控制编号：HNZM/JL-083

CMA
221821130550

CNAS
中国认可
检测
TESTING
CNAS L9548

测 试 报 告

Testing Report

文档编号（Reg_ID）：HNZM-2025-YS-158-V1.0

项目编号（Project ID）：	HNZM-2025-YS-158
项目名称（Project Name）：	融合多学科知识与原理的认知智能系统、物质科学本体-知识库、大模型轻量化系统
软件版本(Version NO)：	V2.0、V1.0、V1.0
测试类型（Test Type）：	验收测试
委托单位（Client）：	中国科学技术大学

本报告未经公司书面批准，不准部分复印。
The report must not be partially duplicated without permission authorized by HNZM.

湖南卓码软件测评有限公司

注 意 事 项

1、报告无检测单位“检测专用章”无效。

2、复制报告未重新加盖检测单位公章无效。

3、报告无编制、审核、批准人签章无效。

4、报告部分复印无效。

5、报告涂改无效。

6、对测试报告若有异议，应于收到报告之日起十五日内向检测单位提出，逾期不予受理。

7、一般情况，测试报告仅对收到的样品负责。

8、测试结论数据详见“测试结果”部分。

地 址：湖南省长沙市高新开发区岳麓西大道588号芯城科技园5幢7层703房

邮 编：410205

电 话：400-607-0568/0731-82568809

网 址：https://www.zmtests.com

修改历史记录

编号	日期	修改人	批准人	所修改章节及页码	修改的原因及内容概要	修改/创建
01	2025.12.24	吴涛	熊亦男	ALL		创建

目 录

1 项目概况 1

2 测试结论 3

3 测试依据及环境 5

3.1 测试工具介绍 6
3.1.1 LoadRunner 6

4 测试结果 7

4.1 病毒检查 7
4.2 质量测试结果 7
4.2.1 功能性测试结果 7
4.2.1.1 融合多学科知识与原理的认知智能系统功能性测试 7
4.2.1.2 物质科学本体-知识库功能性测试 7
4.2.1.3 大模型轻量化系统功能性测试 8
4.2.2 性能效率测试结果 8
4.2.2.1 融合多学科知识与原理的认知智能系统性能效率测试 8
4.2.2.2 大模型轻量化系统性能效率测试 9
4.2.3 可靠性测试结果 9
4.2.3.1 融合多学科知识与原理的认知智能系统可靠性测试 9

5 测试用例与记录 10

5.1 功能性测试 10
5.1.1 融合多学科知识与原理的认知智能系统功能性测试 10
5.1.1.1 智能体基础交互 10
5.1.1.2 物质科学推理 10 步响应准确率测试 11
5.1.2 物质科学本体-知识库功能性测试 13
5.1.2.1 数据量统计测试 13
5.1.2.2 自然语言知识的结构规范度测试 15
5.1.2.3 知识方程语法通过率测试 17
5.1.2.4 知识图谱的结构规范度测试 18
5.1.3 大模型轻量化系统功能性测试 20
5.1.3.1 4bit 量化模型与基准模型对比测试 20
5.1.3.2 8bit 量化模型与基准模型对比测试 22
5.1.3.3 优化部署模型的准确率得分损伤测试 24
5.1.3.4 优化部署模型的 AUROC 得分损伤测试 28
5.2 性能效率测试 31
5.2.1 融合多学科知识与原理的认知智能系统性能效率测试 31
5.2.1.1 物质科学推理 10 步响应时间测试 31
5.2.2 大模型轻量化系统性能效率测试 33
5.2.2.1 资源利用率测试 33
5.2.2.2 吞吐量测试 35

5.2.2.3 推理时间 36
5.3 可靠性测试 37
5.3.1 融合多学科知识与原理的认知智能系统可靠性测试 37
5.3.1.1 成熟性 37

1 项目概况

项目名称	融合多学科知识与原理的认知智能系统、物质科学本体-知识库、大模型轻量化系统
项目简介	**融合多学科知识与原理的认知智能系统：** 该软件是第三代知识与逻辑增强大模型对应的软件系统。该系统使用物质科学领域多维度知识库实施知识增强、事实验证增强与逻辑验证增强等技术，进一步强化模型，10步推理准确率在众多基座模型上提升。与通用大模型和其他领域模型相比，本成果基于数据库自动反馈技术利用到了物质科学领域的结构化数据，基于知识增强与验证增强技术使用了物质科学领域的（形式化）知识，因此具备更好的物质科学领域理解与推理能力。 **物质科学本体-知识库：** 本体与知识库：针对大模型知识与逻辑增强的知识底座建设需求，本系统是一套4维知识库，包含4种不同层级的知识模型形态：1、长文本知识：来自书籍、Wiki、论文等自然语言文档级别的源知识，是各形态知识构建的起点与基础。2、短文本知识：通过对长文本的结构特征与语义分段，将其切分为段落级、可被索引的知识单元。3、知识图谱：基于所有短文本，在LLM的辅助下进行实体、关系抽取，形成显式结构化五元组的知识表示，即（主体，主体类型，关系，客体，客体类型）。4、知识方程：在本体库基础上，对原始自然语言文本进行形式化转换，形成可计算、可推理的逻辑表达形式。 **大模型轻量化系统：** 该软件是一套面向化学大语言模型的量化与部署方案，运行于 Linux 平台，采用 Python 语言开发。系统以主流大模型推理框架为基础，支持在分布式推理环境下对化学领域大模型进行量化处理与推理部署。软件主要包含AWQ权重量化处理、离群值分析与截断、量化参数配置以及推理验证模块。该系统可对模型权重进行低比特量化，并支持在消费级显卡运行化学大模型推理任务。该软件适用于化学大模型在资源受限场景下的模型部署。
样品描述	1. 融合多学科知识与原理的认知智能系统 V2.0【20251217-001】 2. 物质科学本体-知识库 V1.0【20251217-002】 3. 大模型轻量化系统 V1.0【20251217-003】

术语表	/		
委托单位名称	中国科学技术大学		
委托单位地址	安徽省合肥市金寨路96号		
送 检 者	程秀秀	电 话	15156083203
收样日期	2025.12.17	检测日期	2025.12.17～2025.12.24
测试单位	湖南卓码软件测评有限公司		
测试地点	长沙高新开发区岳麓西大道588号芯城科技园5栋7层703房		
测 试	吴涛	测试日期	2025.12.18～2025.12.23
编 制	吴涛	编制日期	2025.12.24
审 核	刘子敬	审核日期	2025.12.24
签 发	熊奔男	签发日期	2025.12.26

2 测试结论

融合多学科知识与原理的认知智能系统、物质科学本体-知识库、大模型轻量化系统

湖南卓码软件测评有限公司受中国科学技术大学的委托，按照GB/T 25000.51-2016《系统与软件工程 系统与软件质量要求和评价（SQuaRE）第51部分：就绪可用软件产品（RUSP）的质量要求和测试细则》中的条款，结合《HNZM_ZY_002_软件产品检测作业指导书II》对软件的功能性、性能效率、可靠性进行测试。

一、功能性

本次对融合多学科知识与原理的认知智能系统V2.0、物质科学本体-知识库V1.0、大模型轻量化系统V1.0中的功能进行了测试，经测试，结果表明：

1. 融合多学科知识与原理的认知智能系统

1） 智能体基础交互：智能体请求可正常被调用；

2） 第一次执行结果：使用物质科学10步推理测试数据集（625题）进行测试，实际推理的平均准确率为 63.04%；第二次执行结果：使用物质科学10步推理测试数据集（625题）进行测试，实际推理的平均准确率为 63.52%。

2. 物质科学本体-知识库

1） 自然语言知识条目数量统计正确，知识方程数量统计正确，知识图谱五元组数量统计正确；

2） 使用测试脚本对自然语言知识库中的1269050条数据是否符合预定的字段规范进行检测，从而得到符合结构规范的自然语言知识条目数，最终通过公式计算得到自然语言知识的结构规范度为100%；

3） 使用测试脚本对知识方程知识库中1568条数据进行语法解析检测，得到合法的知识方程条目数，最终通过公式计算得到知识方程的语法通过率为100%；

4） 使用测试脚本对知识图谱知识库中1186457条数据进行空值、冗余、元组检测，从而得到合法的知识图谱元组数，最终通过公式计算得到知识图谱的结构规范度为100%。

3. 大模型轻量化系统

1） 4bit量化模型与基准模型对比测试：基于9个ChemBench4K数据集推理结果的平均准确率，4比特量化模型相对基准模型(16bit量化模型)的性能损失率为3.78%；

2） 8bit量化模型与基准模型对比测试：基于9个ChemBench4K数据集推理结果的平均准确率，8比特量化模型相对基准模型(16bit量化模型)的性能提升率为0.09%；

3） 优化部署模型的准确率得分损伤测试：基于9个ChemBench4K数据集推理结果的准确率，优

化部署模型对9个ChemBench4K数据集推理结果准确率得分损伤率都为0；

4） 优化部署模型的AUROC得分损伤测试：基于9个ChemBench4K数据集推理结果中每道题ABCD选项的预测概率绘制ROC曲线，进行平均AUROC得分对比，优化部署模型的平均AUROC得分损伤率为0。

二、性能效率

本次对融合多学科知识与原理的认知智能系统V2.0、大模型轻量化系统V1.0中的性能效率进行了测试，经测试，结果表明：

1．融合多学科知识与原理的认知智能系统

1） 第一次执行结果：使用物质科学10步推理测试数据集（625题）进行测试，实际推理总耗时6小时57分33秒，平均推理耗时为 40.08 秒；第二次执行结果：使用物质科学10步推理测试数据集（625题）进行测试，实际推理总耗时6小时3分53秒，平均推理耗时为34.93 秒。

2．大模型轻量化系统

1） 显存利用率、吞吐量、推理时间分析结果下表所示（tg32代表部署框架在单批次下连续生成长度为32的随机Token序列的解码任务；tg64代表部署框架在单批次下连续生成长度为64的随机Token序列的解码任务）：

	显存利用率	tg32吞吐量（t/s）	tg64吞吐量（t/s）	tg32推理时间（ms/token）	Tg64推理时间（ms/token）
原始部署框架	87.48%	3.21	3.21	311.15	311.16
优化部署框架	99.00%	4.31	4.32	231.99	231.28
提升率	13.17%	34.27%	34.58%	25.44%	25.67%

三、可靠性

本次对融合多学科知识与原理的认知智能系统V2.0中的成熟性进行了测试，经测试，结果表明：

1．使用LoadRunner工具发送智能体请求，持续24小时，智能体请求响应正常，系统没有出现故障、服务中断的情况。

湖南卓码软件测评有限公司

2025 年 12 月 26 日

3 测试依据及环境

<table>
<tr><td colspan="4">测试依据</td></tr>
<tr><td>测试标准</td><td colspan="3">GB/T 25000.51-2016《系统与软件工程 系统与软件质量要求和评价（SQuaRE）第51部分：就绪可用软件产品（RUSP）的质量要求和测试细则》</td></tr>
<tr><td>用户文档集及相关文档</td><td colspan="3">《HNZM_ZY_002_软件产品检测作业指导书II》</td></tr>
<tr><td colspan="4">测试环境</td></tr>
<tr><td colspan="3">硬件环境</td><td rowspan="2">软件环境</td></tr>
<tr><td>编号/名称</td><td>型号</td><td>配置（CPU/内存/硬盘/…）</td></tr>
<tr><td>测试终端01</td><td>兼容机</td><td>CPU：2 x Intel(R) Xeon(R) Gold 6430 (128) @ 3.40 GHz
GPU：NVIDIA H100 80G*8
内存：1007GB
硬盘：846GB
带宽：100Mbps</td><td>操作系统：Debian GNU/Linux 13 (trixie) x86_64
软件环境:python 3.13,docker 28.4.0, torch 2.9.0, langchain 1.1.0
被测系统：融合多学科知识与原理的认知智能系统V2.0</td></tr>
<tr><td>测试终端02</td><td>兼容机</td><td>CPU：64 × Intel(R) Xeon(R) Silver 4314 CPU @ 2.40GHz
GPU：2 × NVIDIA RTX 6000 Ada Generation
内存：256GB
硬盘：10TB
带宽：100Mbps</td><td>操作系统：Debian GNU/Linux 12 (bookworm)
软件环境:docker,kubernetes
被测系统：物质科学本体-知识库V1.0</td></tr>
<tr><td>测试终端03</td><td>兼容机</td><td>CPU：2 x AMD EPYC 9654 (192 cores total)
GPU：NVIDIA A100 80GB * 8</td><td>操作系统：Ubuntu 20.04.6 LTS (Focal Fossa), x86_64
软件环境：python 3.7.16,</td></tr>
</table>

		(NVLink supported) 内存：64GB*8，ddr5 硬盘：960GB SSD + 3.84TB NVMe SSD + 20TB HDD * 4	docker 23.0.4，torch 1.13.0 等 被测系统：大模型轻量化系统 V1.0
测试终端04	兼容机	CPU： 1 x Intel Core i7-7700K (4 cores / 8 threads total) GPU： NVIDIA GeForce GTX 1080 Ti * 2 内存： 64GB DDR4 硬盘： 250GB NVMe SSD + 4TB HDD	操作系统：Ubuntu 22.04.5 LTS (Jammy Jellyfish) 软件环境： python3.11.5, torch 2.3.1, cmake 3.22.1, ggml 被测系统：大模型轻量化系统 V1.0
HNZM-SB-C-011 性能测试专用机	HUAWEI MateBook D15	CPU：12th Gen Intel（R）Core（TM）i7-1260P 2.10GHz 内存：16G 硬盘：512G	操作系统：Windows 11 性能测试工具：LoadRunner 2023
HNZM-SB-C-019 远程测试终端	惠普战66八代	CPU：Intel(R) Core(TM) 7 240H (2.50 GHz) 内存：32G 硬盘：1TB	操作系统：Windows 11 远程工具：腾讯会议 3.40.1、Xshell 7
网络环境	互联网		
物理环境	温度：15℃~20℃，湿度：40%~70%		
测试工具	LoadRunner 2023 设备编号 HNZM-SB-O-011		

3.1 测试工具介绍

3.1.1 LoadRunner

LoadRunner是一种预测系统行为和性能的负载测试工具。通过模拟多用户实施并发负载及实时性能监测的方式来确认和查找问题。

可适用于各种体系架构的自动负载测试，能预测系统行为并评估系统性能。

4 测试结果

4.1 病毒检查

测试项目		测试结果	说明
病毒检查	用360杀毒软件V7.0.0.1060版本，于2025.12.17进行病毒检查，现场检测环境应不含病毒。	无毒	/

4.2 质量测试结果

对本系统的功能性测试、可靠性测试采用等价类划分方法，本系统的效率测试采用黑盒测试法。

4.2.1 功能性测试结果

4.2.1.1 融合多学科知识与原理的认知智能系统功能性测试

测试项目	测试结果	说明
智能体基础交互	智能体请求可正常被调用。	/
物质科学推理10步响应准确率测试	第一次执行结果：使用物质科学10步推理测试数据集（625题）进行测试，实际推理的平均准确率为 63.04%； 第二次执行结果：使用物质科学10步推理测试数据集（625题）进行测试，实际推理的平均准确率为 63.52%。	/

4.2.1.2 物质科学本体-知识库功能性测试

测试项目	测试结果	说明
数据量统计测试	自然语言知识条目数量统计正确； 知识方程数量统计正确； 知识图谱五元组数量统计正确。	/
自然语言知识的结构规范度测试	使用测试脚本对自然语言知识库中的1269050条数据是否符合预定的字段规范进行检测，从而得到符合结构规范的自然语言知识条目数，最终通过公式计算得到自然语言知识的结构规范度为100%。	/

知识方程语法通过率测试	使用测试脚本对知识方程知识库中1568条数据进行语法解析检测，得到合法的知识方程条目数，最终通过公式计算得到知识方程的语法通过率为100%。	/
知识图谱的结构规范度测试	使用测试脚本对知识图谱知识库中1186457条数据进行空值、冗余、元组检测，从而得到合法的知识图谱元组数，最终通过公式计算得到知识图谱的结构规范度为100%。	/

4.2.1.3 大模型轻量化系统功能性测试

测试项目	测试结果	说明
4bit量化模型与基准模型对比测试	基于9个ChemBench4K数据集推理结果的平均准确率，4比特量化模型相对基准模型(16bit量化模型)的性能损失为3.78%。	/
8bit量化模型与基准模型对比测试	基于9个ChemBench4K数据集推理结果的平均准确率，8比特量化模型相对基准模型(16bit量化模型)的性能提升率为0.09%。	/
优化部署模型的准确率得分损伤测试	基于9个ChemBench4K数据集推理结果的准确率，优化部署模型对9个ChemBench4K数据集推理结果准确率得分损伤率都为0。	/
优化部署模型的AUROC得分损伤测试	基于9个ChemBench4K数据集推理结果中每道题ABCD选项的预测概率绘制ROC曲线，进行平均AUROC得分对比，优化部署模型的平均AUROC得分损伤率为0。	/

4.2.2 性能效率测试结果

4.2.2.1 融合多学科知识与原理的认知智能系统性能效率测试

测试项	测试结果	说明

物质科学推理10步响应时间测试	第一次执行结果：使用物质科学10步推理测试数据集（625题）进行测试，实际推理总耗时6小时57分33秒，平均推理耗时为 40.08 秒； 第二次执行结果：使用物质科学10步推理测试数据集（625题）进行测试，实际推理总耗时6小时3分53秒，平均推理耗时为 34.93 秒。	/

4.2.2.2 大模型轻量化系统性能效率测试

测试项	测试结果	说明
资源利用率测试	原始部署框架的平均显存利用率为87.48%，优化部署框架的平均显存利用率为99.00%，相比于原始框架显存利用率，优化部署模型显存利用率提升率为13.17%。	/
吞吐量测试	相对原始框架，优化部署框架单批次下连续生成长度为32的随机Token序列的解码任务的吞吐量为4.31t/s、吞吐量提升率为34.27%，单批次下连续生成长度为64的随机Token序列的解码任务的吞吐量为4.32t/s、吞吐量提升率为34.58%。	/
推理时间	相对原始框架，优化部署框架单批次下连续生成长度为32的随机Token序列的解码任务的平均推理时间为231.99ms/token、平均推理时间提升率为25.44%，单批次下连续生成长度为64的随机Token序列的解码任务的平均推理时间为231.28ms/token、平均推理时间提升率为25.67%。	/

4.2.3 可靠性测试结果

4.2.3.1 融合多学科知识与原理的认知智能系统可靠性测试

测试项目	测试结果	说明
成熟性	使用LoadRunner工具发送智能体请求，持续24小时，智能体请求响应正常，系统没有出现故障、服务中断的情况。	/

5 测试用例与记录

5.1 功能性测试

5.1.1 融合多学科知识与原理的认知智能系统功能性测试

5.1.1.1 智能体基础交互

编号	RHGN-01-01
测试用例目标	对“智能体基础交互”进行测试。
用例的初始化	/
实际结果评估标准	测试的每一个步骤都与期望结果一致。
测试过程	

1. 智能体请求输入问题，下图所示：

```
# 2. 验证 /chat
echo "步骤 2: 发送测试请求到 /chat"
TEST_MESSAGE="What is the molar weight (g/mol) of the molecule with the canonical SMILES representation 'CC(C(=O)NC(CC1(C=CC(=O)C=C1)F)C(=O)OC)NC(=O)C'
?\n\noption:\nA.407.400\nB.326.320\nC.400.400\nD.403.400"
RESPONSE=$(curl -sf --max-time $TIMEOUT \
  -H "Content-Type: application/json" \
  -d "{\"message\": \"${TEST_MESSAGE}\"}" \
  "${BASE_URL}/chat")
```

2. 通过.sh脚本调用智能体请求得到答案B，下图所示：

```
pyx@hopper-sxm-1:~/source/third-gen-interface$ . ./test_interface.sh
验证 API 接口结构，端口: 12345

期望的接口结构如下：

1. GET /healthz
   - 响应必须为 JSON 对象
   - 必须包含字段：{ "status": "healthy" }

2. POST /chat
   - 请求体必须为 JSON，包含字段：
     { "message": "<字符串>", "session_id": "<可选字符串>" }
   - 响应必须为 JSON 对象，包含以下两个字段：
     {
       "response": "<字符串，允许为空，但不能为 null>",
       "session_id": "<非空字符串>"
     }

开始验证...
步骤 1: 检查 /healthz 端点
通过: /healthz 返回了预期的健康状态

步骤 2: 发送测试请求到 /chat
{"response":"B","session_id":"1766371221"}
通过: /chat 响应包含符合要求的 'response' 和 'session_id' 字段

验证完成：所有接口结构符合预期。
```

特殊的规程说明	/
用例间的依赖关系	/

用例编写人员	吴涛	编写日期	2025.12.18
用例执行人员	吴涛	执行日期	2025.12.18~2025.12.19

5.1.1.2 物质科学推理10步响应准确率测试

编号	RHGN-02-01
测试用例目标	物质科学10步推理测试数据集的实际推理的平均准确率测试。
用例的初始化	采用物质科学领域多个典型评测数据集（包括ChemBench、SciBench、GAOKAO-Bench、ChemBench4K Benchmark等），从中选择多个领域 、多个类型的题目，进行数据质量评估与过滤，选出其中大于等于10步的长推理数据，形成测试数据集。
实际结果评估标准	测试的每一个步骤都与期望结果一致。

测试过程

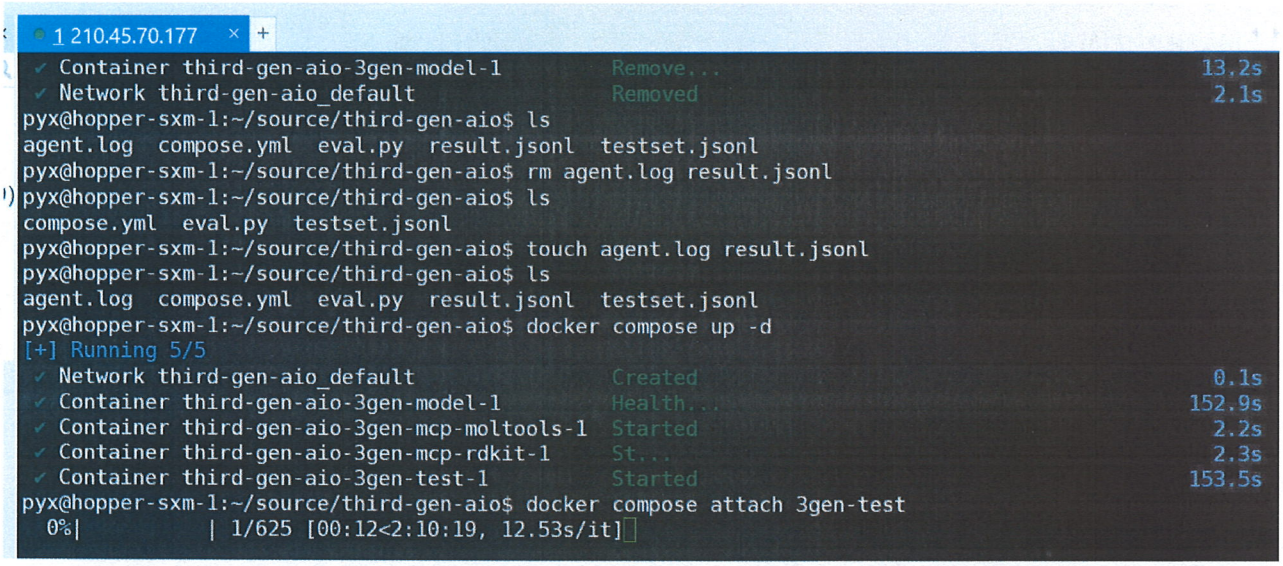
1. 测试数据集包含625题，如下所示：

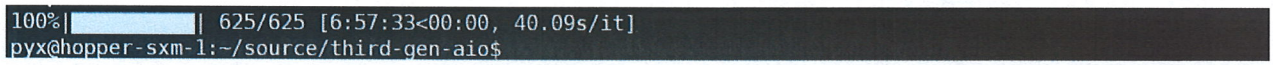
```
pyx@hopper-sxm-1:~/source/third-gen-aio$ wc -l testset.jsonl
625 testset.jsonl
```


{"question": "What are the common materials employed in the fabrication of CCCNC(=O)c1c(C(c2ccccc2)c2ccccc2)c2ccccc2n1CCN(C)C ?\n\n\noptions:\nA: Components used in the production of CCCNC(=O)c1c(C(c2ccccc2)c2ccccc2)c2ccccc2n1CCN(C)C can be selected from Cc1ccc(F)c(-c2c(F)cccc2F)n1, O, and O=[Mn](=O)(=O)[O-]. Reactions may transpire, and Cc1ccc(F)c(-c2c(F)cccc2F)n1.O=[Mn](=O)(=O)[O-].[K+].O>>CCCNC(=O)c1c(C(c2ccccc2)c2ccccc2)c2ccccc2n1CCN(C)C. might denote the reaction equations involved.\nB: Reactions may occur, and CC(C(=O)O)N1CCN(CC(=O)O)CCN(C(C)C(=O)O)CCN(C(C)C(=O)O)CC1.O=C(O)CN1CCN(C(COCc2cccc2)C(=O)O)CCN(C(COCc2ccccc2)C(=O)O)CCN(C(COCc2ccccc2)C(=O)O)CC1>>CCCNC(=O)c1c(C(c2ccccc2)c2ccccc2)c2ccccc2n1CCN(C)C. could serve as indicators of reaction equations. Ingredients utilized in the composition of CCCNC(=O)c1c(C(c2ccccc2)c2ccccc2)c2ccccc2n1CCN(C)C can be chosen from O=C(O)CN1CCN(C(COCc2cccccc2)C(=O)O)CCN(C(COCc2ccccc2)C(=O)O)CCN(C(COCc2ccccc2)C(=O)O)CC1.\nC: The occurrence of reactions is a possibility, with Cl.CCN=C=NCCCN(C)C.CN(C)CCn1c(C(=O)O)c(C(c2ccccc2)c2ccccc2)c2ccccc21.CCCN.O>ClCCl>CCCNC(=O)c1c(C(c2ccccc2)c2ccccc2)c2ccccc2n1CCN(C)C. potentially indicating reaction equations. Agents utilized in the fabrication of CCCNC(=O)c1c(C(c2ccccc2)c2ccccc2)c2ccccc2n1CCN(C)C can be selected from CCN=C=NCCCN(C)C and CN(C)CCn1c(C(=O)O)c(C(c2ccccc2)c2ccccc2)c2ccccc21.\nD: Agents used in the production of CCCNC(=O)c1c(C(c2ccccc2)c2ccccc2)c2ccccc2n1CCN(C)C can be selected from COC(=O)[C@H](C)N and Cc1cccc(C)c1N(CCCC(=O)O)C(=O)c1ccc(Cl)cc1. The potential for reactions to arise exists, with Cc1cccc(C)c1N(CCCC(=O)O)C(=O)c1ccc(Cl)cc1.COC(=O)[C@H](C)N>>CCCNC(=O)c1c(C(c2ccccc2)c2ccccc2)c2ccccc2n1CCN(C)C. potentially serving as indicators of reaction equations.", "answer": "C", "source": "chembench4k/Retrosynthesis_benchmark.jsonl", "is_mcq": true, "instruction": ""}
{"question": "Please answer the following questions: In substitutional solid solution alloys, the diffusion of solute atoms is achieved through which mechanism? \nA. Atomic exchange mechanism \nB. Interstitial diffusion mechanism \nC. Vacancy mechanism\nHere are some examples provided: [{'human': 'Please answer the following question: Which structural feature of flavone nucleus is unstable in alkaline solution? \nA. Ortho-diphenol hydroxyl group \nB. 3-hydroxyl \nC. 5-hydroxyl \nD. 7-hydroxyl', 'assistant': 'A'}, {'human': 'Please answer the following question: The delocalized π bond of the amino carbonate ion $\\left(\\mathrm{H}_{2}\\mathrm{NCO_{2}}\\right)$ is (   ). \nA. $\\Pi_{3}^{4}$ \nB. $\\Pi_{4}^{6}$ \nC. $\\Pi_{4}^{5}$ \nD. $\\Pi_{4}^{4}$', 'assistant': 'B'}, {'human': 'Please answer the following question: From the vertical cross-section diagram of the ternary phase diagram, one can (): \nA. Analyze the variation rules of phase compositions \nB. Analyze the alloy solidification process \nC. Use the lever rule to calculate the relative proportions of each phase', 'assistant': 'B'}]", "answer": "C", "source": "ChemEval/选择任务.json.jsonl", "is_mcq": true, "instruction": ""}
{"question": "How many rotatable bonds are there in the molecule with the IUPAC name '(2S)-2-[[(3S)-2-[(2S)-2-[[(2R)-2-amino-3-sulfanylpropyl]amino]-3-methylpentyl]-3,4-dihydro-1H-isoquinoline-3-carbonyl]amino]-4-methylsulfanylbutanoic acid;methyl (2S)-2-[[(3S)-2-[(2S)-2-[[(2R)-2-amino-3-sulfanylpropyl]amino]-3-methylpentyl]-3,4-dihydro-1H-isoquinoline-3-carbonyl]amino]-4-methylsulfanylbutanoate'?\n\noption:\nA.24\nB.21\nC.22\nD.29", "answer": "D", "source": "SciKnowEval/molecule_structure_prediction_output.jsonl", "is_mcq": true, "instruction": ""}
{"question": "What is a typical use for acidic salts?\n\nOptions:\nA. Catalysts\nB. Oxidizers\nC. Strong reducing agents\nD. Weak bases.", "answer": "A", "source": "chembench_569/reasoning_chembench.jsonl", "is_mcq": true, "instruction": ""}
{"question": "Could you tell me what the solvent medium for the reaction linked with the SMILES O=C(Nc1ccc(OC(F)(F)F)cc1)c1cnc(N2CC[C@@H](O)C2)c(Cl)c1.CC1(C)OB(c2csc(CO)c2)OC1(C)C.O=C([O-])[O-].[Na+].[Na+]>>O=C(Nc1ccc(OC(F)(F)F)cc1)c1cnc(N2CC[C@@H](O)C2)c(-c2csc(CO)c2)c1. is?\n\noptions:\nA: The solvents CO are integral to this chemical reaction.\nB: The chemical solvents CCO are used in this reaction.\nC: The reaction is conducted using solvents ClCCl.\nD: Solvents COCCOC are employed in this chemical reaction.", "answer": "D", "source": "chembench4k/Solvent_Prediction_benchmark.jsonl", "is_mcq": true, "instruction": ""}

2. 第一次执行：

2.1. 执行命令docker compose attach 3gen-test进行推理，下图所示：





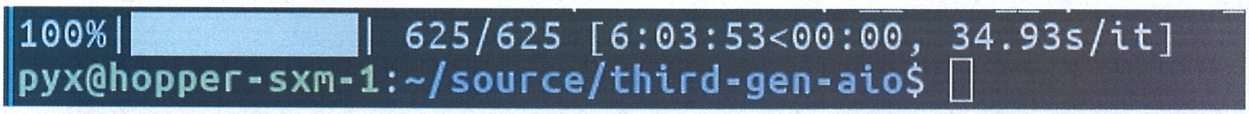
2.2. 根据推理结果统计推理正确的题目数量，执行命令uv run python eval.py result.jsonl，统计结果下图所示：



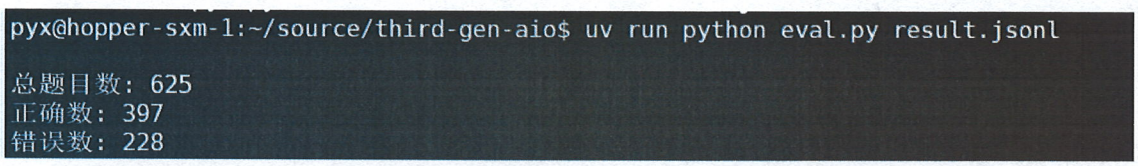
2.3. 物质科学10步推理测试数据集的实际推理的平均准确率= 394/625 * 100% = 63.04%；

3. 第二次执行：

3.1. 执行命令docker compose attach 3gen-test进行推理，下图所示：



3.2. 根据推理结果统计推理正确的题目数量，执行命令uv run python eval.py result.jsonl，统计结果下图所示：



3.3. 物质科学10步推理测试数据集的实际推理的平均准确率= 397/625 * 100% = 63.52%。

特殊的规程说明	/		
用例间的依赖关系	/		
用例编写人员	吴涛	编写日期	2025.12.18

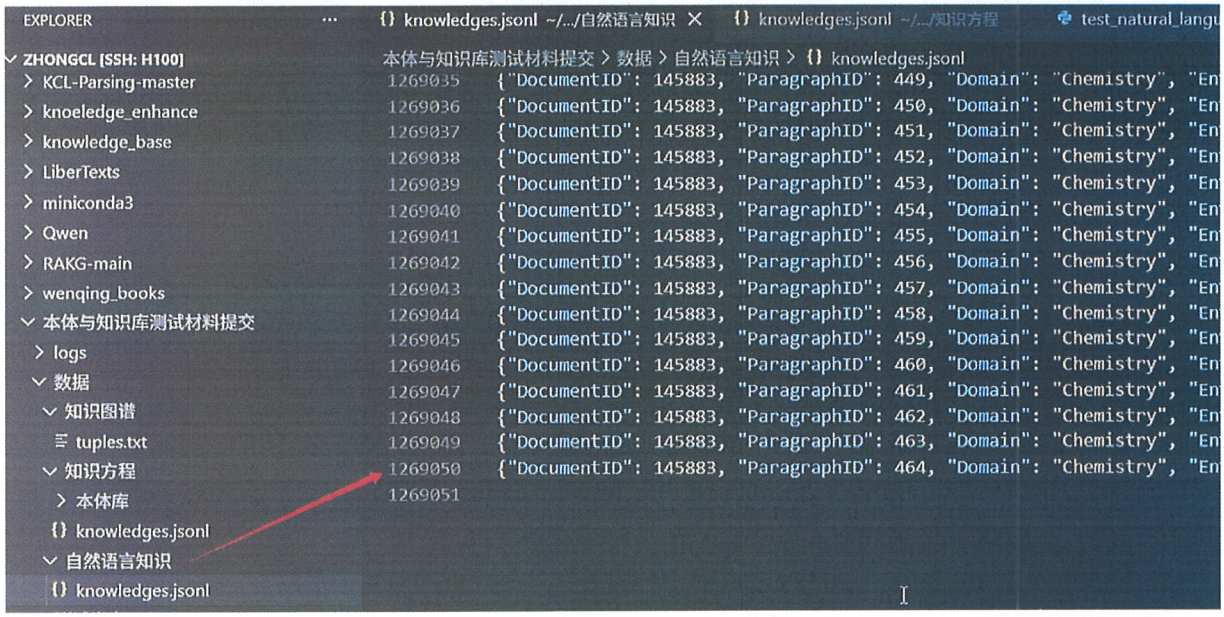
用例执行人员	吴涛	执行日期	2025.12.18~2025.12.19 2025.12.23

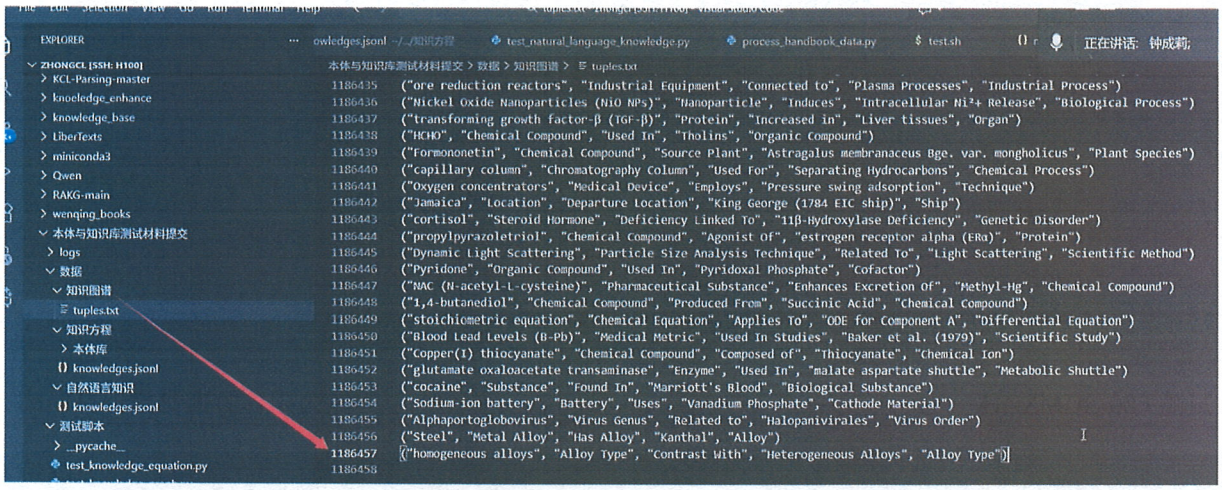
5.1.2 物质科学本体-知识库功能性测试

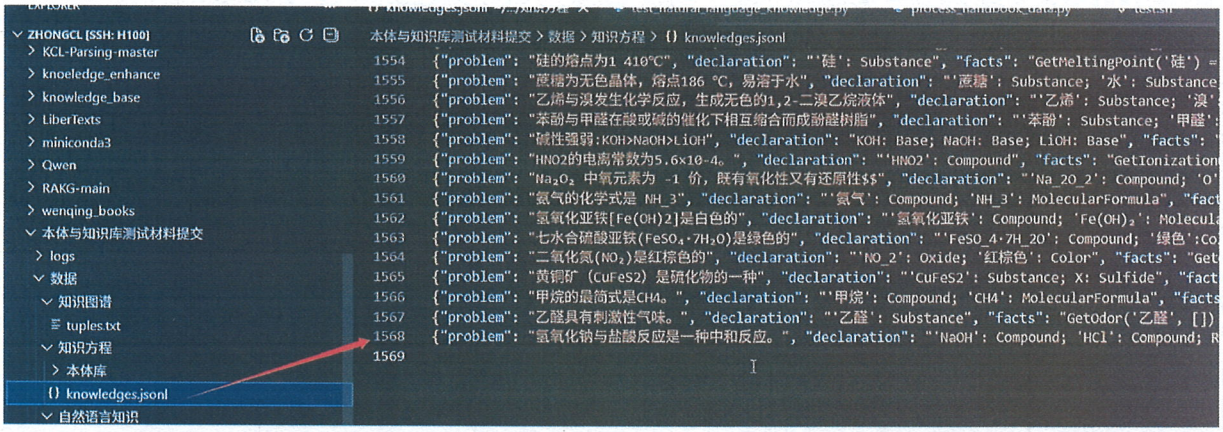
5.1.2.1 数据量统计测试

编号	ZSGN-01-01
测试用例目标	使用自动化 Python 脚本对三类知识库对应的文件进行扫描与解析，分别统计： 自然语言知识条目数量 知识方程数量 知识图谱五元组数量
用例的初始化	/
实际结果评估标准	测试的每一个步骤都与期望结果一致。
测试过程	

1. 数据集包括：自然语言知识、知识图谱、知识方程三类知识库，分别包括1269050条、1186457条、1568条，下图所示：







2. 使用自动化 Python 脚本对三类知识库对应的文件进行扫描与解析

2.1. 自然语言知识条目数量为1269050条，与实际数量一致，下图所示：

```
(chem) zhongcl@hopper-sxm-4:~$ bash /public/home/zhongcl/本体与知识库测试材料提交/测试脚本/test.sh
正在处理自然语言知识文件：本体与知识库测试材料提交/数据/自然语言知识
2025-12-19 14:01:55,366 - test_natural_language_knowledge - INFO - 开始处理文件夹：本体与知识库测试材料提交/数据/自然语言知识
2025-12-19 14:01:55,367 - test_natural_language_knowledge - INFO - 在 本体与知识库测试材料提交/数据/自然语言知识 中找到 1 个jsonl文件
2025-12-19 14:01:55,367 - test_natural_language_knowledge - INFO - 开始处理 1 个文件，使用 128 个线程
2025-12-19 14:02:29,478 - test_natural_language_knowledge - INFO - 文件处理完成，耗时：34.11 秒
2025-12-19 14:02:29,479 - test_natural_language_knowledge - INFO - 处理统计：文件数=1，行数=1269050，失败行数=0
2025-12-19 14:02:29,479 - test_natural_language_knowledge - INFO - 自然语言知识处理完成：本体与知识库测试材料提交/数据/自然语言知识
2025-12-19 14:02:29,479 - test_natural_language_knowledge - INFO - 总计数量：1269050
2025-12-19 14:02:29,479 - test_natural_language_knowledge - INFO - 去重后数量：1269050
2025-12-19 14:02:29,479 - test_natural_language_knowledge - INFO - 结构有效数量：1269050
2025-12-19 14:02:29,479 - test_natural_language_knowledge - INFO - 有效占比：100.00%
```

2.2. 知识图谱五元组条目数量为11864457条，与实际数量一致，下图所示：

```
[2/3] 计算知识图谱指标...
正在处理知识图谱文件：本体与知识库测试材料提交/数据/知识图谱
2025-12-19 14:02:30,521 - test_knowledge_graph - INFO - 开始处理知识图谱文件夹：本体与知识库测试材料提交/数据/知识图谱
2025-12-19 14:02:30,521 - test_knowledge_graph - INFO - 在 本体与知识库测试材料提交/数据/知识图谱 中找到 1 个txt文件
2025-12-19 14:02:30,522 - test_knowledge_graph - INFO - 开始处理 1 个文件，使用 128 个线程
2025-12-19 14:02:40,451 - test_knowledge_graph - INFO - 文件处理完成，耗时：9.93 秒
2025-12-19 14:02:40,451 - test_knowledge_graph - INFO - 处理统计：文件数=1，行数=1186457，失败行数=0
2025-12-19 14:02:40,685 - test_knowledge_graph - INFO - 知识图谱处理完成：本体与知识库测试材料提交/数据/知识图谱
2025-12-19 14:02:40,685 - test_knowledge_graph - INFO - 总计数量：1186457
2025-12-19 14:02:40,685 - test_knowledge_graph - INFO - 去重后数量：1186457
2025-12-19 14:02:40,685 - test_knowledge_graph - INFO - 结构有效数量：1186457
2025-12-19 14:02:40,685 - test_knowledge_graph - INFO - 去重率：0.00%
2025-12-19 14:02:40,685 - test_knowledge_graph - INFO - 有效占比：100.00%
```

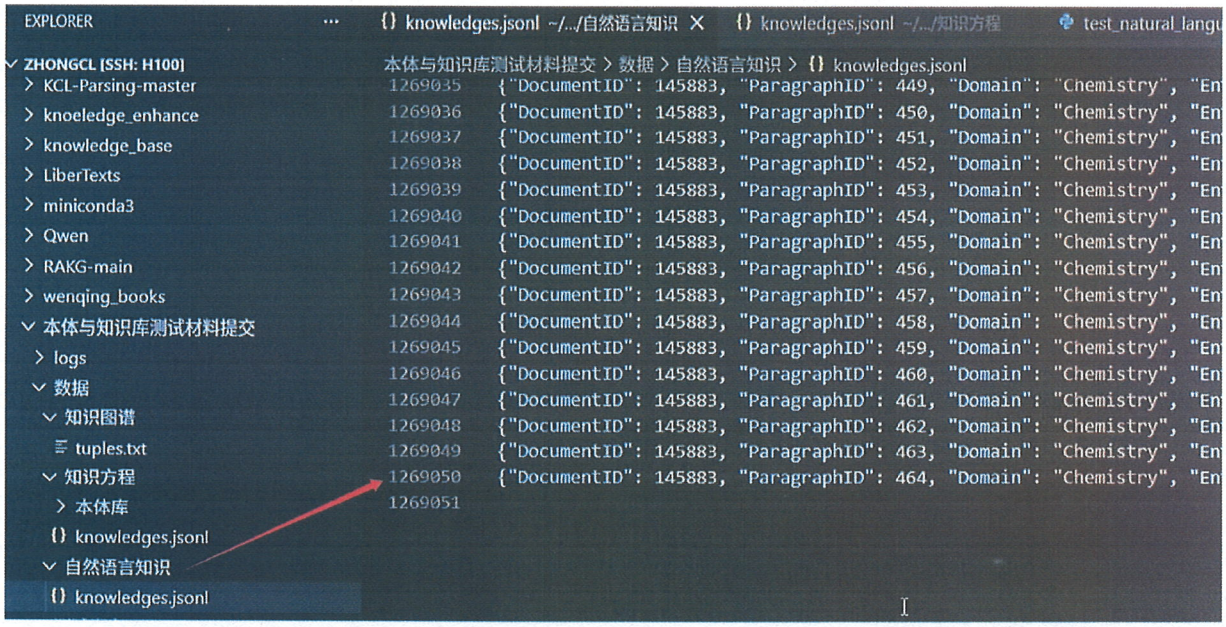
2.3. 知识方程条目数量为1568，与实际数量一致，下图所示：

```
[3/3] 计算知识方程指标...
正在处理知识方程文件：本体与知识库测试材料提交/数据/知识方程/knowledges.jsonl
2025-12-19 14:02:41,170 - test_knowledge_equation - INFO - 开始处理知识方程验证
加载概念文件本体与知识库测试材料提交/数据/知识方程/本体库/concepts.json，一共有313
加载算子文件本体与知识库测试材料提交/数据/知识方程/本体库/operators.json，一共有387
处理后，一共有313个概念
处理后，一共有387个算子
2025-12-19 14:02:41,185 - test_knowledge_equation - INFO - 加载知识方程文件本体与知识库测试材料提交/数据/知识方程/knowledges.jsonl，一共有1568
2025-12-19 14:02:41,257 - test_knowledge_equation - INFO - 知识方程文件处理完成
总计数量：1568
去重后数量：1568
结构有效数量：1568
结果已保存到：本体与知识库测试材料提交/metrics_results.json
```

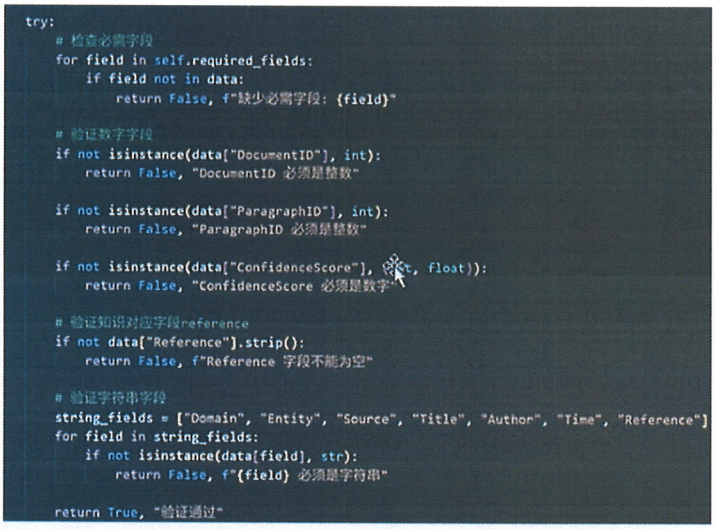
特殊的规程说明	/		
用例间的依赖关系	/		
用例编写人员	吴涛	编写日期	2025.12.18
用例执行人员	吴涛	执行日期	2025.12.18~2025.12.19

5.1.2.2 自然语言知识的结构规范度测试

编号	ZSGN-02-01
测试用例目标	自然语言知识的结构规范度测试。
用例的初始化	/
实际结果评估标准	测试的每一个步骤都与期望结果一致。
测试过程	
1. 数据集：自然语言知识库，包括1269050条，下图所示：	



2. 使用自动化 Python 脚本检测每条自然语言知识是否符合预定的字段规范，即是否包含且仅包含预定义字段："DocumentID"，"ParagraphID"，"Domain"，"Entity"，"Source"，"Title"，"Author"，"Time"，"Reference"，"ConfidenceScore"，下图所示：



```
try:
    # 检查必需字段
    for field in self.required_fields:
        if field not in data:
            return False, f"缺少必需字段: {field}"

    # 验证数字字段
    if not isinstance(data["DocumentID"], int):
        return False, "DocumentID 必须是整数"

    if not isinstance(data["ParagraphID"], int):
        return False, "ParagraphID 必须是整数"

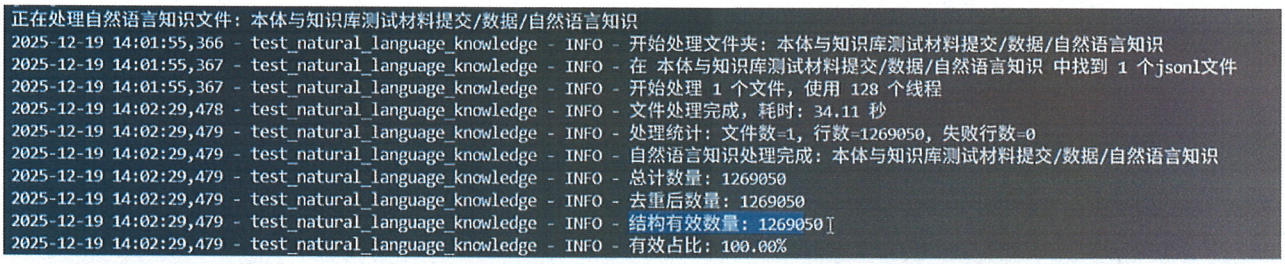
    if not isinstance(data["ConfidenceScore"], (int, float)):
        return False, "ConfidenceScore 必须是数字"

    # 验证知识对应字段reference
    if not data["Reference"].strip():
        return False, f"Reference 字段不能为空"

    # 验证字符串字段
    string_fields = ["Domain", "Entity", "Source", "Title", "Author", "Time", "Reference"]
    for field in string_fields:
        if not isinstance(data[field], str):
            return False, f"{field} 必须是字符串"

    return True, "验证通过"
```

3. 最终得到自然语言知识的结构规范度为100%，输出结果如下：



```
正在处理自然语言知识文件：本体与知识库测试材料提交/数据/自然语言知识
2025-12-19 14:01:55,366 - test_natural_language_knowledge - INFO - 开始处理文件夹：本体与知识库测试材料提交/数据/自然语言知识
2025-12-19 14:01:55,367 - test_natural_language_knowledge - INFO - 在 本体与知识库测试材料提交/数据/自然语言知识 中找到 1 个jsonl文件
2025-12-19 14:01:55,367 - test_natural_language_knowledge - INFO - 开始处理 1 个文件，使用 128 个线程
2025-12-19 14:02:29,478 - test_natural_language_knowledge - INFO - 文件处理完成，耗时：34.11 秒
2025-12-19 14:02:29,479 - test_natural_language_knowledge - INFO - 处理统计：文件数=1，行数=1269050，失败行数=0
2025-12-19 14:02:29,479 - test_natural_language_knowledge - INFO - 自然语言知识处理完成：本体与知识库测试材料提交/数据/自然语言知识
2025-12-19 14:02:29,479 - test_natural_language_knowledge - INFO - 总计数量：1269050
2025-12-19 14:02:29,479 - test_natural_language_knowledge - INFO - 去重后数量：1269050
2025-12-19 14:02:29,479 - test_natural_language_knowledge - INFO - 结构有效数量：1269050
2025-12-19 14:02:29,479 - test_natural_language_knowledge - INFO - 有效占比：100.00%
```

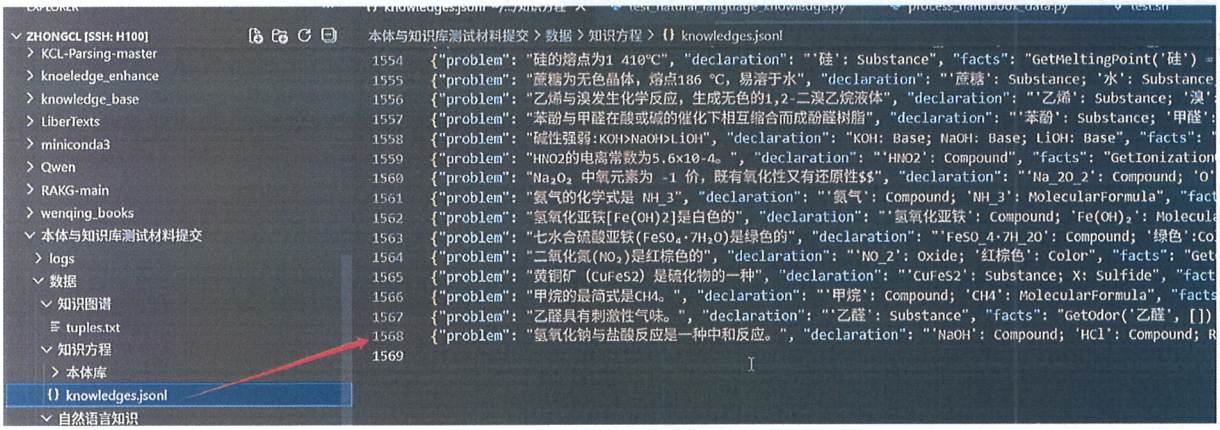
特殊的规程说明	自然语言知识结构规范度=符合结构规范的自然语言知识条目数/自然语言知识条目

	总数*100%		
用例间的依赖关系	/		
用例编写人员	吴涛	编写日期	2025.12.18
用例执行人员	吴涛	执行日期	2025.12.18~2025.12.19

5.1.2.3 知识方程语法通过率测试

编号	ZSGN-03-01
测试用例目标	知识方程语法通过率测试。
用例的初始化	/
实际结果评估标准	测试的每一个步骤都与期望结果一致。
测试过程	

1. 数据集：知识方程知识库，包括1568条，下图所示：



2. 使用自动化 Python 脚本对每条知识方程进行语法解析，检查其使用的 算子（Operator） 和 概念（Concept） 是否全部来自给定的本体库：(1)算子合法性（是否在算子词表中）、(2)概念合法性（是否在概念词表中）、(3)参数类型是否匹配（输入、输出概念是否符合定义），下图所示：

```
caculate_grammar_pass_rate(self):
"""统计知识方程数据的语法通过率"""
legal_datas = []
ilegal_datas = []
uniq_datas = []

for index, data in enumerate(self.knowledge_equation_data):
    problem = data["problem"]
    # 去重
    if problem in uniq_datas:
        continue
    uniq_datas.append(problem)
    declarations = data["declaration"].split(';')
    facts = data["facts"].split(';')

    # 检查概念是否合法
    concept_legality, variable_concept = self.test_concept_Legality(declarations=declarations)
    if not concept_legality:
        ilegal_datas.append(data)
        continue

    # 检查算子是否合法
    operator_legality, operator_parameter = self.test_operator_Legality(facts=facts)
    if not operator_legality:
        ilegal_datas.append(data)
        continue

    # 检查算子的输入和输出类型是否合法
    operator_parameter_legality = self.test_operator_parameter_type_Legality(variable_concept,
    if not operator_parameter_legality:
        ilegal_datas.append(data)
        continue
```

3. 最终统计得到知识方程的语法通过率为100%，输出结果下图所示：

```
[3/3] 计算知识方程指标...
正在处理知识方程文件：本体与知识库测试材料提交/数据/知识方程/knowledges.jsonl
2025-12-19 14:02:41,170 - test_knowledge_equation - INFO - 开始处理知识方程验证
加载概念文件本体与知识库测试材料提交/数据/知识方程/本体库/concepts.json，一共有313
加载算子文件本体与知识库测试材料提交/数据/知识方程/本体库/operators.json，一共有387
处理后，一共有313个概念
处理后，一共有387个算子
2025-12-19 14:02:41,185 - test_knowledge_equation - INFO - 加载知识方程文件本体与知识库测试材料提交/数据/知识方程/knowledges.jsonl，一共有1568
2025-12-19 14:02:41,257 - test_knowledge_equation - INFO - 知识方程文件处理完成
总计数量：1568
去重后数量：1568
结构有效数量：1568
结果已保存到：本体与知识库测试材料提交/metrics_results.json
```

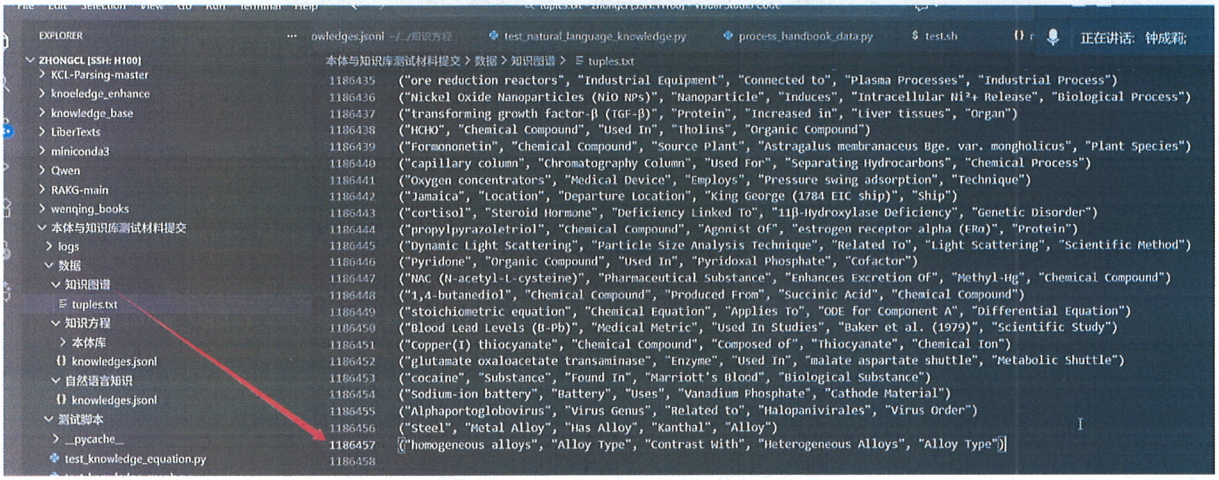
特殊的规程说明	知识方程的语法通过率=合法的知识方程条目数/知识方程条目总数 * 100%		
用例间的依赖关系	/		
用例编写人员	吴涛	编写日期	2025.12.18
用例执行人员	吴涛	执行日期	2025.12.18~2025.12.19

5.1.2.4 知识图谱的结构规范度测试

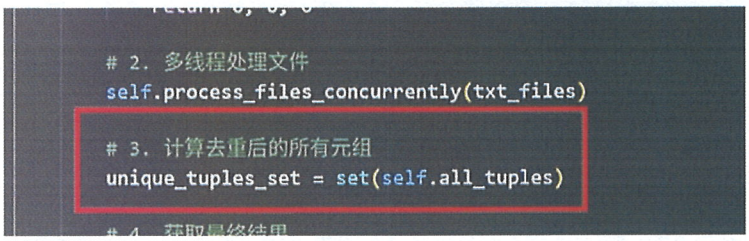
编号	ZSGN-04-01
测试用例目标	知识图谱的结构规范度测试。

用例的初始化	/
实际结果评估标准	测试的每一个步骤都与期望结果一致。
测试过程	

1. 数据集包括：知识图谱知识库，包括1186457条，下图所示：



2. 使用自动化 Python 脚本对知识图谱中的每个元组执行以下检查：(1)空值检测：检查是否存在空字段、(2)冗余检测：检查是否存在重复五元组、(3)元组检测：检查每个元组的元素数量是否为5，下图所示：



```python
def is_valid_quintuple(self, tuple_data: Tuple[str, ...]) -> Tuple[bool, str]:
    """
    检查是否为合法的五元组

    Args:
        tuple_data: 元组数据

    Returns:
        (是否有效, 错误信息)
    """
    try:
        # 检查元素数量
        if len(tuple_data) != 5:
            return False, f"元素数量不正确: 期望5个, 实际{len(tuple_data)}个"

        # 检查是否有空值
        for i, item in enumerate(tuple_data, 1):
            if not item or str(item).strip() == "":
                return False, f"第{i}个元素为空"

        return True, "验证通过"

    except Exception as e:
        return False, f"验证过程中发生异常: {str(e)}"
```

3. 最终得到知识图谱的结构规范度为100%，输出结果下图所示：

```
2025-12-19 14:02:30,521 - test_knowledge_graph - INFO - 开始处理知识图谱文件夹: 本体与知识库测试材料提交/数据/知识图谱
2025-12-19 14:02:30,521 - test_knowledge_graph - INFO - 在 本体与知识库测试材料提交/数据/知识图谱 中找到 1 个txt文件
2025-12-19 14:02:30,522 - test_knowledge_graph - INFO - 开始处理 1 个文件，使用 128 个线程
2025-12-19 14:02:40,451 - test_knowledge_graph - INFO - 文件处理完成，耗时: 9.93 秒
2025-12-19 14:02:40,451 - test_knowledge_graph - INFO - 处理统计: 文件数=1, 行数=1186457, 失败行数=0
2025-12-19 14:02:40,685 - test_knowledge_graph - INFO - 知识图谱处理完成: 本体与知识库测试材料提交/数据/知识图谱
2025-12-19 14:02:40,685 - test_knowledge_graph - INFO - 总计数量: 1186457
2025-12-19 14:02:40,685 - test_knowledge_graph - INFO - 去重后数量: 1186457
2025-12-19 14:02:40,685 - test_knowledge_graph - INFO - 结构有效数量: 1186457
2025-12-19 14:02:40,685 - test_knowledge_graph - INFO - 去重率: 0.00%
2025-12-19 14:02:40,685 - test_knowledge_graph - INFO - 有效占比: 100.00%
```

特殊的规程说明	知识图谱结构规范度=合法的知识图谱元组数/知识图谱元组总数*100%		
用例间的依赖关系	/		
用例编写人员	吴涛	编写日期	2025.12.18
用例执行人员	吴涛	执行日期	2025.12.18~2025.12.19

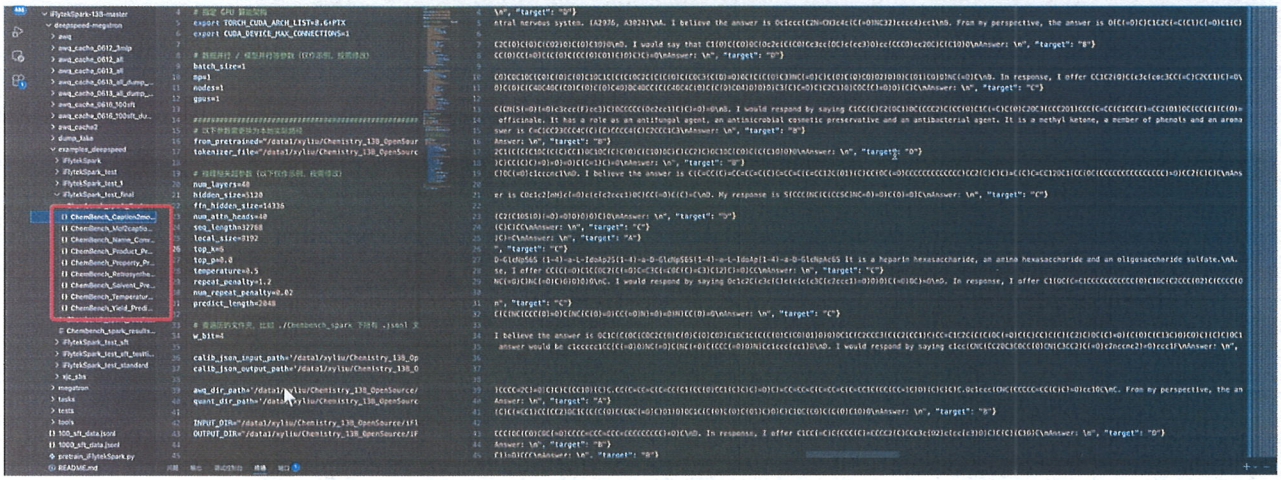
5.1.3 大模型轻量化系统功能性测试

5.1.3.1 4bit 量化模型与基准模型对比测试

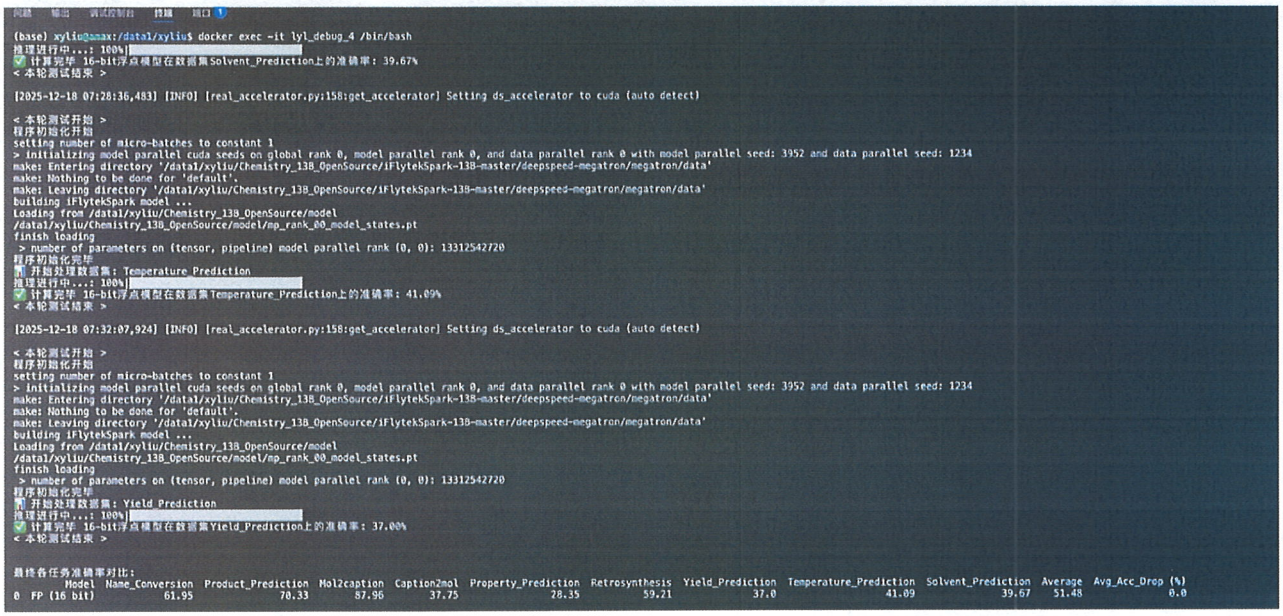
编号	DMXGN-01-01
测试用例目标	基于ChemBench4K数据集的评估结果，4比特量化模型性能损失不超过6%。
用例的初始化	/
实际结果评估标准	测试的每一个步骤都与期望结果一致。

测试过程

1. 测试数据集：包含9个ChemBench4K数据集文件，每个文件中包含选择题和对应正确答案信息，下图所示：



2. 执行基准模型(16bit量化模型)脚本，通过模型推理答案，然后与正确答案比较得到9个ChemBench4K数据集推理结果的准确率，最终计算得到基准模型的平均准确率，输出结果下图所示：



```
(base) xyliu@amax:/data1/xyliu$ docker exec -it lyl_debug_4 /bin/bash
推理进行中...: 100%|
✅ 计算完毕 16-bit浮点模型在数据集Solvent_Prediction上的准确率: 39.67%
< 本轮测试结束 >

[2025-12-18 07:28:36,483] [INFO] [real_accelerator.py:158:get_accelerator] Setting ds_accelerator to cuda (auto detect)

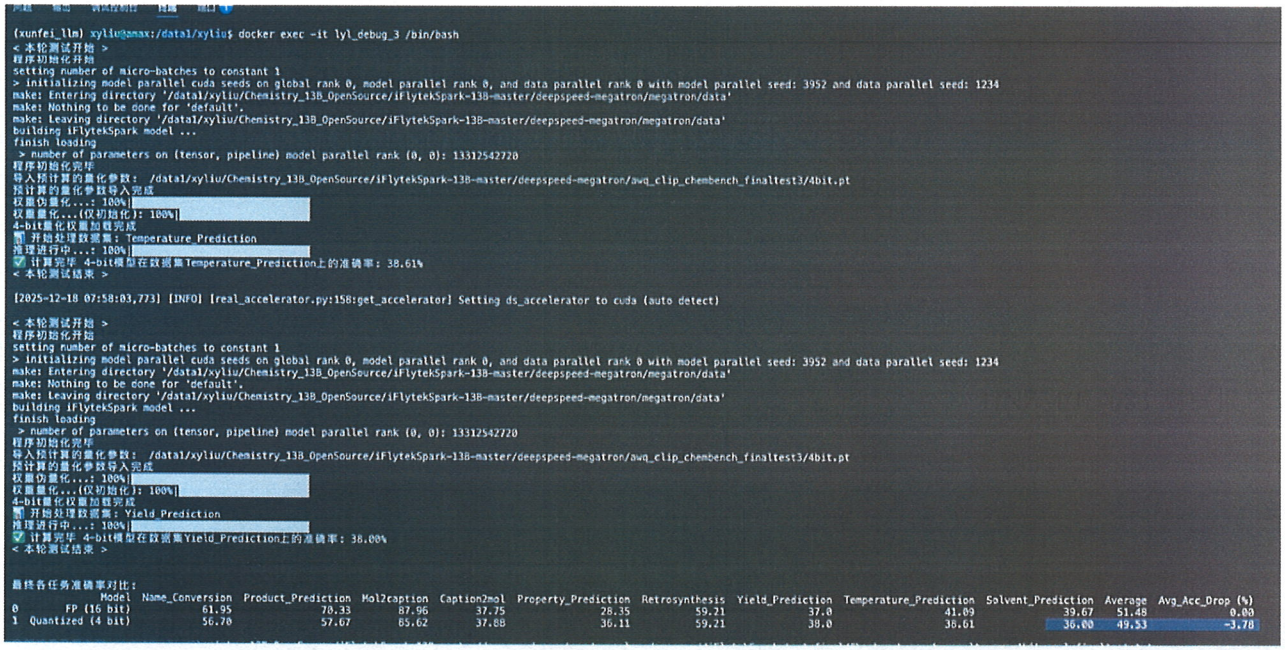
< 本轮测试开始 >
程序初始化开始
setting number of micro-batches to constant 1
> initializing model parallel cuda seeds on global rank 0, model parallel rank 0, and data parallel rank 0 with model parallel seed: 3952 and data parallel seed: 1234
make: Entering directory '/data1/xyliu/Chemistry_13B_OpenSource/iFlytekSpark-13B-master/deepspeed-megatron/megatron/data'
make: Nothing to be done for 'default'.
make: Leaving directory '/data1/xyliu/Chemistry_13B_OpenSource/iFlytekSpark-13B-master/deepspeed-megatron/megatron/data'
building iFlytekSpark model ...
Loading from /data1/xyliu/Chemistry_13B_OpenSource/model
/data1/xyliu/Chemistry_13B_OpenSource/model/mp_rank_00_model_states.pt
finish loading
 > number of parameters on (tensor, pipeline) model parallel rank (0, 0): 13312542720
程序初始化完毕
📊 开始处理数据集: Temperature_Prediction
推理进行中...: 100%|
✅ 计算完毕 16-bit浮点模型在数据集Temperature_Prediction上的准确率: 41.09%
< 本轮测试结束 >

[2025-12-18 07:32:07,924] [INFO] [real_accelerator.py:158:get_accelerator] Setting ds_accelerator to cuda (auto detect)

< 本轮测试开始 >
程序初始化开始
setting number of micro-batches to constant 1
> initializing model parallel cuda seeds on global rank 0, model parallel rank 0, and data parallel rank 0 with model parallel seed: 3952 and data parallel seed: 1234
make: Entering directory '/data1/xyliu/Chemistry_13B_OpenSource/iFlytekSpark-13B-master/deepspeed-megatron/megatron/data'
make: Nothing to be done for 'default'.
make: Leaving directory '/data1/xyliu/Chemistry_13B_OpenSource/iFlytekSpark-13B-master/deepspeed-megatron/megatron/data'
building iFlytekSpark model ...
Loading from /data1/xyliu/Chemistry_13B_OpenSource/model
/data1/xyliu/Chemistry_13B_OpenSource/model/mp_rank_00_model_states.pt
finish loading
 > number of parameters on (tensor, pipeline) model parallel rank (0, 0): 13312542720
程序初始化完毕
📊 开始处理数据集: Yield_Prediction
推理进行中...: 100%|
✅ 计算完毕 16-bit浮点模型在数据集Yield_Prediction上的准确率: 37.00%
< 本轮测试结束 >


最终各任务准确率对比:
      Model  Name_Conversion  Product_Prediction  Mol2caption  Caption2mol  Property_Prediction  Retrosynthesis  Yield_Prediction  Temperature_Prediction  Solvent_Prediction  Average  Avg_Acc_Drop (%)
0  FP (16 bit)            61.95               70.33        87.96        37.75                28.35           59.21              37.0                   41.09               39.67    51.48             0.0
```

3. 执行4bit量化模型脚本，通过模型推理答案，然后与正确答案比较得到9个数据集推理结果的准确率，最终计算得到4bit量化模型推理结果的平均准确率，输出结果下图所示：



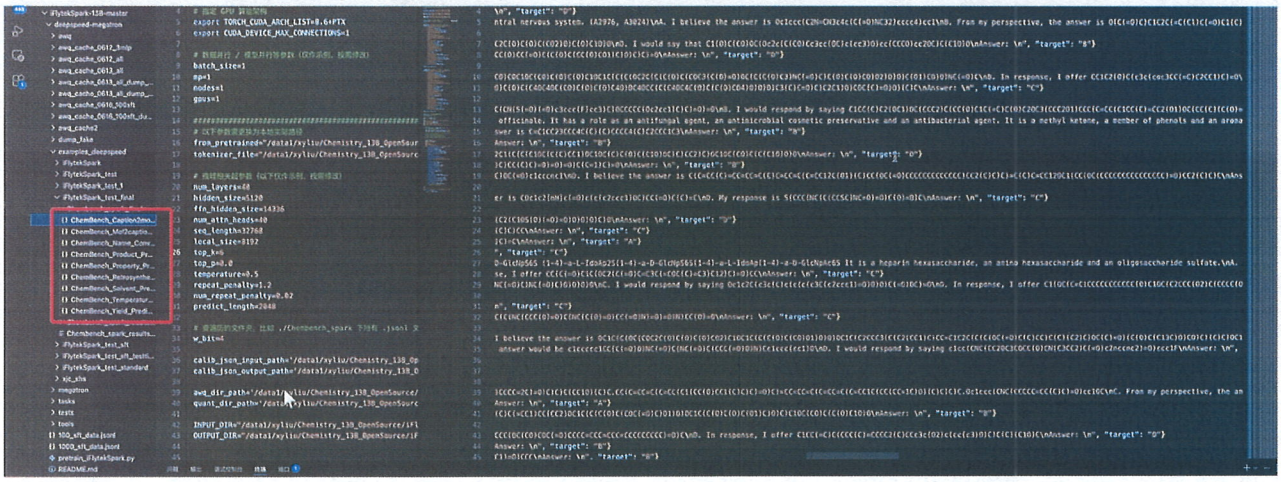
4. 性能损失/提升率 = (4比特量化模型平均准确率-基准模型平均准确率)/基准模型平均准确率 * 100% = (49.53-51.48) /51.48 * 100% = -3.78%，即4比特量化模型相对基准模型(16bit量化模型)的性能损失为3.78%。

特殊的规程说明	计算公式： 性能损失/提升率 = (4比特量化模型平均准确率-基准模型平均准确率)/基准模型平均准确率 * 100%		
用例间的依赖关系	/		
用例编写人员	吴涛	编写日期	2025.12.18
用例执行人员	吴涛	执行日期	2025.12.18~2025.12.19

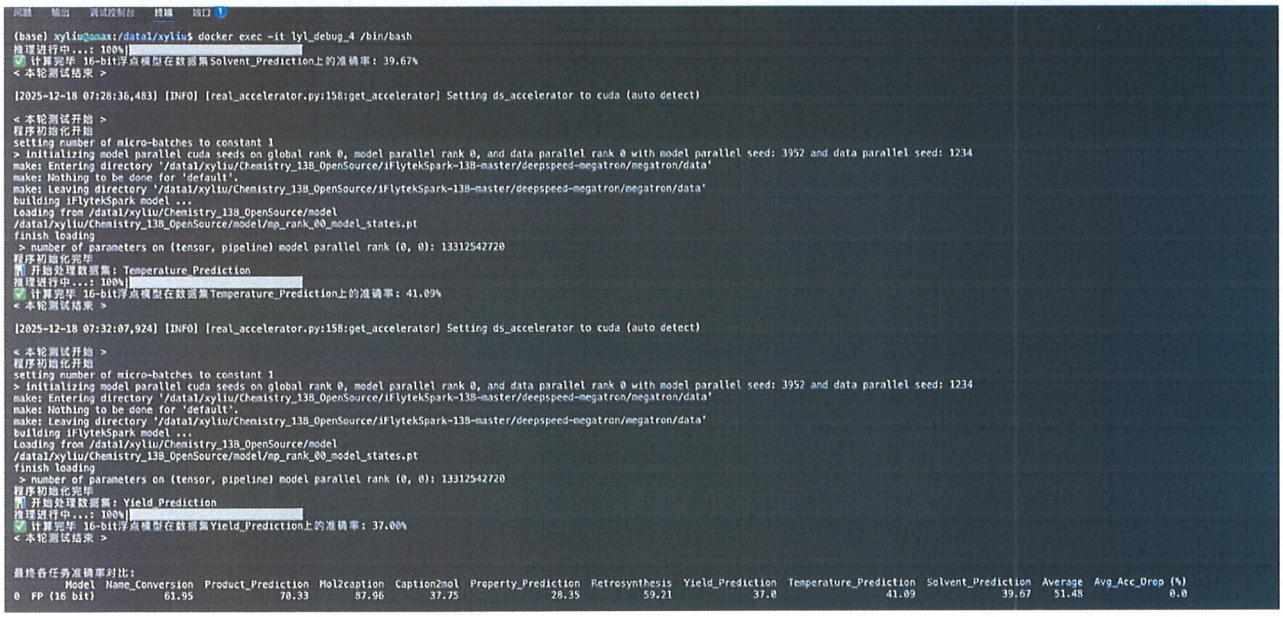
5.1.3.2 8bit 量化模型与基准模型对比测试

编号	DMXGN-02-01
测试用例目标	基于ChemBench4K数据集的评估结果，8比特量化模型性能损失不超过6%。
用例的初始化	/
实际结果评估标准	测试的每一个步骤都与期望结果一致。
测试过程	
1. 测试数据集：包含9个ChemBench4K数据集文件，每个文件中包含选择题和对应正确答案信息，下图所	

示：



2. 执行基准模型(16bit量化模型)脚本，通过模型推理答案，然后与正确答案比较得到9个ChemBench4K数据集推理结果的准确率，最终计算得到基准模型的平均准确率，输出结果下图所示：



```
(base) xyliu@amax:/data1/xyliu$ docker exec -it lyl_debug_4 /bin/bash
推理进行中...: 100%|
✅ 计算完毕 16-bit浮点模型在数据集Solvent_Prediction上的准确率: 39.67%
< 本轮测试结束 >

[2025-12-18 07:28:36,483] [INFO] [real_accelerator.py:158:get_accelerator] Setting ds_accelerator to cuda (auto detect)

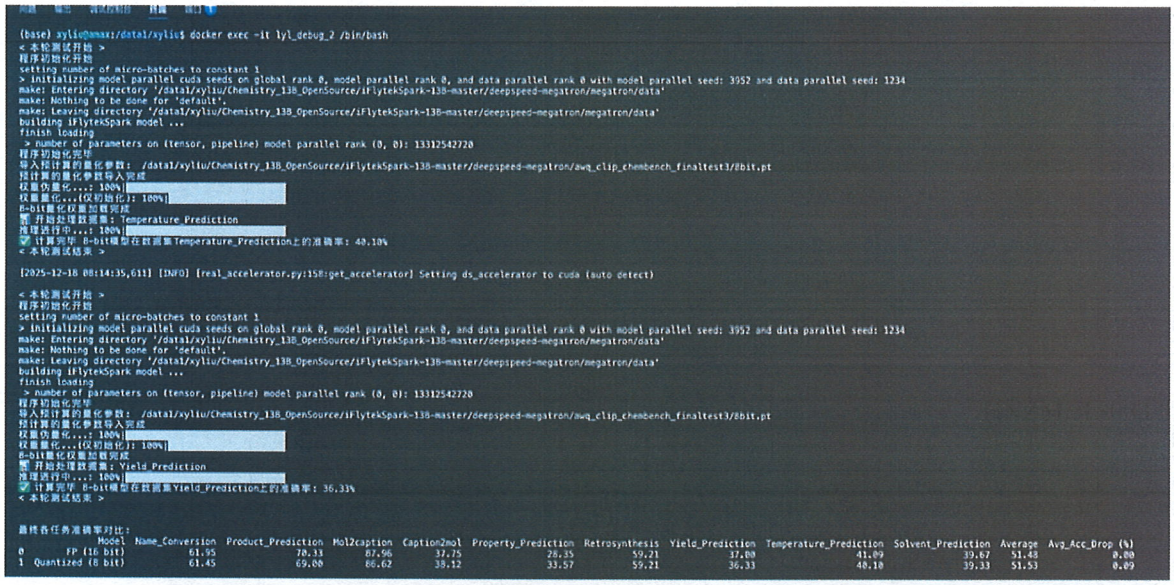
< 本轮测试开始 >
程序初始化开始
setting number of micro-batches to constant 1
> initializing model parallel cuda seeds on global rank 0, model parallel rank 0, and data parallel rank 0 with model parallel seed: 3952 and data parallel seed: 1234
make: Entering directory '/data1/xyliu/Chemistry_13B_OpenSource/iFlytekSpark-13B-master/deepspeed-megatron/megatron/data'
make: Nothing to be done for 'default'.
make: Leaving directory '/data1/xyliu/Chemistry_13B_OpenSource/iFlytekSpark-13B-master/deepspeed-megatron/megatron/data'
building iFlytekSpark model ...
Loading from /data1/xyliu/Chemistry_13B_OpenSource/model
/data1/xyliu/Chemistry_13B_OpenSource/model/mp_rank_00_model_states.pt
finish loading
 > number of parameters on (tensor, pipeline) model parallel rank (0, 0): 13312542720
程序初始化完毕
📊 开始处理数据集: Temperature_Prediction
推理进行中...: 100%|
✅ 计算完毕 16-bit浮点模型在数据集Temperature_Prediction上的准确率: 41.09%
< 本轮测试结束 >

[2025-12-18 07:32:07,924] [INFO] [real_accelerator.py:158:get_accelerator] Setting ds_accelerator to cuda (auto detect)

< 本轮测试开始 >
程序初始化开始
setting number of micro-batches to constant 1
> initializing model parallel cuda seeds on global rank 0, model parallel rank 0, and data parallel rank 0 with model parallel seed: 3952 and data parallel seed: 1234
make: Entering directory '/data1/xyliu/Chemistry_13B_OpenSource/iFlytekSpark-13B-master/deepspeed-megatron/megatron/data'
make: Nothing to be done for 'default'.
make: Leaving directory '/data1/xyliu/Chemistry_13B_OpenSource/iFlytekSpark-13B-master/deepspeed-megatron/megatron/data'
building iFlytekSpark model ...
Loading from /data1/xyliu/Chemistry_13B_OpenSource/model
/data1/xyliu/Chemistry_13B_OpenSource/model/mp_rank_00_model_states.pt
finish loading
 > number of parameters on (tensor, pipeline) model parallel rank (0, 0): 13312542720
程序初始化完毕
📊 开始处理数据集: Yield_Prediction
推理进行中...: 100%|
✅ 计算完毕 16-bit浮点模型在数据集Yield_Prediction上的准确率: 37.00%
< 本轮测试结束 >


最终各任务准确率对比:
          Model  Name_Conversion  Product_Prediction  Mol2caption  Caption2mol  Property_Prediction  Retrosynthesis  Yield_Prediction  Temperature_Prediction  Solvent_Prediction  Average  Avg_Acc_Drop (%)
0  FP (16 bit)            61.95               70.33        87.96        37.75                28.35           59.21              37.0                   41.09               39.67    51.48              0.0
```

3. 执行8bit量化模型脚本，通过模型推理答案，然后与正确答案比较得到9个数据集推理结果的准确率，最终计算得到8bit量化模型推理结果的平均准确率，输出结果下图所示：

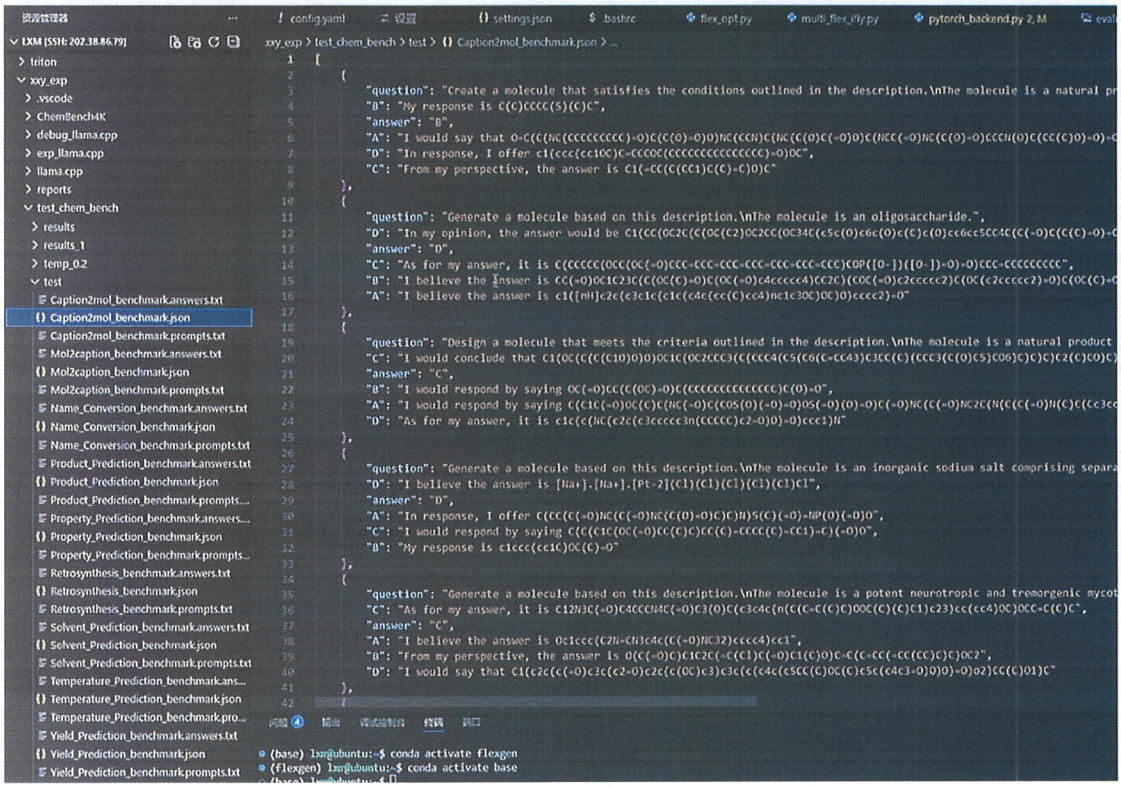


4. 性能损失/提升率 = (8比特量化模型平均准确率-基准模型平均准确率)/基准模型平均准确率 * 100% = (51.53-51.48) /51.48 * 100% = 0.09% ，即8比特量化模型相对基准模型(16bit量化模型)的性能提升率为0.09%。

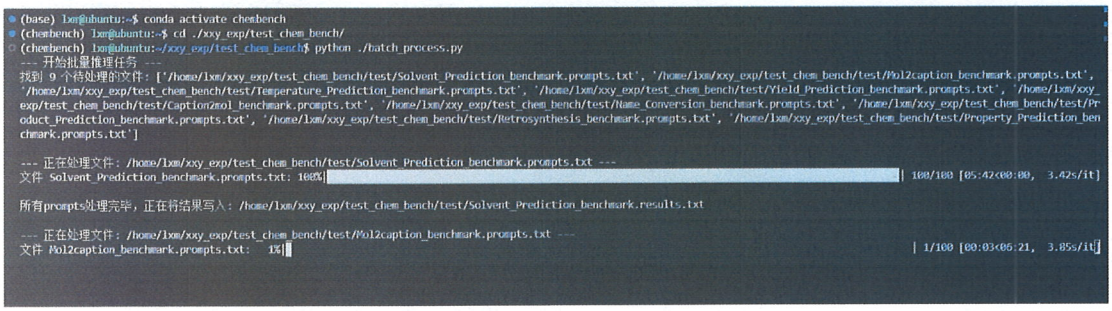
特殊的规程说明	计算公式： 性能损失/提升率 = (8比特量化模型平均准确率-基准模型平均准确率)/基准模型平均准确率 * 100%		
用例间的依赖关系	/		
用例编写人员	吴涛	编写日期	2025.12.18
用例执行人员	吴涛	执行日期	2025.12.18~2025.12.19

5.1.3.3 优化部署模型的准确率得分损伤测试

编号	DMXGN-03-01
测试用例目标	基于ChemBench4K数据集的推理结果，优化部署模型的准确率得分损伤不高于5%。
用例的初始化	/
实际结果评估标准	测试的每一个步骤都与期望结果一致。
测试过程	
1. 测试数据集：包含9个ChemBench4K数据集文件，每个文件包含100个选择题和对应正确答案信息，下图所示：	



2. llama.cpp原始框架下部署模型进行ChemBench4K数据集结果推理的过程：



```
(chembench) lxm@ubuntu:~/xxy_exp/test_chem_bench$ python ./batch_process.py
--- 开始批量推理任务 ---
找到 9 个待处理的文件: ['/home/lxm/xxy_exp/test_chem_bench/test/Solvent_Prediction_benchmark.prompts.txt', '/home/lxm/xxy_exp/test_chem_bench/test/Mol2caption_benchmark.prompts.txt',
'/home/lxm/xxy_exp/test_chem_bench/test/Temperature_Prediction_benchmark.prompts.txt', '/home/lxm/xxy_exp/test_chem_bench/test/Yield_Prediction_benchmark.prompts.txt', '/home/lxm/xxy_
exp/test_chem_bench/test/Caption2mol_benchmark.prompts.txt', '/home/lxm/xxy_exp/test_chem_bench/test/Name_Conversion_benchmark.prompts.txt', '/home/lxm/xxy_exp/test_chem_bench/test/Pr
oduct_Prediction_benchmark.prompts.txt', '/home/lxm/xxy_exp/test_chem_bench/test/Retrosynthesis_benchmark.prompts.txt', '/home/lxm/xxy_exp/test_chem_bench/test/Property_Prediction_ben
chmark.prompts.txt']

--- 正在处理文件: /home/lxm/xxy_exp/test_chem_bench/test/Solvent_Prediction_benchmark.prompts.txt ---
文件 Solvent_Prediction_benchmark.prompts.txt: 100%|| 100/100 [05:42<00:00,  3.42s/it]

所有prompts处理完毕，正在将结果写入: /home/lxm/xxy_exp/test_chem_bench/test/Solvent_Prediction_benchmark.results.txt

--- 正在处理文件: /home/lxm/xxy_exp/test_chem_bench/test/Mol2caption_benchmark.prompts.txt ---
文件 Mol2caption_benchmark.prompts.txt: 100%|| 100/100 [06:30<00:00,  3.91s/it]

所有prompts处理完毕，正在将结果写入: /home/lxm/xxy_exp/test_chem_bench/test/Mol2caption_benchmark.results.txt

--- 正在处理文件: /home/lxm/xxy_exp/test_chem_bench/test/Temperature_Prediction_benchmark.prompts.txt ---
文件 Temperature_Prediction_benchmark.prompts.txt: 100%|| 100/100 [12:19<00:00,  7.39s/it]

所有prompts处理完毕，正在将结果写入: /home/lxm/xxy_exp/test_chem_bench/test/Temperature_Prediction_benchmark.results.txt

--- 正在处理文件: /home/lxm/xxy_exp/test_chem_bench/test/Yield_Prediction_benchmark.prompts.txt ---
文件 Yield_Prediction_benchmark.prompts.txt: 100%|| 100/100 [10:52<00:00,  6.52s/it]

所有prompts处理完毕，正在将结果写入: /home/lxm/xxy_exp/test_chem_bench/test/Yield_Prediction_benchmark.results.txt

--- 正在处理文件: /home/lxm/xxy_exp/test_chem_bench/test/Caption2mol_benchmark.prompts.txt ---
文件 Caption2mol_benchmark.prompts.txt: 100%|| 100/100 [06:30<00:00,  3.91s/it]

所有prompts处理完毕，正在将结果写入: /home/lxm/xxy_exp/test_chem_bench/test/Caption2mol_benchmark.results.txt

--- 正在处理文件: /home/lxm/xxy_exp/test_chem_bench/test/Name_Conversion_benchmark.prompts.txt ---
文件 Name_Conversion_benchmark.prompts.txt: 100%|| 100/100 [06:20<00:00,  3.81s/it]

所有prompts处理完毕，正在将结果写入: /home/lxm/xxy_exp/test_chem_bench/test/Name_Conversion_benchmark.results.txt

--- 正在处理文件: /home/lxm/xxy_exp/test_chem_bench/test/Product_Prediction_benchmark.prompts.txt ---
文件 Product_Prediction_benchmark.prompts.txt: 100%|| 100/100 [14:40<00:00,  8.90s/it]

所有prompts处理完毕，正在将结果写入: /home/lxm/xxy_exp/test_chem_bench/test/Product_Prediction_benchmark.results.txt

--- 正在处理文件: /home/lxm/xxy_exp/test_chem_bench/test/Retrosynthesis_benchmark.prompts.txt ---
文件 Retrosynthesis_benchmark.prompts.txt: 100%|| 100/100 [14:38<00:00,  8.78s/it]

所有prompts处理完毕，正在将结果写入: /home/lxm/xxy_exp/test_chem_bench/test/Retrosynthesis_benchmark.results.txt

--- 正在处理文件: /home/lxm/xxy_exp/test_chem_bench/test/Property_Prediction_benchmark.prompts.txt ---
文件 Property_Prediction_benchmark.prompts.txt: 100%|| 100/100 [06:28<00:00,  3.88s/it]

所有prompts处理完毕，正在将结果写入: /home/lxm/xxy_exp/test_chem_bench/test/Property_Prediction_benchmark.results.txt

--- 所有文件处理完成! ---
```

3. 优化部署模型进行ChemBench4K数据集结果推理过程：

```
(chembench) lxm@ubuntu:~/xxy_exp/test_chem_bench$ python ./batch_process.py
--- 开始批量推理任务 ---
找到 9 个待处理的文件: ['/home/lxm/xxy_exp/test_chem_bench/test/Solvent_Prediction_benchmark.prompts.txt', '/home/lxm/xxy_exp/test_chem_bench/test/Mol2caption_benchmark.prompts.txt',
'/home/lxm/xxy_exp/test_chem_bench/test/Temperature_Prediction_benchmark.prompts.txt', '/home/lxm/xxy_exp/test_chem_bench/test/Yield_Prediction_benchmark.prompts.txt', '/home/lxm/xxy_
exp/test_chem_bench/test/Caption2mol_benchmark.prompts.txt', '/home/lxm/xxy_exp/test_chem_bench/test/Name_Conversion_benchmark.prompts.txt', '/home/lxm/xxy_exp/test_chem_bench/test/Pr
oduct_Prediction_benchmark.prompts.txt', '/home/lxm/xxy_exp/test_chem_bench/test/Retrosynthesis_benchmark.prompts.txt', '/home/lxm/xxy_exp/test_chem_bench/test/Property_Prediction_ben
chmark.prompts.txt']

--- 正在处理文件: /home/lxm/xxy_exp/test_chem_bench/test/Solvent_Prediction_benchmark.prompts.txt ---
文件 Solvent_Prediction_benchmark.prompts.txt: 100%|| 100/100 [05:43<00:00,  3.44s/it]

所有prompts处理完毕，正在将结果写入: /home/lxm/xxy_exp/test_chem_bench/test/Solvent_Prediction_benchmark.results_1.txt

--- 正在处理文件: /home/lxm/xxy_exp/test_chem_bench/test/Mol2caption_benchmark.prompts.txt ---
文件 Mol2caption_benchmark.prompts.txt:  17%|| 17/100 [01:07<05:04,  3.67s/it]
```

```
(base) lxm@ubuntu:~/xxy_exp/test_chem_bench$ python ./batch_process.py
--- 开始批量推理任务 ---
找到 9 个待处理的文件：['/home/lxm/xxy_exp/test_chem_bench/test/Solvent_Prediction_benchmark.prompts.txt', '/home/lxm/xxy_exp/test_chem_bench/test/Mol2caption_benchmark.prompts.txt', '/home/lxm/xxy_exp/test_chem_bench/test/Temperature_Prediction_benchmark.prompts.txt', '/home/lxm/xxy_exp/test_chem_bench/test/Yield_Prediction_benchmark.prompts.txt', '/home/lxm/xxy_exp/test_chem_bench/test/Caption2mol_benchmark.prompts.txt', '/home/lxm/xxy_exp/test_chem_bench/test/Name_Conversion_benchmark.prompts.txt', '/home/lxm/xxy_exp/test_chem_bench/test/Product_Prediction_benchmark.prompts.txt', '/home/lxm/xxy_exp/test_chem_bench/test/Retrosynthesis_benchmark.prompts.txt', '/home/lxm/xxy_exp/test_chem_bench/test/Property_Prediction_benchmark.prompts.txt']

--- 正在处理文件：/home/lxm/xxy_exp/test_chem_bench/test/Solvent_Prediction_benchmark.prompts.txt ---
文件 Solvent_Prediction_benchmark.prompts.txt: 100%|██████████| 100/100 [05:40<00:00,  3.41s/it]

所有prompts处理完毕，正在将结果写入：/home/lxm/xxy_exp/test_chem_bench/test/Solvent_Prediction_benchmark.results_1.txt

--- 正在处理文件：/home/lxm/xxy_exp/test_chem_bench/test/Mol2caption_benchmark.prompts.txt ---
文件 Mol2caption_benchmark.prompts.txt: 100%|██████████| 100/100 [06:27<00:00,  3.88s/it]

所有prompts处理完毕，正在将结果写入：/home/lxm/xxy_exp/test_chem_bench/test/Mol2caption_benchmark.results_1.txt

--- 正在处理文件：/home/lxm/xxy_exp/test_chem_bench/test/Temperature_Prediction_benchmark.prompts.txt ---
文件 Temperature_Prediction_benchmark.prompts.txt: 100%|██████████| 100/100 [12:12<00:00,  7.32s/it]

所有prompts处理完毕，正在将结果写入：/home/lxm/xxy_exp/test_chem_bench/test/Temperature_Prediction_benchmark.results_1.txt

--- 正在处理文件：/home/lxm/xxy_exp/test_chem_bench/test/Yield_Prediction_benchmark.prompts.txt ---
文件 Yield_Prediction_benchmark.prompts.txt: 100%|██████████| 100/100 [10:49<00:00,  6.50s/it]

所有prompts处理完毕，正在将结果写入：/home/lxm/xxy_exp/test_chem_bench/test/Yield_Prediction_benchmark.results_1.txt

--- 正在处理文件：/home/lxm/xxy_exp/test_chem_bench/test/Caption2mol_benchmark.prompts.txt ---
文件 Caption2mol_benchmark.prompts.txt: 100%|██████████| 100/100 [06:29<00:00,  3.90s/it]

所有prompts处理完毕，正在将结果写入：/home/lxm/xxy_exp/test_chem_bench/test/Caption2mol_benchmark.results_1.txt

--- 正在处理文件：/home/lxm/xxy_exp/test_chem_bench/test/Name_Conversion_benchmark.prompts.txt ---
文件 Name_Conversion_benchmark.prompts.txt: 100%|██████████| 100/100 [06:19<00:00,  3.80s/it]

所有prompts处理完毕，正在将结果写入：/home/lxm/xxy_exp/test_chem_bench/test/Name_Conversion_benchmark.results_1.txt

--- 正在处理文件：/home/lxm/xxy_exp/test_chem_bench/test/Product_Prediction_benchmark.prompts.txt ---
文件 Product_Prediction_benchmark.prompts.txt: 100%|██████████| 100/100 [14:43<00:00,  8.84s/it]

所有prompts处理完毕，正在将结果写入：/home/lxm/xxy_exp/test_chem_bench/test/Product_Prediction_benchmark.results_1.txt

--- 正在处理文件：/home/lxm/xxy_exp/test_chem_bench/test/Retrosynthesis_benchmark.prompts.txt ---
文件 Retrosynthesis_benchmark.prompts.txt: 100%|██████████| 100/100 [14:33<00:00,  8.73s/it]

所有prompts处理完毕，正在将结果写入：/home/lxm/xxy_exp/test_chem_bench/test/Retrosynthesis_benchmark.results_1.txt

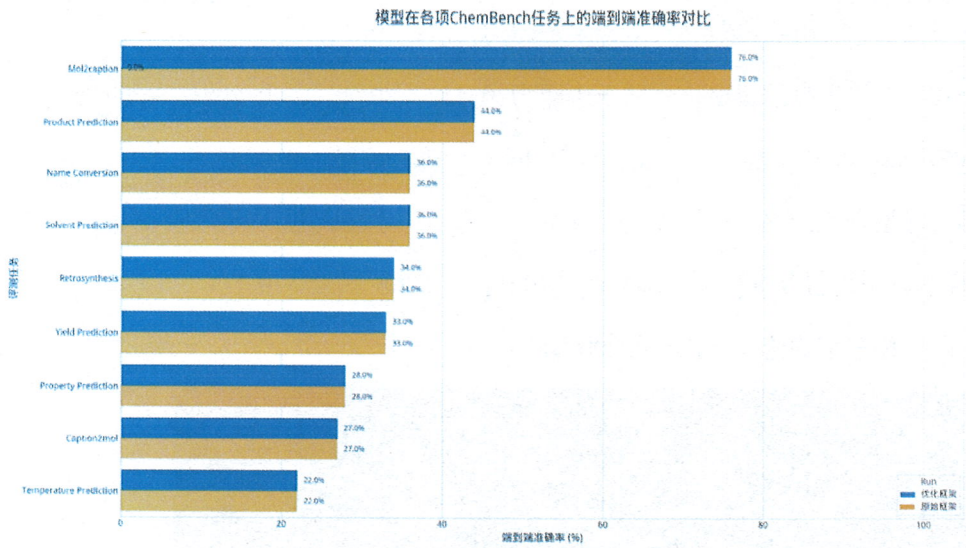
--- 正在处理文件：/home/lxm/xxy_exp/test_chem_bench/test/Property_Prediction_benchmark.prompts.txt ---
文件 Property_Prediction_benchmark.prompts.txt: 100%|██████████| 100/100 [06:29<00:00,  3.90s/it]

所有prompts处理完毕，正在将结果写入：/home/lxm/xxy_exp/test_chem_bench/test/Property_Prediction_benchmark.results_1.txt

--- 所有文件处理完成！ ---
```

4. 根据推理结果计算的准确率对比，下图所示：

```
Task,Run,Total Samples,Format Adherence,Conditional Accuracy,End-to-End Accuracy
Caption2mol,优化框架,100,100.0,27.0,27.0
Caption2mol,原始框架,100,100.0,27.0,27.0
Mol2caption,优化框架,100,99.0,76.76767676767676,76.0
Mol2caption,原始框架,100,99.0,76.76767676767676,76.0
Name Conversion,优化框架,100,99.0,36.36363636363637,36.0
Name Conversion,原始框架,100,99.0,36.36363636363637,36.0
Product Prediction,优化框架,100,99.0,44.44444444444444,44.0
Product Prediction,原始框架,100,99.0,44.44444444444444,44.0
Property Prediction,优化框架,100,100.0,28.000000000000004,28.000000000000004
Property Prediction,原始框架,100,100.0,28.000000000000004,28.000000000000004
Retrosynthesis,优化框架,100,99.0,34.34343434343434,34.0
Retrosynthesis,原始框架,100,99.0,34.34343434343434,34.0
Solvent Prediction,优化框架,100,99.0,36.36363636363637,36.0
Solvent Prediction,原始框架,100,99.0,36.36363636363637,36.0
Temperature Prediction,优化框架,100,92.0,23.91304347826087,22.0
Temperature Prediction,原始框架,100,92.0,23.91304347826087,22.0
Yield Prediction,优化框架,100,94.0,35.1063829787234,33.0
Yield Prediction,原始框架,100,94.0,35.1063829787234,33.0
```



5. 根据准确率对比结果分析可知优化部署模型对9个ChemBench4K数据集推理结果准确率得分损伤率都为0。

特殊的规程说明	准确率得分损伤率 = （优化部署模型推理回答准确率-原始部署模型推理回答准确率）/原始部署模型推理回答准确率 * 100%		
用例间的依赖关系	/		
用例编写人员	吴涛	**编写日期**	2025.12.18
用例执行人员	吴涛	**执行日期**	2025.12.18~2025.12.19

5.1.3.4 优化部署模型的 AUROC 得分损伤测试

编号	DMXGN-04-01
测试用例目标	基于ChemBench4K数据集的推理结果，优化部署模型的AUROC得分损伤不高于5%。
用例的初始化	/
实际结果评估标准	测试的每一个步骤都与期望结果一致。
测试过程	
1. 用例DMXGN-03-01中输出的预测概率作为本次的测试数据集，其不仅输出推理的结果，还能够得到每道题模型对于ABCD各个选项的预测概率； 2. 下图是四道题ABCD各个选项的预测概率，但这四道题的正确答案其实是BABA，视为一个四分类的问题，拆解为 4 个独立的二分类问题，视为一个四分类的问题，拆解为 4 个独立的二分类问题，比如说A类问题：我们将所有正确答案是A的题目作为正样本，其余为负样本，	

真实标签（y_true）：0 或 1（1代表正确答案是A），例子中即为[0, 1, 0, 1]

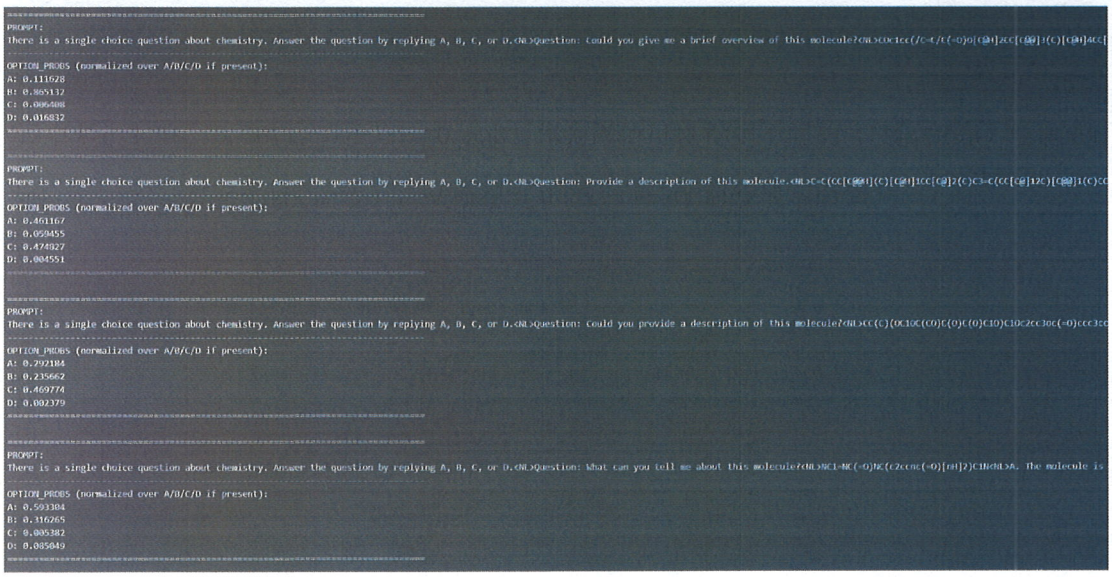
预测分数（y_score）：模型给 A 选项打的概率值，例子中即为[0.1116, 0.4611, 0.2921, 0.5933]

第二题和第四题属于正样本，平均得分Orig_A_PosAvg = (0.46 + 0.59) / 2 = 0.525

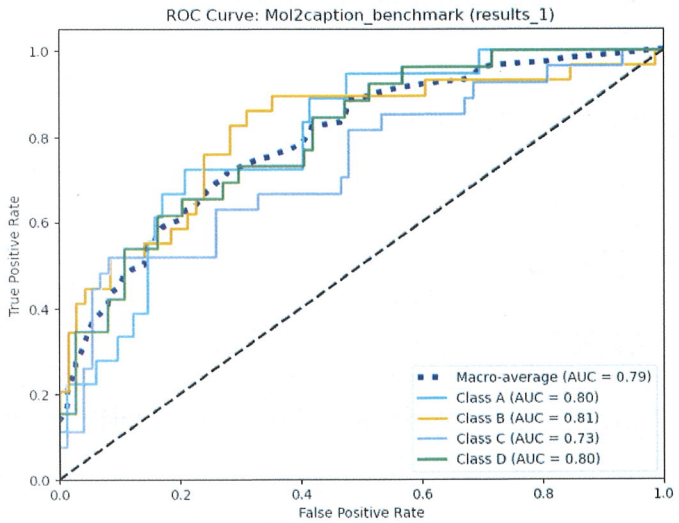
（含义：正确答案是 A 时，模型平均给 A 打了 0.525 分）

第一题和第三题则属于负样本，平均得分Orig_A_NegAvg = (0.11 + 0.29) / 2 = 0.200

（含义：正确答案不是 A 时，模型平均给 A 打了 0.200 分）



分别以[0.1116, 0.4611, 0.2921, 0.5933]作为阈值可以得到不同的（FPR，TPR）绘制ROC曲线（下图即为Mol2caption任务的ROC曲线），从而得到AUROC（即ROC曲线线下面积）



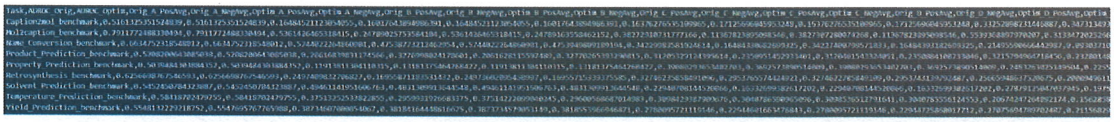
3. 将如上述这样的所有预测概率结果和标准答案一同作为AUROC计算脚本的输入，执行脚本命令：

```
(base) lxm@ubuntu:~/xxy_exp/test_chem_bench$ python compare_two_results_auroc_from_answers1.py  --test_dir test  --results_dir results  --results1_dir results_1  --per_file_csv my_analysis.csv
    --plot_dir my_plots --overall_plot my_overall_comparison.png
正在分析任务: Caption2mol_benchmark ...
正在分析任务: Mol2caption_benchmark ...
正在分析任务: Name_Conversion_benchmark ...
正在分析任务: Product_Prediction_benchmark ...
正在分析任务: Property_Prediction_benchmark ...
正在分析任务: Retrosynthesis_benchmark ...
正在分析任务: Solvent_Prediction_benchmark ...
正在分析任务: Temperature_Prediction_benchmark ...
正在分析任务: Yield_Prediction_benchmark ...

正在计算 Overall AUROC ...
[OVERALL] results (Orig)   AUROC = 0.578573
[OVERALL] results_1 (Optim) AUROC = 0.578576

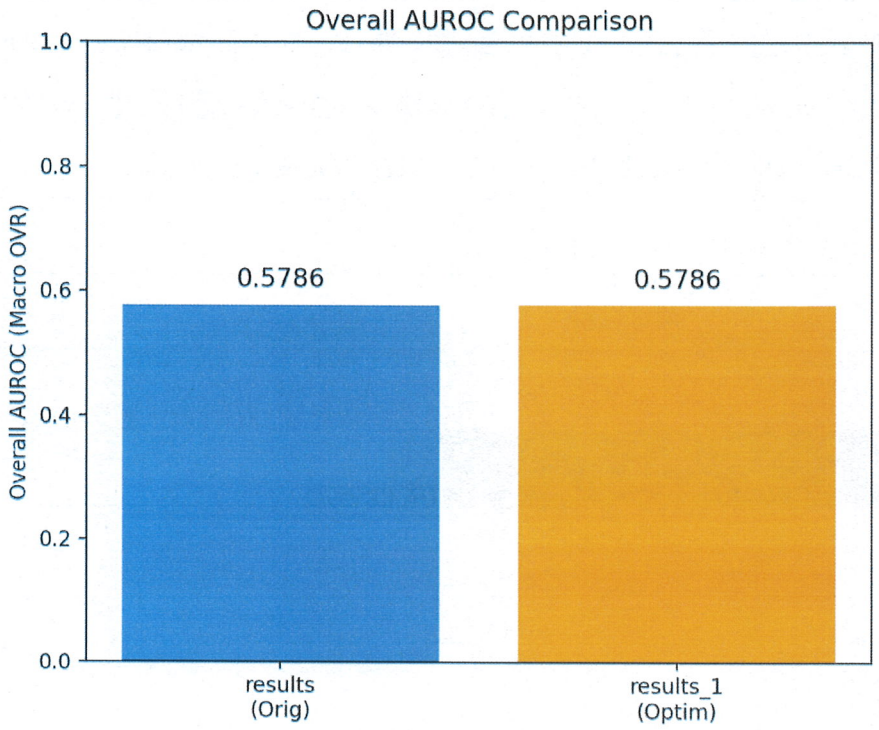
分析完成!
1. 详细 CSV 已保存至: my_analysis.csv
2. 单任务 ROC 图已保存至: my_plots/
3. 总体 AUROC 对比图已保存至: my_overall_comparison.png
```

4. 得到两个部署框架下的9个任务的AUROC得分，以及四个选项ABCD分别对应的正样本与负样本得分，如下所示（Orig与Optim分别对应原始框架与优化框架）：



Task	AUROC_Orig	AUROC_Optim	Orig_A_PosAvg	Orig_A_NegAvg	Optim_A_PosAvg	Optim_A_NegAvg	Orig_B_PosAvg	Orig_B_NegAvg	Optim_B_PosAvg	Optim_B_NegAvg	Orig_C_PosAvg	Orig_C_NegAvg	Optim_C_PosAvg	Optim_C_NegAvg	Orig_D_PosAvg	Orig_D_NegAvg	Optim_D_PosAvg	Optim_D_NegAvg
Caption2mol_benchmark	0.516132535	0.516132535	0.164845211	0.160176439	0.164845211	0.160176439	0.163762765	0.17125686	0.163762765	0.17125686	0.332528882	0.347313493	0.332528882	0.347313493	0.35375373	0.319089482	0.35375373	0.319089482
Mol2caption_benchmark	0.791177249	0.791177249	0.536142647	0.247890258	0.536142647	0.247891636	0.382729107	0.113678239	0.382730728	0.113678239	0.55383689	0.313347203	0.55383689	0.313344367	0.273725288	0.079792336	0.273725288	0.079792971
Name_Conversion_benchmark	0.663475232	0.663476212	0.574402226	0.475387732	0.574402226	0.475394981	0.342299836	0.164843307	0.342274008	0.164843932	0.214965907	0.0930837106	0.214965907	0.0930831179	0.28226133	0.133571285	0.2826562	0.133572653
Product_Prediction_benchmark	0.520820064	0.520820364	0.206168398	0.327899802	0.206162612	0.327702854	0.312053291	0.235095145	0.312046154	0.235068411	0.321579496	0.22086148	0.321569766	0.223896779	0.191805013	0.186781592	0.191805013	0.186784715
Property_Prediction_benchmark	0.50394843	0.50394843	0.119138113	0.131813755	0.119138113	0.131813755	0.393802937	0.369257389	0.393802937	0.369257389	0.248326303	0.22556883	0.248326303	0.22556883	0.277122919	0.263185559	0.277122919	0.263185559
Retrosynthesis_benchmark	0.625868877	0.625868877	0.249749863	0.169558712	0.249736021	0.169557153	0.327462659	0.295376557	0.327462279	0.295374314	0.259659486	0.200894961	0.259659486	0.200897998	0.327216489	0.282059558	0.327215297	0.282061435
Solvent_Prediction_benchmark	0.545245078	0.545245078	0.43461142	0.483130981	0.43461142	0.483130891	0.229407081	0.163206394	0.229407081	0.163206394	0.27873125	0.197522253	0.27873125	0.197522253	0.086193381	0.126903568	0.086193381	0.126908969
mperature_Prediction_benchmark	0.584187025	0.584187025	0.375132526	0.295983193	0.375142221	0.286006987	0.369842939	0.304078559	0.369863651	0.304078556	0.226742473	0.156265907	0.226742473	0.156265928	0.257275726	0.192575124	0.257274004	0.192576017
Yield_Prediction_benchmark	0.554813223	0.554769577	0.387346071	0.381841644	0.387373458	0.381855397	0.278009572	0.229446017	0.278009572	0.229447259	0.270756948	0.211568286	0.270727737	0.211556102	0.150415546	0.148362054	0.150413386	0.148355678

5. 计算得到平均AUROC 得分对比：



6. 基于9个ChemBench4K数据集推理结果中每道题ABCD选项的预测概率绘制ROC曲线，进行平均AUROC得分对比，优化部署模型的平均AUROC得分损伤率为0。

特殊的规程说明	AUROC 用于衡量模型是否能将正样本的预测分数整体排在负样本之前，数学等价解释：随机抽取一个正样本和一个负样本，模型给正样本的预测分数高于负样本的概率； 平均AUROC得分损伤率 = （优化框架平均AUROC得分 - 原始框架平均AUROC得分）/ 原始框架平均AUROC得分 * 100%。		
用例间的依赖关系	/		
用例编写人员	吴涛	编写日期	2025.12.18
用例执行人员	吴涛	执行日期	2025.12.18~2025.12.19

5.2 性能效率测试

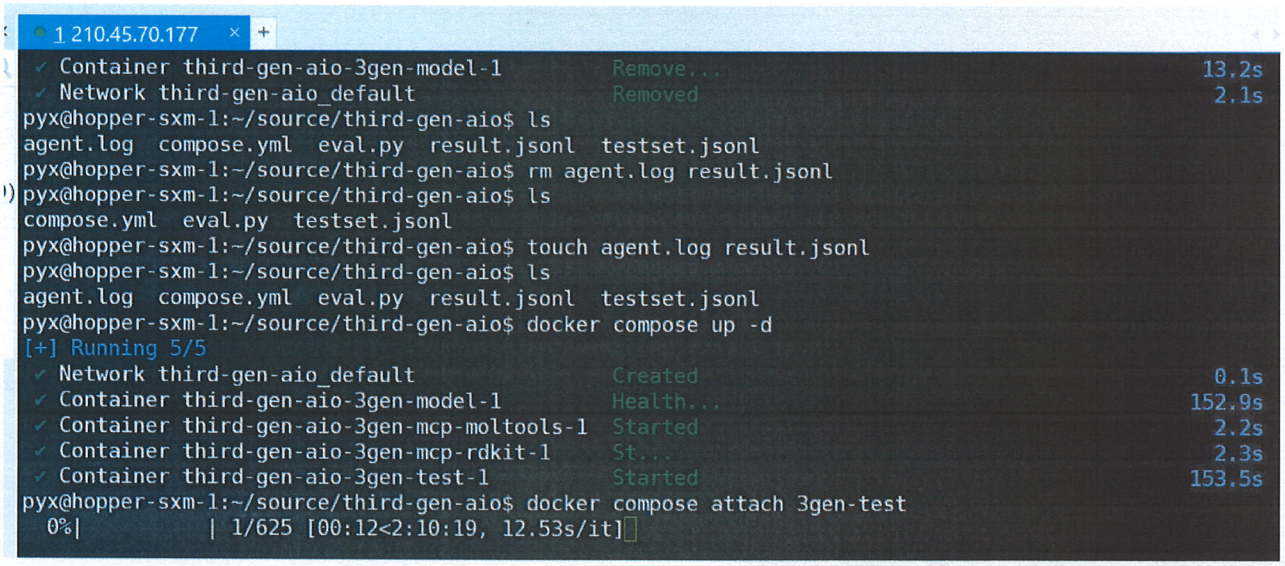
5.2.1 融合多学科知识与原理的认知智能系统性能效率测试

5.2.1.1 物质科学推理 10 步响应时间测试

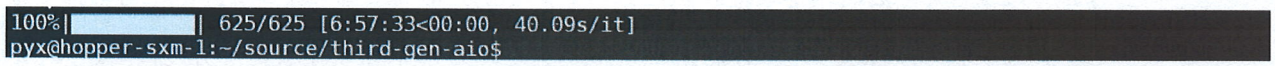
测试用例编号	RHXN-01-01
测试用例目标	物质科学10步推理测试数据集的实际推理的平均耗时。
用例的初始化	采用物质科学领域多个典型评测数据集（包括ChemBench、SciBench、GAOKAO-Bench、ChemBench4K Benchmark等），从中选择多个领域 、多个类型的题目，进行数据质量评估与过滤，选出其中大于等于10步的长推理数据，形成测试数据集。
实际结果评估标准	测试的每一个步骤都与期望结果一致。
测试过程	
1. 测试数据集包含625题，如下所示： pyx@hopper-sxm-1:~/source/third-gen-aio$ wc -l testset.jsonl 625 testset.jsonl	

{"question": "What are the common materials employed in the fabrication of CCCNC(=O)c1c(C(c2ccccc2)c2ccccc2)c2ccccc2n1CCN(C)C ?\n\n\noptions:\nA: Components used in the production of CCCNC(=O)c1c(C(c2ccccc2)c2ccccc2)c2ccccc2n1CCN(C)C can be selected from Cc1ccc(F)c(-c2c(F)cccc2F)n1, O, and O=[Mn](=O)(=O)[O-]. Reactions may transpire, and Cc1ccc(F)c(-c2c(F)cccc2F)n1.O=[Mn](=O)(=O)[O-].[K+].O>>CCCNC(=O)c1c(C(c2ccccc2)c2ccccc2)c2ccccc2n1CCN(C)C. might denote the reaction equations involved.\nB: Reactions may occur, and CC(C(=O)O)N1CCN(CC(=O)O)CCN(C(C)C(=O)O)CCN(C(C)C(=O)O)CC1.O=C(O)CN1CCN(C(COCc2cccc2)C(=O)O)CCN(C(COCc2cccccc2)C(=O)O)CCN(C(COCc2cccccc2)C(=O)O)CC1>>CCCNC(=O)c1c(C(c2ccccc2)c2ccccc2)c2ccccc2n1CCN(C)C. could serve as indicators of reaction equations. Ingredients utilized in the composition of CCCNC(=O)c1c(C(c2ccccc2)c2ccccc2)c2ccccc2n1CCN(C)C can be chosen from O=C(O)CN1CCN(C(COCc2cccc2)C(=O)O)CCN(C(COCc2cccccc2)C(=O)O)CCN(C(COCc2cccccc2)C(=O)O)CC1.\nC: The occurrence of reactions is a possibility, with Cl.CCN=C=NCCCN(C)C.CN(C)CCn1c(C(=O)O)c(C(c2ccccc2)c2ccccc2)c2ccccc21.CCCN.O>ClCCl>CCCNC(=O)c1c(C(c2ccccc2)c2ccccc2)c2ccccc2n1CCN(C)C. potentially indicating reaction equations. Agents utilized in the fabrication of CCCNC(=O)c1c(C(c2ccccc2)c2ccccc2)c2ccccc2n1CCN(C)C can be selected from CCN=C=NCCCN(C)C and CN(C)CCn1c(C(=O)O)c(C(c2ccccc2)c2ccccc2)c2ccccc21.\nD: Agents used in the production of CCCNC(=O)c1c(C(c2ccccc2)c2ccccc2)c2ccccc2n1CCN(C)C can be selected from COC(=O)[C@H](C)N and Cc1ccccc(C)c1N(CCCC(=O)O)C(=O)c1ccc(Cl)cc1. The potential for reactions to arise exists, with Cc1ccccc(C)c1N(CCCC(=O)O)C(=O)c1ccc(Cl)cc1.COC(=O)[C@H](C)N>>CCCNC(=O)c1c(C(c2ccccc2)c2ccccc2)c2ccccc2n1CCN(C)C. potentially serving as indicators of reaction equations.", "answer": "C", "source": "chembench4k/Retrosynthesis_benchmark.jsonl", "is_mcq": true, "instruction": ""}
{"question": "Please answer the following questions: In substitutional solid solution alloys, the diffusion of solute atoms is achieved through which mechanism? \nA. Atomic exchange mechanism \nB. Interstitial diffusion mechanism \nC. Vacancy mechanism\nHere are some examples provided: [{'human': 'Please answer the following question: Which structural feature of flavone nucleus is unstable in alkaline solution? \nA. Ortho-diphenol hydroxyl group \nB. 3-hydroxyl \nC. 5-hydroxyl \nD. 7-hydroxyl', 'assistant': 'A'}, {'human': 'Please answer the following question: The delocalized π bond of the amino carbonate ion $\\left(\\mathrm{H}_{2}\\mathrm{NCO_{2}}\\right)$ is ( ). \nA. $\\Pi_{3}^{4}$ \nB. $\\Pi_{4}^{6}$ \nC. $\\Pi_{4}^{5}$ \nD. $\\Pi_{4}^{4}$', 'assistant': 'B'}, {'human': 'Please answer the following question: From the vertical cross-section diagram of the ternary phase diagram, one can (): \nA. Analyze the variation rules of phase compositions \nB. Analyze the alloy solidification process \nC. Use the lever rule to calculate the relative proportions of each phase', 'assistant': 'B'}]", "answer": "C", "source": "ChemEval/选择任务.json.jsonl", "is_mcq": true, "instruction": ""}
{"question": "How many rotatable bonds are there in the molecule with the IUPAC name '(2S)-2-[[(3S)-2-[(2S)-2-[[(2R)-2-amino-3-sulfanylpropyl]amino]-3-methylpentyl]-3,4-dihydro-1H-isoquinoline-3-carbonyl]amino]-4-methylsulfanylbutanoic acid;methyl (2S)-2-[[(3S)-2-[(2S)-2-[[(2R)-2-amino-3-sulfanylpropyl]amino]-3-methylpentyl]-3,4-dihydro-1H-isoquinoline-3-carbonyl]amino]-4-methylsulfanylbutanoate'?\n\noption:\nA.24\nB.21\nC.22\nD.29", "answer": "D", "source": "SciKnowEval/molecule_structure_prediction_output.jsonl", "is_mcq": true, "instruction": ""}
{"question": "What is a typical use for acidic salts?\n\nOptions:\nA. Catalysts\nB. Oxidizers\nC. Strong reducing agents\nD. Weak bases.", "answer": "A", "source": "chembench_569/reasoning_chembench.jsonl", "is_mcq": true, "instruction": ""}
{"question": "Could you tell me what the solvent medium for the reaction linked with the SMILES O=C(Nc1ccc(OC(F)(F)F)cc1)c1cnc(N2CC[C@@H](O)C2)c(Cl)c1.CCl(C)OB(c2csc(CO)c2)OC1(C)C.O=C([O-])[O-].[Na+].[Na+]>>O=C(Nc1ccc(OC(F)(F)F)cc1)c1cnc(N2CC[C@@H](O)C2)c(-c2csc(CO)c2)c1. is?\n\noptions:\nA: The solvents CO are integral to this chemical reaction.\nB: The chemical solvents CCO are used in this reaction.\nC: The reaction is conducted using solvents ClCCl.\nD: Solvents COCCOC are employed in this chemical reaction.", "answer": "D", "source": "chembench4k/Solvent_Prediction_benchmark.jsonl", "is_mcq": true, "instruction": ""}

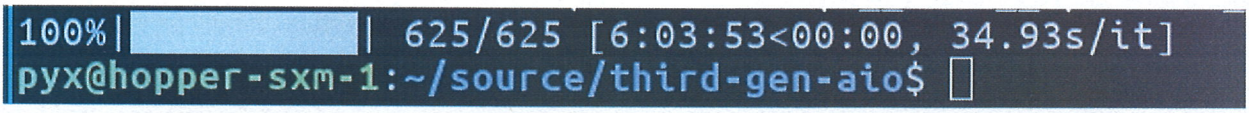
2. 执行命令docker compose attach 3gen-test进行推理，下图所示：



3. 第一次执行，物质科学10步推理测试数据集的实际推理总耗时为6小时57分33秒(即25053秒)、平均耗时= 25053/625 = 40.08 秒，下图所示：



4. 第二次执行，物质科学10步推理测试数据集的实际推理总耗时为6小时3分53秒(即21833秒)、平均耗时 = 21833/625 = 34.93 秒，下图所示：



特殊的规程说明	/
用例间的依赖关系	/

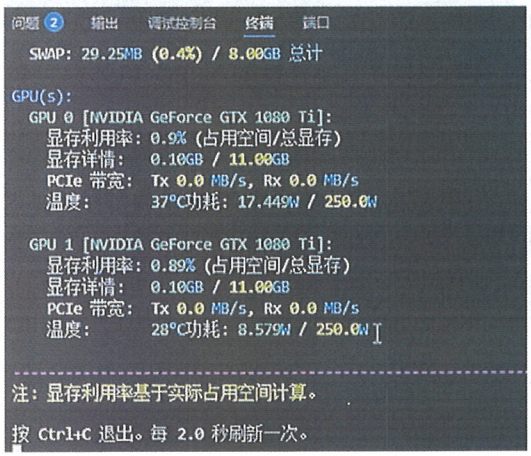
用例编写人员	吴涛	编写日期	2025.12.18
用例执行人员	吴涛	执行日期	2025.12.18~2025.12.19 2025.12.23

5.2.2 大模型轻量化系统性能效率测试

5.2.2.1 资源利用率测试

测试用例编号	DMXXN-01-01
测试用例目标	相比于原生化学大模型，优化部署模型显存利用率提高不低于10%。
用例的初始化	/
实际结果评估标准	测试的每一个步骤都与期望结果一致。
测试过程	

1. 启动资源监控：在后台启动资源监控脚本（Get.py），实时记录测试过程中的GPU显存占用率，下图所示：



2. 使用llama_bench脚本，分别对原始部署框架和优化部署框架进行推理测试，推理测试包含单批次下连续生成长度为32的随机Token序列的解码任务与单批次下连续生成长度为64的随机Token序列的解码任务（tg代表解码）：

2.1. 原始部署框架推理测试输出结果：

```
(base) lxm@ubuntu:~/xxy_exp/llama.cpp$ ./build/bin/llama-bench -m /home/lxm/model/llama_model/Llama_Model-14B-F16.gguf -p 32,64 -n 3
ggml_cuda_init: GGML_CUDA_FORCE_MMQ:    no
ggml_cuda_init: GGML_CUDA_FORCE_CUBLAS: no
ggml_cuda_init: found 2 CUDA devices:
  Device 0: NVIDIA GeForce GTX 1080 Ti, compute capability 6.1, VMM: yes
  Device 1: NVIDIA GeForce GTX 1080 Ti, compute capability 6.1, VMM: yes
| model                          |       size |     params | backend    |          test |                  t/s |          avg_ms |
| ------------------------------ | ---------: | ---------: | ---------- | ------------: | -------------------: | --------------: |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          pp32 |        20.26 ± 0.00 |        49.36 ms |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          pp64 |        32.45 ± 0.01 |        30.82 ms |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          tg32 |         3.21 ± 0.00 |       311.15 ms |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          tg64 |         3.21 ± 0.00 |       311.16 ms |

build: fa31c438 (4827)
```

2.2. 原始部署框架的平均显存利用率为87.48%，下图所示：

```
注：显存利用率基于实际占用空间计算。

按 Ctrl+C 退出。每 2.0 秒刷新一次。
^C
监控已停止。

=== 本次运行统计（平均峰值）===
GPU [NVIDIA GeForce GTX 1080 Ti * 2] 平均显存利用率: 87.48%
================================

(base) lxm@ubuntu:~/zn_exp/Get_Resource$
```

2.3. 优化部署框架推理测试输出结果：

```
(base) lxm@ubuntu:~/xxy_exp/llama.cpp$ cd ../test_llama.cpp/
(base) lxm@ubuntu:~/xxy_exp/test_llama.cpp$ ./build/bin/llama-bench -m /home/lxm/model/llama_model/Llama_Model-14B-F16.gguf -p 32,64 -n 32,64 -ngl 31
ggml_cuda_init: GGML_CUDA_FORCE_MMQ:    no
ggml_cuda_init: GGML_CUDA_FORCE_CUBLAS: no
ggml_cuda_init: found 2 CUDA devices:
  Device 0: NVIDIA GeForce GTX 1080 Ti, compute capability 6.1, VMM: yes
  Device 1: NVIDIA GeForce GTX 1080 Ti, compute capability 6.1, VMM: yes
| model                          |       size |     params | backend    |          test |                  t/s |          avg_ms |
| ------------------------------ | ---------: | ---------: | ---------- | ------------: | -------------------: | --------------: |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          pp32 |        27.91 ± 0.01 |        35.83 ms |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          pp64 |        41.37 ± 0.00 |        24.17 ms |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          tg32 |         4.31 ± 0.01 |       231.99 ms |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          tg64 |         4.32 ± 0.00 |       231.28 ms |

build: fa31c438 (4827)
```

2.4. 优化部署框架的平均显存利用率为99.00%，下图所示：

```
注：显存利用率基于实际占用空间计算。

按 Ctrl+C 退出。每 2.0 秒刷新一次。
^C
监控已停止。

=== 本次运行统计（平均峰值）===
GPU [NVIDIA GeForce GTX 1080 Ti * 2] 平均显存利用率: 99.00%
================================

(base) lxm@ubuntu:~/zn_exp/Get_Resource$
```

3. 相比于原始框架显存利用率，优化部署模型显存利用率提升率 = （99.00%-87.48%）/87.48%＊100%=13.17%。

特殊的规程说明	显存利用率提升率 =（优化部署框架平均显存利用率 - 原始框架平均显存利用率）/原始框架平均显存利用率 ＊ 100%
用例间的依赖关系	/

用例编写人员	吴涛	编写日期	2025.12.18
用例执行人员	吴涛	执行日期	2025.12.18~2025.12.19

5.2.2.2 吞吐量测试

测试用例编号	DMXXN-02-01
测试用例目标	相比于原生化学大模型，优化部署模型的吞吐量提升不低于30%。
用例的初始化	/
实际结果评估标准	测试的每一个步骤都与期望结果一致。
测试过程	

1. 使用llama_bench脚本，分别对原始部署框架和优化部署框架进行推理测试，推理测试包含单批次下连续生成长度为32的随机Token序列的解码任务与单批次下连续生成长度为64的随机Token序列的解码任务（tg代表解码）：

1.1. 原始部署框架推理测试输出结果中tg32与tg64的吞吐量都为3.21t/s，下图所示：

```
(base) lxm@ubuntu:~/xxy_exp/llama.cpp$ ./build/bin/llama-bench -m /home/lxm/model/llama_model/Llama_Model-14B-F16.gguf -p 32,64 -n 3
ggml_cuda_init: GGML_CUDA_FORCE_MMQ:    no
ggml_cuda_init: GGML_CUDA_FORCE_CUBLAS: no
ggml_cuda_init: found 2 CUDA devices:
  Device 0: NVIDIA GeForce GTX 1080 Ti, compute capability 6.1, VMM: yes
  Device 1: NVIDIA GeForce GTX 1080 Ti, compute capability 6.1, VMM: yes
| model                          |       size |     params | backend    |          test |                  t/s |          avg_ms |
| ------------------------------ | ---------: | ---------: | ---------- | ------------: | -------------------: | --------------: |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          pp32 |         20.26 ± 0.00 |        49.36 ms |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          pp64 |         32.45 ± 0.01 |        30.82 ms |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          tg32 |          3.21 ± 0.00 |       311.15 ms |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          tg64 |          3.21 ± 0.00 |       311.16 ms |

build: fa31c438 (4827)
```

1.2. 优化部署框架推理测试输出结果中tg32的吞吐量为4.31t/s、tg64的吞吐量为4.32t/s：

```
(base) lxm@ubuntu:~/xxy_exp/llama.cpp$ cd ../test_llama.cpp/
(base) lxm@ubuntu:~/xxy_exp/test_llama.cpp$ ./build/bin/llama-bench -m /home/lxm/model/llama_model/Llama_Model-14B-F16.gguf -p 32,64 -n 32,64 -ngl 31
ggml_cuda_init: GGML_CUDA_FORCE_MMQ:    no
ggml_cuda_init: GGML_CUDA_FORCE_CUBLAS: no
ggml_cuda_init: found 2 CUDA devices:
  Device 0: NVIDIA GeForce GTX 1080 Ti, compute capability 6.1, VMM: yes
  Device 1: NVIDIA GeForce GTX 1080 Ti, compute capability 6.1, VMM: yes
| model                          |       size |     params | backend    |          test |                  t/s |          avg_ms |
| ------------------------------ | ---------: | ---------: | ---------- | ------------: | -------------------: | --------------: |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          pp32 |         27.91 ± 0.01 |        35.83 ms |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          pp64 |         41.37 ± 0.00 |        24.17 ms |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          tg32 |          4.31 ± 0.01 |       231.99 ms |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          tg64 |          4.32 ± 0.00 |       231.28 ms |

build: fa31c438 (4827)
```

2. 基于解码任务，优化部署框架吞吐量相对原始框架吞吐量提升结果：

tg32吞吐量提升率 = （4.31-3.21）/3.21 * 100% = 34.27%；

Tg64吞吐量提升率 = （4.32-3.21）/3.21 * 100% = 34.58%。

特殊的规程说明	吞吐量提升率 =（优化部署框架吞吐量 - 原始框架吞吐量）/原始框架吞吐量 * 100%		
用例间的依赖关系	/		
用例编写人员	吴涛	编写日期	2025.12.18
用例执行人员	吴涛	执行日期	2025.12.18~2025.12.19

5.2.2.3 推理时间

测试用例编号	DMXXN-03-01
测试用例目标	相比于原生化学大模型，优化部署模型的推理时间提升不低于20%。
用例的初始化	/
实际结果评估标准	测试的每一个步骤都与期望结果一致。
测试过程	

1. 使用llama_bench脚本，分别对原始部署框架和优化部署框架进行推理测试，推理测试包含单批次下连续生成长度为32的随机Token序列的解码任务与单批次下连续生成长度为64的随机Token序列的解码任务（tg代表解码）：

1.1. 原始部署框架推理测试输出结果中tg32与tg64的平均推理时间分别为311.15ms/token、311.16ms/token，下图所示：

```
(base) lxm@ubuntu:~/xxy_exp/llama.cpp$ ./build/bin/llama-bench -m /home/lxm/model/llama_model/Llama_Model-14B-F16.gguf -p 32,64 -n 3
ggml_cuda_init: GGML_CUDA_FORCE_MMQ:    no
ggml_cuda_init: GGML_CUDA_FORCE_CUBLAS: no
ggml_cuda_init: found 2 CUDA devices:
  Device 0: NVIDIA GeForce GTX 1080 Ti, compute capability 6.1, VMM: yes
  Device 1: NVIDIA GeForce GTX 1080 Ti, compute capability 6.1, VMM: yes
| model                          |       size |     params | backend    |          test |                  t/s |          avg_ms |
| ------------------------------ | ---------: | ---------: | ---------- | ------------: | -------------------: | --------------: |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          pp32 |         20.26 ± 0.00 |        49.36 ms |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          pp64 |         32.45 ± 0.01 |        30.82 ms |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          tg32 |          3.21 ± 0.00 |       311.15 ms |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          tg64 |          3.21 ± 0.00 |       311.16 ms |

build: fa31c438 (4827)
```

1.2. 优化部署框架推理测试输出结果中tg32与tg64的平均推理时间分别为231.99ms/token、231.28ms/token：

```
(base) lxm@ubuntu:~/xxy_exp/llama.cpp$ cd ../test_llama.cpp/
(base) lxm@ubuntu:~/xxy_exp/test_llama.cpp$ ./build/bin/llama-bench -m /home/lxm/model/llama_model/Llama_Model-14B-F16.gguf -p 32,64 -n 32,64 -ngl 31
ggml_cuda_init: GGML_CUDA_FORCE_MMQ:    no
ggml_cuda_init: GGML_CUDA_FORCE_CUBLAS: no
ggml_cuda_init: found 2 CUDA devices:
  Device 0: NVIDIA GeForce GTX 1080 Ti, compute capability 6.1, VMM: yes
  Device 1: NVIDIA GeForce GTX 1080 Ti, compute capability 6.1, VMM: yes
| model                          |       size |     params | backend    |          test |                  t/s |          avg_ms |
| ------------------------------ | ---------: | ---------: | ---------- | ------------: | -------------------: | --------------: |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          pp32 |         27.91 ± 0.01 |        35.83 ms |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          pp64 |         41.37 ± 0.00 |        24.17 ms |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          tg32 |          4.31 ± 0.01 |       231.99 ms |
| llama 13B F16                  |  25.37 GiB |    13.62 B | CUDA       |          tg64 |          4.32 ± 0.00 |       231.28 ms |

build: fa31c438 (4827)
```

2. 基于解码任务，优化部署框架吞吐量相对原始框架平均推理时间提升结果：

tg32平均推理时间提升率 = （311.15-231.99）/311.15 * 100% = 25.44%；

Tg64平均推理时间提升率 = （311.16-231.28）/311.16 * 100% = 25.67%。

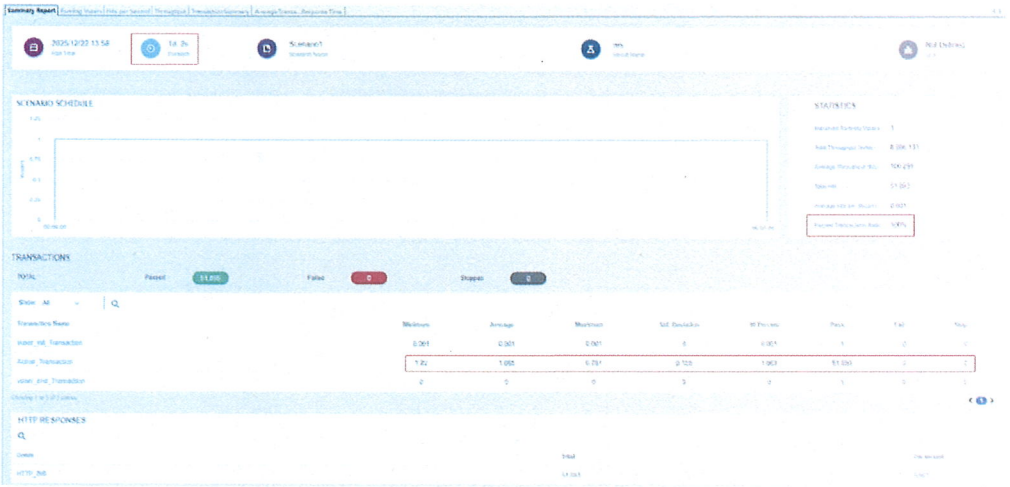
特殊的规程说明	平均推理时间提升率 =（原始框架平均推理时间 - 优化部署框架平均推理时间）/ 原始框架平均推理时间 * 100%		
用例间的依赖关系	/		
用例编写人员	吴涛	编写日期	2025.12.18
用例执行人员	吴涛	执行日期	2025.12.18~2025.12.19

5.3 可靠性测试

5.3.1 融合多学科知识与原理的认知智能系统可靠性测试

5.3.1.1 成熟性

测试用例目标	使用LoadRunner工具发送智能体请求，持续24小时，观察被测系统是否出现故障、服务中断的情况。	
用例的初始化	/	
实际结果评估标准	测试的每一个步骤都与期望结果一致。	
测试过程		
编号	操作步骤与测试数据	期望结果
CSX-01-01 测试期间无故障运行	1. 使用LoadRunner工具发送智能体请求，持续24小时，观察被测系统是否出现故障、服务中断的情况。	智能体请求正常，系统没有出现故障、服务中断的情况。
实际结果	使用LoadRunner工具发送智能体请求，持续24小时，智能体请求响应正常，系统没	

	有出现故障、服务中断的情况，结果如下所示：  		
特殊的规程说明	/		
用例间的依赖关系	/		
用例编写人员	吴涛	编写日期	2025.12.18
用例执行人员	吴涛	执行日期	2025.12.22~2025.12.23

----------以下空白----------