理解歧义,难以准确捕捉变量;二是可以提升系统规划能力的灵活性和泛化性,如果直接将宏观计划和具体实施计划混合在一起解决某一任务,那么将很难将该计划运用到其他任务上,因为涉及了特定任务的具体变量。而两者分离则提高了规划能力的灵活性和泛化性,让某一任务的成功宏观计划指令的复用成为可能。

[0119] 执行模块

[0120] 执行模块包括一切可获取的外部工具,例如函数API、Python执行环境、浏览器、航空领域词汇字典等;自定义的测试用例工具,例如迭代执行器,在传入数据上迭代执行指定的工具链;多模态问答器,使用开源大模型,能够通过自然语言指令获取图片中蕴含的信息;字段解释器,以航空软件测试专家的视角解析表格中的各个字段;封装常用执行过程的子智能体,例如进退维护智能体,几乎每个测试用例在开启测试之前都会设置部件的初始状态。图8是传统测试用例自动生成框图。

[0121] 在实现上,本发明设计了四种类型的工具:智能体(Agent)、组件(Component)、管道(Pipeline)、可调用函数(Callable Function)。其中智能体类型的工具控制了其余三类

工具,主要负责调度;而组件、管道、可调用函数则负责处理并输出测试用例的组成成分。

[0122] 关于工具的更详细内容例如工具之间的关系如下文工具树部分介绍。

[0123] 工具树

[0124] 在航空软件测试领域,完成一个复杂的任务需要几百个步骤,其中还有大量的重复步骤,如果将所有步骤都线性地组织在一个或几个智能体中,必然会大量消耗大模型的上下文窗口的token数甚至溢出,导致模型失去正常能力或显存溢出。同时据调研,目前开源模型中一般仅支持4096tokens的上下文窗口,目前有一些工作正在研究长窗口模型,但过长的窗口一般也会带来更大的显存需求。线性地组织所有步骤还会导致整个计划的迁移性不强,无法针对性地复用其中的某些功能过程。针对这两个问题,受传统编程中函数封装的启发,设计了名为工具树的规划执行架构,通过配合专业的航空软件测试人员的测试经验,将常用的步骤过程封装一个个智能体类型的工具。但这些工具与传统函数不同,后者有着固定的处理逻辑,即使是泛型编程,也需要代码能力较强的程序员来操作,而本发明设计的智能体类型的工具可以由其中的自然形式高层规划来控制具体的行为,这对于用户来说大大降低了使用门槛。

[0125] 本发明设计的工具树拥有和树结构一样的性质,如图6所示,其中叶结点是不可再分的元工具,即组件、可调用函数类型的工具,例如表格检索器、行列选择器、字段解释器等,这对应了测试用例的组成成分;中间节点中智能体类型的工具可根据反馈自动修改或人工修改该节点的高层规划,使其重新选择、实例化和排列组合其成员工具,此时工具树中一部分是原有节点,一部分则是新节点,从而使得整个系统能够快速变换模拟对象或场景。更进一步地,这种策略使得发明新的工具成为可能。

[0126] 类似于拓扑排序,工具树同级节点之间存在前后调用顺序限制关系,例如在调用<行列选择器>之前,必须调用<表格检索器>,这需要在上下文指令中说明。

[0127] 在实现方面,不同的指令文本标识了不同的智能体,但调用的大模型是同一个,也就是说在不同的上下文指令下,大模型可以执行不同的任务。

[0128] 当某个智能体类型工具节点输出响应,规划出下一步将要执行的步骤(包括思考、工具名、工具输入),会在后台解析该响应,获取工具名和工具输入,然后运行该工具得到运行结果。

[0129] 根据运行结果的性质,对其采用不同的策略:如果该运行结果涉及多跳推理,那么参照ReAct做法,将运行结果反馈到该智能体的上下文窗口中,参与后续规划步骤;如果运行结果不涉及后续推理,或者含有多个tokens(例如40以上),特别是表格内容等,将其卸载到记忆模块缓存,具体情况将在记忆模块部分介绍。

[0130] 在应用方面,可以使用YAML文件来快速创建工具树。

[0131] 最后,工具树的过程封装机制也为指令复用提供了准备工作。

[0132] 校验模块

[0133] 在上文的指令分解模块提到校验模块反馈错误原因给低级规划模块,帮助其重新执行当前步骤,保证了整个任务完成过程的稳定性。在执行低级规划模块提出的工具和工具输入时,会将预期结果的信息传递给校验模块,如果不符合预期信息,例如预期输出三个整数,实际却输出两个整数;预期输出浮点类型或整型,实际却输出其他字符串,那么校验模块则会将错误信息反馈给底层规划模块。

[0134] 在智能测试智能框架程序运行期间,任何可能抛出异常的位置,都可以反馈到底层规划模块以及高层规划(指令分解模块)。例如,在自定义工具“行列选择器”的实现中,如果待操作表格为空,那么程序会抛出异常提升“行列选择器之前没有执行表格检索器,或表格检索器执行出错。请重新执行表格检索步骤”。通过将抛出的异常提示回调到根节点(或者有更好的选择)的大模型指令提示中,就实现了整个系统的运行流程更新。

[0135] 记忆模块

[0136] 人类的记忆分为短期记忆和长期记忆,其中短期记忆是指大脑对信息和任务的即时处理和存储。它能够暂时保留信息,持续几秒到几分钟,如果没有通过重复或其他方式加以巩固,通常会迅速遗忘;长期记忆是对信息的长期存储。与短期记忆相比,它的容量几乎是无限的,涉及了经验、知识和技能。

[0137] 在上文提到大型语言模型的上下文窗口通常仅有4090个token,这相当于人类的短期记忆,包含了模型处理当前任务的情况,例如保持对话的一致性和相关性、推理演绎等;而系统中的记忆模块则充当了长期记忆的角色,本发明设计的记忆模块主要有三个作用。

[0138] 第一,收集每个智能体类型工具的交互过程作为历史交互记录,为长窗口对话提供了条件;方便排查系统故障;同时也可以用于指令复用,收集这些成功执行的工具链交互轨迹,当面临类似的新场景时,可以将其作为示例或者根据实际上下文进行调整;收集的交互过程也是实际的运行场景,是满足Agent tuning训练数据格式的天然标记数据。

[0139] 第二,缓存机制,缓存所有类型工具输出的中间结果。使用当前工具位置所在的工具链路径涉及的各个工具名组成的字符串作为一层键,当前工具的输入的哈希值作为二层键,这是因为即使是同一工具链路径下的同一层中具有前后关系的工具中可能存在同名工具,因此将工具输入作为识别同一工具的不同运行。这样做的好处有多重:一是节约tokens,大大减轻了模型的上下文窗口压力,把剩余的上下文空间留给推理过程;二是方便维护中间变量,大模型不可避免地会产生幻觉,在上下文不断变长时更是如此,因此及时保存这些中间变量有助于系统的准确性;三是航空软件测试用例的生成要求提供依据,及时将工具链名称和工具输入保存下来供用户查询;四是提高系统处理效率,在测试用例过程中,工具树的各个节点都可能重复访问某个数据源或用相同的工具处理相同的输入,通过直接访问缓存,可以明显提高系统的响应速度。

[0140] 第三,测试用例维护与管理,在企业级应用中,一个比较集中的问题就是测试用例的复用、继承和重载问题。如果没有很好的维护方式,到了一定规模就很难维护。当生成测试用例时,就将其对应依据、生成过程、组成成分、最终测试用例联系在一起,可以有效地维护和管理大量测试用例集合。

[0141] 测试用例生成算法(探索模块)

[0142] 借助大模型智能体出色的规划能力、与外部工具交互反馈的能力来改进传统的测试用例生成算法,例如遗传算法、IPO算法、蚁群算法等。在本发明中,智能体的主要作用是分析测试结果和环境反馈,例如测试用例的覆盖率、发现的错误种类、执行时间等指标,调整遗传算法的参数和策略,从而提高软件测试的效率和效果。

[0143] 在系统架构方面,本部分对应图3的探索模块。根据飞行测试仪器的反馈,如果输入的某测试用例使得飞行测试仪器显式本次测试不通过,且测试用例不存在异常,那么本

测试用例中的组成成分,例如已采样的变量参数和模拟的故障列表是待搜索的重点方向,通过在遗传算法等现有搜索算法中赋予这些变量参数和故障列表更大的权重,这样的实时反馈改进了遗传算法容易陷入局部最优的情况,从而缓解了遗传算法容易忽略实际故障的情况。

[0144] 具体措施有以下几个步骤。

[0145] 动态调整遗传算法参数

[0146] 智能体分析每轮测试的结果,尤其关注未发现故障的测试用例。如果智能体发现某些故障类型(如数据更新故障)被频繁忽视,它可以选择调整遗传算法,增加这类故障的权重,从而生成更可能揭示此类故障的测试用例。

[0147] 引入专家知识

[0148] 将航空系统测试的历史数据和专家知识纳入系统的本地知识库。智能体使用这些信息来指导遗传算法,例如,如果历史数据显示C通道更容易出现信号故障,智能体可以调整算法以生成更多涉及C通道的测试用例。

[0149] 实时反馈循环

[0150] 智能体实时监控仿真设备的反馈验证信息。例如分析测试仪器的反馈,如果发现实际结果与预期不符,立即反馈给遗传算法。遗传算法利用这些信息来调整后续的测试用例生成,例如,增加可能导致类似问题的属性组合的探索。

[0151] 具体的步骤如下:

[0152] 初始化遗传算法:生成初步的测试用例集,涵盖所有参数和故障类型的组合。

[0153] 执行测试并收集数据:运行测试用例,智能体收集测试仪器的反馈以及测试过程中的任何异常。

[0154] 分析测试结果:智能体分析测试数据,确定哪些故障类型最常被检测到,哪些最容易被忽略。

[0155] 调整算法参数:基于智能体的分析,调整遗传算法的适应度函数或算子权重,以更好地探索未被充分测试的故障类型。

[0156] 迭代和优化:重复测试和分析过程,持续优化遗传算法,直至达符合要求的故障检测率和测试覆盖率。

[0157] 该智能体系统能够最大程度重复利用指令模板以及在任务执行过程中产生的成功或失败的执行轨迹,这使得系统具备泛化、迁移的能力。基于航空测试领域内经验的一致性和普适性,该系统可以从某一型号部件的有限测试场景中所生成的测试用例,推广到该型号其他测试场景、不同型号的同类部件,甚至是不同的部件。

[0158] 指令重用模块

[0159] 为了能够最大程度重复利用指令模板以及在任务执行过程中的产生的成功或失败的执行轨迹,达到让系统能够从自己的经历中自主学习的效果,参考Andrew Zhao等人的工作设计了指令重用模块,主要涉及收集经验、从经验中提取宏观规律、使用曾经成功的执行轨迹和宏观规律尝试执行未见过的任务。

[0160] 收集经验

[0161] 在收集经验阶段,也是“训练阶段”,与Andrew Zhao等人提出的“经验池”类似,本发明系统在“指令分解模块-规划模块-工具树执行反馈”这一整个过程中,通过记忆模块收

集每个节点的交互过程、中间结果以及外部反馈,如果该节点是子智能体,本发明还收集该节点的上层规划。具体来说,在最多Z次的时间内不断重试训练任务。首先定义以下符号:

[0162] $\tau_{n,z}$:表示第n个任务的第z次尝试的执行轨迹;

[0163] F_{examp}:表示与$\tau_{n,z}$相关的Few-shot示例;

[0164] $V_{n,z}$:表示第n个任务的第z次及之前所有尝试获取的所有反馈(最初$V_{n,0}$是空字符串);

[0165] 在接下来的过程中,智能体会经历几个状态:

[0166] 初始状态:智能体尝试使用与第n个任务初次尝试的执行轨迹τ相关的Few-shot

[0167] n,0

[0168] 示例作为上下文来执行任务,使用ReAct作为大型语言模型(LLMs)的基本规划算法,将执行轨迹$\tau_{n,0}$、轨迹示例F_{examp}、初始反馈$V_{n,0}$进行拼接,作为模型的输入:

[0169] $LLMReAct(\cdot|\tau_{n,0},F_{examp},V_{n,0})$

[0170] 在第z次实验中(z的取值范围是0到Z),当智能体完成任务或达到最大步数时,记忆模块收集轨迹τ。如果智能体成功完成任务,则执行下一个任务。

[0171] n,z

[0172] 如果智能体失败,让智能体回顾失败的轨迹并进行自我反思,产生当前步的反馈,并与之前的反馈拼接拼接进行结合:

[0173] $V_{n,z+1}=concat(V_{n,z},LLMreflect(\tau_{n,z}))$

[0174] 在下一次重试该任务时,智能体会使用反馈$V_{n,z+1}$来增强上下文,此时上下文是:

[0175] $LLMReAct(\cdot|\tau_{n,z+1},F_{exa\square\square},V_{n,z+1})$

[0176] 在经历这几个状态的过程,记忆模块可以收集成功或失败的轨迹。

[0177] 从经验中学习并提取宏观经验

[0178] 从经验中学习并提取宏观经验类似非策略学习(off policy learning,Q-learning),其中智能体可从行为策略的经验中学习。

[0179] 给LLM代理的指令I可以分解为任务说明和Few-shot示例。

[0180] (1)类似的经验作为示例

[0181] 其中对于示例,从记忆模块中检索成功的轨迹,根据与待评估任务具有最大内积文本相似度的前k个成功轨迹(按需要进行混合检索以及重排)。

[0182] (2)从成功和失败中学习

[0183] 宏观经验通常指某些领域中普遍适用的、具有泛化性质的抽象规律和原则。与Andrew Zhao在通用领域问答中积累的宏观经验不同,本发明专注于航空测试领域的经验积累,系统通过不断从成功或失败轨迹中总结出航空领域中普遍适用的、具有泛化性质的抽象规律和原则,构建“宏观经验”集合。值得注意的是,本发明创新性地将该集合作为一个敏感信息接口,这比原工作的动机更加必要。这是因为当前在许多领域,包括一些高度重视保密的领域,通用而具体的测试经验常常只存在于专业人士的记忆中,而不会被明确记录在书面文档上。例如,关于“测试规则通常位于哪个章节”、“如何在多个含有目标关键词的段落中找到需提取的具体段落”等信息通常不会书面化。因此,通过构建一个“宏观经验”集合的接口,使用者可以实时提供一些敏感而保密的信息,以帮助更有效地生成测试用例,同时确保这些信息不被泄露。

[0184] 具体来说,本发明按照Andrew Zhao等人的做法为模型提供几个操作符,维护一个“宏观经验”集合,该集合初始为空集。迭代地为模型提供失败和成功轨迹对,或者提供从记忆模块中检索的L个成功轨迹(L的取值范围是2到8)。

[0185] 模型可以执行的操作是:ADD(添加)一个新的见解,EDIT(编辑)现有见解的内容,DOWNVOTE(否决)不同意现有见解;UPVOTE(赞成票)与现有见解一致。

[0186] 新添加的见解初始将被赋予两个重要性计数,并且如果随后应用了UPVOTE或EDIT操作,这个计数将会增加;如果应用了DOWNVOTE操作,计数则会减少。如果某个见解的重要性计数降到零,它将会被移除,如表1所示。

[0187] 表1

[0188]

您是一个高级推理代理，可以根据对过去任务轨迹的新批判，在您现有的规则集中添加、编辑或删除规则。您将会得到...
两个以前的任务试验，在其中您...
[任务描述]...
一个成功的和一个不成功的试验。您之所以失败是因为...
[任务失败的原因]。
以下是两个用来比较和评判的之前试验：
[失败/成功轨迹]
这里是现有的规则：
[当前存在的见解]
通过审查...
并与成功的试验进行对比...
以及现有规则列表，您可以执行以下操作：添加、编辑、反对或支持，以使新规则是一般性和高层次的...
对失败试验的批评...
或提出的思考方式，以便...
在将来遇到不同问题时避免类似的失败。
重点强调...
批评如何...
进行更好的思考和行动。
请按照这些指导进行您的操作。
遵循下面的格式：
<操作><规则编号>： <规则>
可用的操作有：UPVOTE（如果现有规则对任务非常相关）、DOWNVOTE（如果某一现有规则与其他现有规则相矛盾或类似/重复）、EDIT（如果任何现有规则不够通用或可以增强。重写并改进它）、ADD（添加与现有规则非常不同且与其他任务相关的新规则）。每个操作都需要严格遵循相应的格式：
UPVOTE <现有规则编号>：<现有规则>
DOWNVOTE <现有规则编号>：<现有规则>
EDIT <现有规则编号>：<新修改的规则>
ADD <新规则编号>：<新规则>
不要在规则中提到试验，因为所有规则都应该是普遍适用的。每个规则应该简洁易懂。任何操作都可以多次使用。每个现有规则最多只能进行一次操作。以下是您对以上现有规则列表所进行的操作：

[0189] 在推理评估阶段,智能体尝试生成它之前未见过的测试用例。这包括针对同一设备在不同测试场景(如多余度通信与多余度投票)的测试用例,以及适用于不同型号设备的测试用例生成任务。

[0190] 提取前两个阶段收集的见解和成功轨迹,并建立成功轨迹的向量知识库。对于每项任务,任务描述由提取的完整见解列表串联来扩充,即$\hat{l}=\mathrm{concat}(l_1,l_2..l_n)$,当见解过多时,可以通过检索技术缓解上下文窗口压力。然后检索具有最高任务相似性的前k个轨迹作为Few-shot上下文示例F_{retrie_examp}。本发明参考Andrew Zhao设计了一个示例指令模板,如图7所示。

[0191] 值得强调的是,结合本发明的高低层规划和工具树,该方法还展现了更强的迁移学习潜力。以航空测试领域为例,可以从某一型号部件的有限测试场景中所生成的测试用例,推广到该型号其他测试场景、不同型号的同类部件,甚至是不同的部件。这得益于航空测试领域内经验的一致性和普适性,这些经验可以指导指令分解模块制定恰当的上层规划,从而影响具体执行的下层规划,并最终决定工具树的组合方式。如此一来,便能迅速实现对其他情境的泛化应用。

[0192] 具体的做法是,使用从源测试场景中提取的宏观见解或使用人员实时注入的宏观见解和从目标测试场景提取较少的执行轨迹示例(例如2～4个执行轨迹示例)来“适配”宏观见解,使其更适用于目标测试场景的高低层规划,进而指导执行模块使用合适的工具链。

[0193] 本发明提供了大模型智能体驱动的航空文档分析与测试用例生成系统,具体实现该技术方案的方法和途径很多,以上所述仅是本发明的优选实施方式,应当指出,对于本技术领域的普通技术人员来说,在不脱离本发明原理的前提下,还可以做出若干改进和润饰,这些改进和润饰也应视为本发明的保护范围。本实施例中未明确的各组成部分均可用现有技术加以实现。

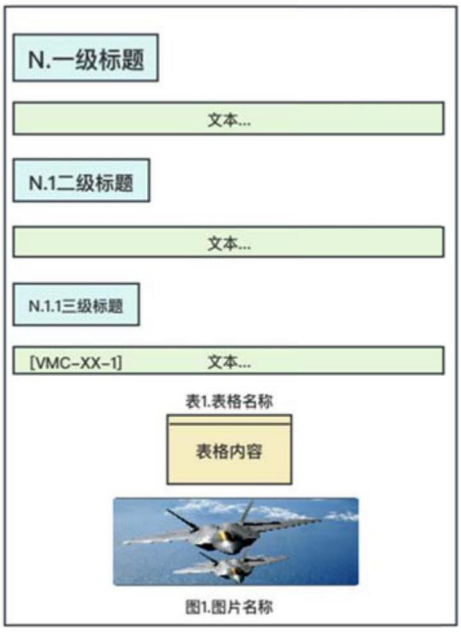

N.一级标题
文本...
N.1二级标题
文本...
N.1.1三级标题
[VMC-XX-1] 文本...
表1.表格名称
表格内容
图1.图片名称

图1


解析各类文档：PDF、Docx、Markdown...
标题栈
表格
文本
图片
表格序列化处理
上下文结构自适应切割
嵌入向量化
本地知识库

图2

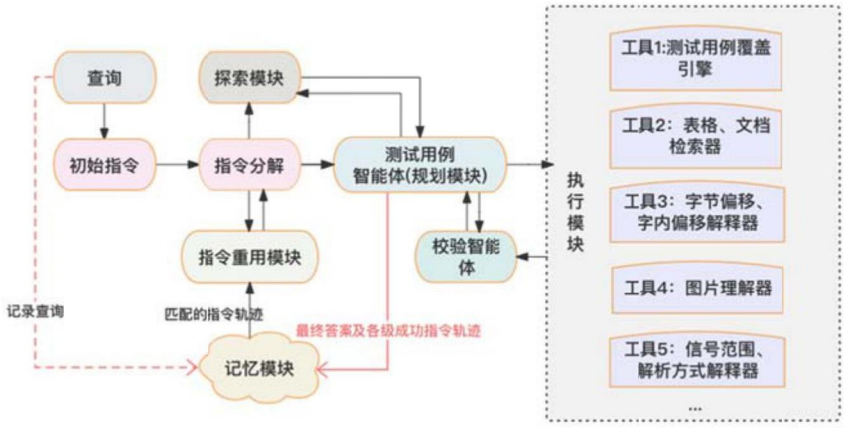

查询
探索模块
初始指令
指令分解
测试用例
智能体(规划模块)
指令重用模块
校验智能体
记录查询
匹配的指令轨迹
最终答案及各级成功指令轨迹
记忆模块
执行模块
工具1:测试用例覆盖引擎
工具2：表格、文档检索器
工具3：字节偏移、字内偏移解释器
工具4：图片理解器
工具5：信号范围、解析方式解释器
...

图3

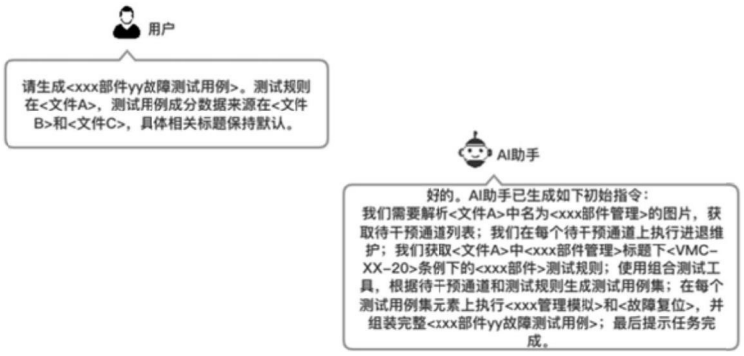

用户
请生成<xxx部件yy故障测试用例>。测试规则在<文件A>，测试用例成分数据来源在<文件B>和<文件C>，具体相关标题保持默认。
AI助手
好的。AI助手已生成如下初始指令：
我们需要解析<文件A>中名为<xxx部件管理>的图片，获取待干预通道列表；我们在每个待干预通道上执行进退维护；我们获取<文件A>中<xxx部件管理>标题下<VMC-XX-20>条例下的<xxx部件>测试规则；使用组合测试工具，根据待干预通道和测试规则生成测试用例集；在每个测试用例集元素上执行<xxx管理模拟>和<故障复位>，并组装完整<xxx部件yy故障测试用例>；最后提示任务完成。

图4

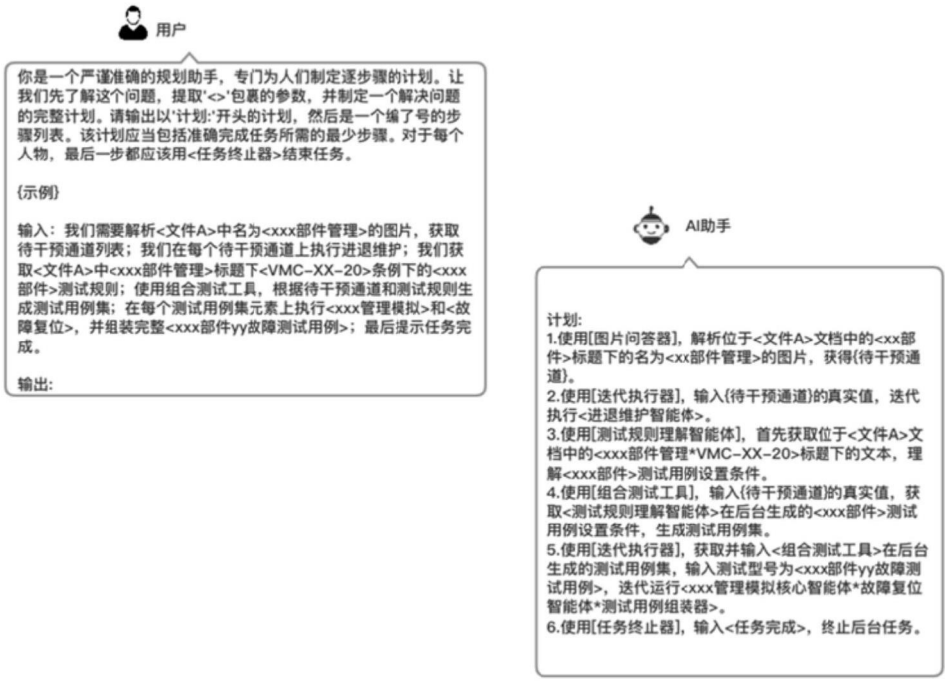

用户
你是一个严谨准确的规划助手，专门为人们制定逐步骤的计划。让我们先了解这个问题，提取'<>'包裹的参数，并制定一个解决问题的完整计划。请输出以'计划:'开头的计划，然后是一个编了号的步骤列表。该计划应当包括准确完成任务所需的最少步骤。对于每个人物，最后一步都应该用<任务终止器>结束任务。
{示例}
输入：我们需要解析<文件A>中名为<xxx部件管理>的图片，获取待干预通道列表；我们在每个待干预通道上执行进退维护；我们获取<文件A>中<xxx部件管理>标题下<VMC-XX-20>条例下的<xxx部件>测试规则；使用组合测试工具，根据待干预通道和测试规则生成测试用例集；在每个测试用例集元素上执行<xxx管理模拟>和<故障复位>，并组装完整<xxx部件yy故障测试用例>；最后提示任务完成。
输出:
AI助手
计划:
1.使用[图片问答器]，解析位于<文件A>文档中的<xx部件>标题下的名为<xx部件管理>的图片，获得{待干预通道}。
2.使用[迭代执行器]，输入{待干预通道}的真实值，迭代执行<进退维护智能体>。
3.使用[测试规则理解智能体]，首先获取位于<文件A>文档中的<xxx部件管理*VMC-XX-20>标题下的文本，理解<xxx部件>测试用例设置条件。
4.使用[组合测试工具]，输入{待干预通道}的真实值，获取<测试规则理解智能体>在后台生成的<xxx部件>测试用例设置条件，生成测试用例集。
5.使用[迭代执行器]，获取并输入<组合测试工具>在后台生成的测试用例集，输入测试型号为<xxx部件yy故障测试用例>，迭代运行<xxx管理模拟核心智能体*故障复位智能体*测试用例组装器>。
6.使用[任务终止器]，输入<任务完成>，终止后台任务。

图5

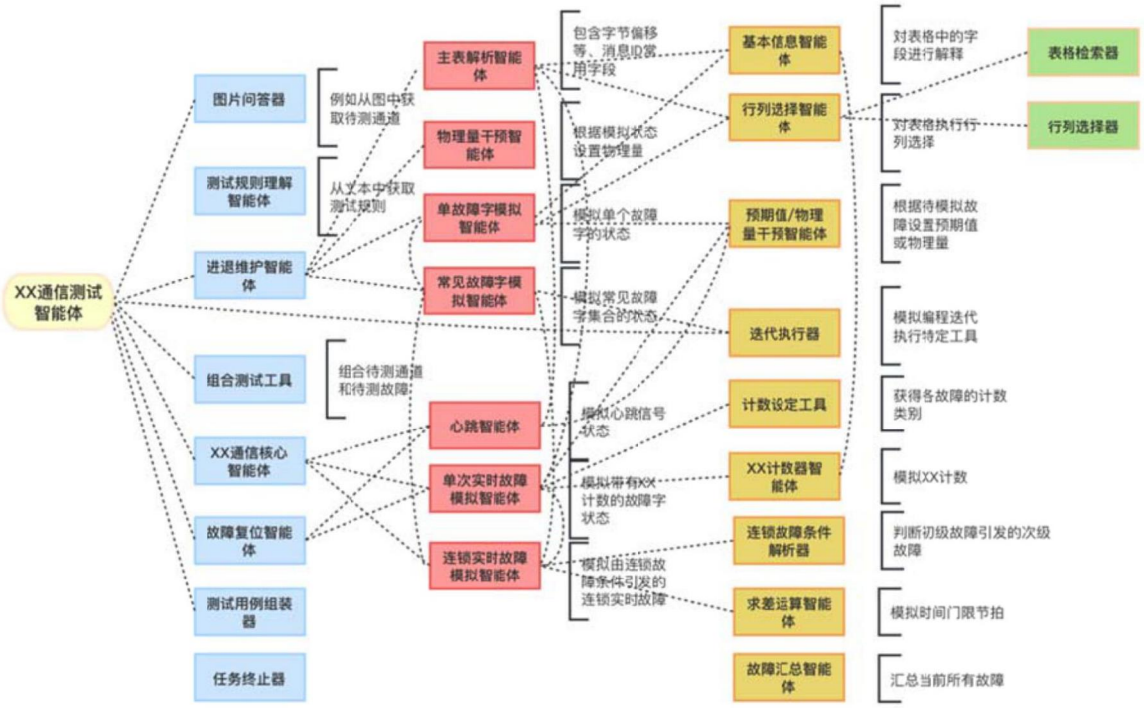

XX通信测试智能体
图片问答器
测试规则理解智能体
进退维护智能体
组合测试工具
XX通信核心智能体
故障复位智能体
测试用例组装器
任务终止器
例如从图中获取待测通道
从文本中获取测试规则
组合待测通道和待测故障
主表解析智能体
物理量干预智能体
单故障字模拟智能体
常见故障字模拟智能体
心跳智能体
单次实时故障模拟智能体
连锁实时故障模拟智能体
包含字节偏移等、消息收发单字段
根据模拟状态设置物理量
模拟单个故障字的状态
模拟常见故障字集合的状态
模拟心跳信号状态
模拟带有XX计数的故障字状态
模拟由连锁故障条件引发的连锁实时故障
基本信息智能体
行列选择智能体
预期值/物理量干预智能体
迭代执行器
计数设定工具
XX计数器智能体
连锁故障条件解析器
求差运算智能体
故障汇总智能体
对表格中的字段进行解释
对表格执行行列选择
表格检索器
行列选择器
根据待模拟故障设置预期值或物理量
模拟编程迭代执行特定工具
获得各故障的计数类别
模拟XX计数
判断初级故障引发的次级故障
模拟时间门限节拍
汇总当前所有故障

图6

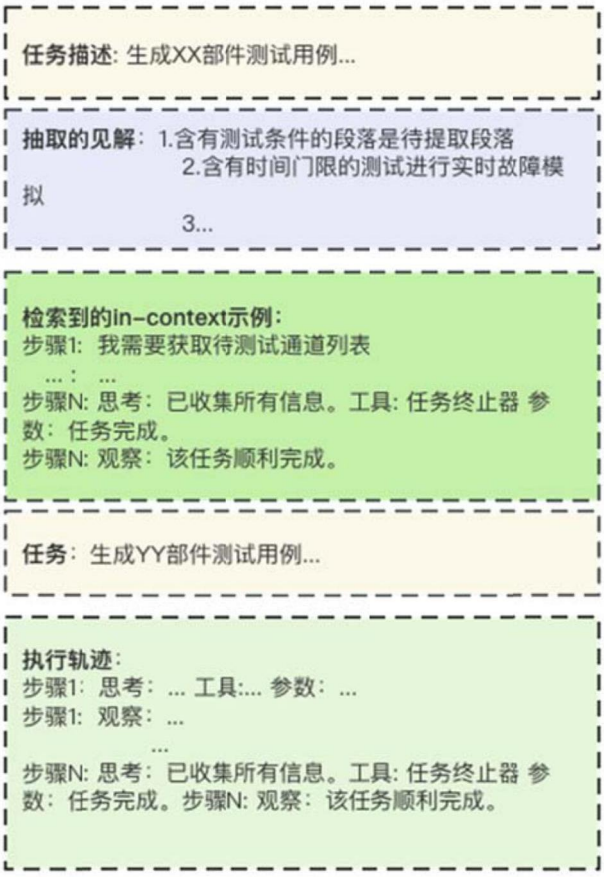

任务描述: 生成XX部件测试用例...
抽取的见解: 1.含有测试条件的段落是待提取段落
2.含有时间门限的测试进行实时故障模拟
3...
检索到的in-context示例:
步骤1: 我需要获取待测试通道列表
... : ...
步骤N: 思考: 已收集所有信息。工具: 任务终止器 参数: 任务完成。
步骤N: 观察: 该任务顺利完成。
任务: 生成YY部件测试用例...
执行轨迹:
步骤1: 思考: ... 工具:... 参数: ...
步骤1: 观察: ...
...
步骤N: 思考: 已收集所有信息。工具: 任务终止器 参数: 任务完成。步骤N: 观察: 该任务顺利完成。

图7

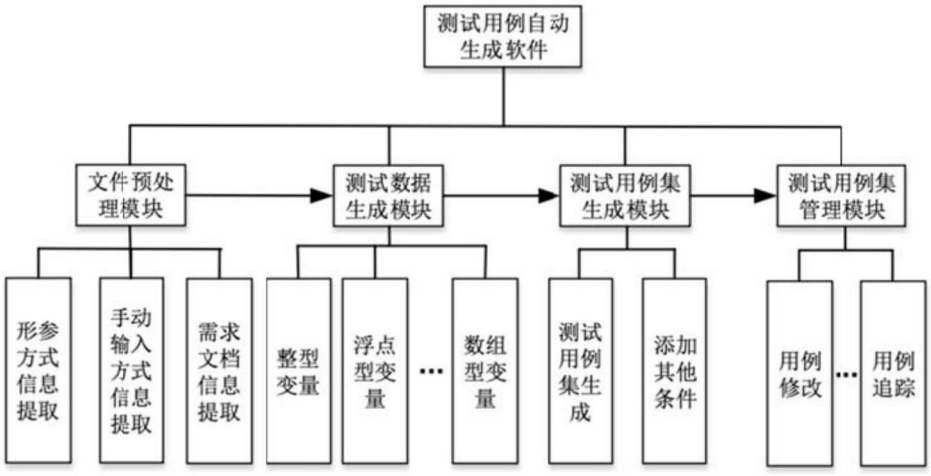

测试用例自动生成软件
文件预处理模块
测试数据生成模块
测试用例集生成模块
测试用例集管理模块
形参方式信息提取
手动输入方式信息提取
需求文档信息提取
整型变量
浮点型变量
数组型变量
测试用例集生成
添加其他条件
用例修改
用例追踪

图8