基于心理理论与大型语言模型的机器人协作系统项目进度报告

项目组

March 31, 2025

Abstract

本报告详细记录了基于心理理论（Theory of Mind, ToM）与大型语言模型（LLM）的机器人协作系统的开发进度。该系统基于 Watch-And-Help（WAH）项目的核心思想，通过观察人类行为演示，理解任务目标，并结合 ToM 推断人类意图和状态，生成动态协作策略并高效执行任务。报告详细介绍了系统架构、各关键模块的实现状态、当前功能演示结果以及下一阶段的开发计划。目前系统已完成了纯软件架构的实现，包括感知与目标推理、ToM 推理、协作规划、虚拟环境模拟等核心模块，能够在模拟环境中完成基础的人机协作任务。

Contents

1 项目概述

1.1 研究背景与意义

人机协作是机器人领域的核心挑战之一。传统协作模式下，机器人往往需要明确的指令才能行动，缺乏对人类意图的深入理解，导致协作效率低下。心理理论（Theory of Mind, ToM）作为认知科学和心理学中的重要概念，指的是理解和推断他人心理状态（如信念、意图、知识和情感）的能力，是高效社交互动的基础。

近年来，大型语言模型（LLM）展现出了强大的推理和生成能力，为将 ToM 能力引入机器人协作系统提供了新的可能性。通过结合 ToM 和 LLM，机器人可以更自然地理解人类意图，预测人类行为，并据此生成更加灵活和适应性强的协作策略。

本项目探索了将 ToM 与 LLM 相结合的机器人协作系统设计方法，以期提高人机协作的自然性和效率。该研究对家庭服务机器人、工业协作机器人以及辅助护理机器人等应用场景具有重要价值。

1.2 项目目标

本项目旨在设计和实现一个基于 ToM 和 LLM 的机器人协作系统，具体目标包括：

1. 设计能够从单次人类行为演示中理解任务目标的感知与推理模块
2. 实现基于 LLM 的 ToM 推理机制，用于推断人类意图、信念和状态
3. 开发结合规划算法与 LLM 的协作决策模块，生成灵活且人性化的协作策略
4. 构建虚拟协作环境，用于测试和评估系统性能
5. 通过对比实验，验证 ToM+LLM 方法相比传统方法的优势

本项目的核心创新点在于将心理理论的认知模型与大型语言模型的强大推理能力相结合，创造更具适应性和理解力的协作系统。

2 系统架构设计

2.1 整体架构

系统采用模块化设计，主要包含四个核心功能模块，以及实验与评估模块。各模块间通过标准化接口交互，形成完整的” 观察-推理-规划-执行” 工作流程。

系统整体架构如图1所示：

Figure 1: ToM-LLM 协作系统架构图

系统主要由以下几个部分组成：

1. **感知与目标推理模块**：处理视频或动作序列输入，提取任务目标和约束条件。
2. **心理理论 (ToM) 模块**：推断协作对象的意图、知识状态和潜在行为。
3. **协作规划模块**：结合任务目标和 ToM 推断结果，生成协作动作计划。
4. **虚拟环境接口**：模拟执行环境，支持动作执行和状态更新。
5. **实验与评估模块**：执行对比实验，收集和分析系统性能数据。

2.2 关键技术与方法

系统实现过程中采用了以下关键技术和方法：

1. **LLM 微调与提示工程**：通过设计特定的提示模板，指导 LLM 执行任务推理和 ToM 推断。
2. **混合规划算法**：结合启发式搜索算法（如蒙特卡洛树搜索）与 LLM 生成，平衡规划效率与灵活性。
3. **信念模型更新机制**：基于环境反馈动态调整 ToM 推断的置信度。
4. **模拟环境构建**：创建支持多代理交互的虚拟环境，模拟物理与社交规则。

3 核心模块实现状态

3.1 感知与目标推理模块

该模块负责通过处理人类演示（视频或动作序列），理解任务目标并进行结构化表示。

3.1.1 当前实现状态

- 完成了 PerceptionModule 类的基本实现
- 支持从动作序列中提取任务目标和步骤分解
- 实现了对象识别和关系分析的基础功能
- 添加了模拟模型支持，可在无实际预训练模型情况下运行

3.1.2 代码示例

以下代码展示了感知模块如何处理动作序列并提取任务目标：

```
def process_demonstration(self, demo_data, env_context):
    """处理演示数据，提取任务目标"""
    if isinstance(demo_data, str) and os.path.exists(demo_data):
        # 处理视频文件
        return self._process_video(demo_data, env_context)
    else:
        # 处理动作序列
        return self._process_action_sequence(demo_data, env_context)

def _process_action_sequence(self, actions, env_context):
    """从动作序列中提取任务目标"""
    # 1. 分析动作序列
    action_analysis = self._analyze_actions(actions)

    # 2. 结合环境上下文推断目标
    goal_inference = self._infer_goal(action_analysis, env_context)

    # 3. 构建结构化的任务目标
    task_goal = TaskGoal(
```

```
        name=goal_inference["goal"],
        target_state=self._construct_target_state(actions,
            env_context),
        constraints=self._extract_constraints(actions, env_context),
        decomposition=goal_inference["steps"],
        priority=1.0
    )

    return task_goal
```

Listing 1: 感知模块处理示例

3.2 心理理论 (ToM) 模块

ToM 模块负责推断人类的意图、信念和知识状态，为协作决策提供重要依据。

3.2.1 当前实现状态

- 完成了 TheoryOfMindModule 类的基本实现
- 实现了代理信念模型的数据结构和表示方法
- 支持基于环境状态和历史动作的信念推断
- 实现了基于环境反馈的信念更新机制
- 添加了模拟信念推断功能，支持系统测试

3.2.2 代码示例

以下代码展示了 ToM 模块如何推断代理信念：

```
def infer_agent_belief(self, agent_id, env_state, actions_history,
    task_context=None):
    """推断代理的信念、意图和目标"""
    # 1. 准备LLM推理的提示文本
    prompt = self._prepare_tom_prompt(
        agent_id, env_state, actions_history, task_context
    )

    # 2. 使用LLM进行推理(实际系统中调用LLM API)
    tom_output = self._simulate_llm_tom_inference(
        prompt, agent_id, actions_history
    )

    # 3. 解析LLM输出为结构化信念
    belief = self._parse_belief_from_llm(tom_output, agent_id)

    # 4. 更新历史信念记录
    self._update_belief_history(agent_id, belief)

    return belief
```

Listing 2: ToM 推理示例

3.3 协作规划模块

协作规划模块基于任务目标和 ToM 推断结果，生成详细的协作动作计划。

3.3.1 当前实现状态

- 完成了 CollaborationPlanningModule 类的基本实现
- 实现了基于 LLM 的高级规划生成功能
- 开发了计划验证和修复机制
- 加入了动作前提条件和效果的检查逻辑
- 添加了模拟计划生成功能，支持系统测试

3.3.2 代码示例

以下代码展示了协作规划模块如何生成动作计划：

```
def generate_collaboration_plan(self, task_goal, env_state,
    agent_beliefs, robot_id="robot"):
    """生成协作计划"""
    # 1. 使用LLM生成高级规划
    high_level_plan = self._generate_high_level_plan(
        task_goal, env_state, agent_beliefs
    )

    # 2. 使用MCTS将高级规划转化为具体动作序列
    action_sequence = self._plan_with_mcts(
        high_level_plan, env_state, agent_beliefs, robot_id
    )

    # 3. 验证计划的可执行性
    validated_plan = self._validate_plan(action_sequence, env_state)

    # 4. 添加自我反思和错误恢复策略
    final_plan = self._add_recovery_strategies(validated_plan,
        task_goal)

    return final_plan
```

Listing 3: 协作规划示例

3.4 虚拟环境接口

虚拟环境接口模拟了物理世界和社交互动规则，为系统提供测试和评估环境。

3.4.1 当前实现状态

- 完成了 VirtualEnvironmentInterface 类的基本实现
- 支持环境状态的表示和更新
- 实现了动作执行和效果模拟
- 支持文本方式的环境状态渲染
- 提供了环境重置和观察获取功能

3.4.2 代码示例

以下代码展示了虚拟环境如何执行动作并更新状态:

```
def step(self, action):
    """执行动作并更新环境状态"""
    # 添加到动作历史
    self.action_history.append(self._action_to_dict(action))

    # 更新环境状态
    self._update_environment_state(action)

    # 获取新的观察
    observation = self.get_observation()
    self.last_observation = observation

    return observation

def _update_environment_state(self, action):
    """根据动作更新环境状态"""
    # 更新时间戳
    self.current_state.timestamp += action.duration

    # 更新代理状态
    agent = self.current_state.agents.get(action.agent_id, {})

    # 根据动作类型更新状态
    if action.action_type == "move" and action.targets:
        target_location = action.targets[0]
        agent["location"] = target_location
    elif action.action_type == "pick" and action.targets:
        # 更新代理手持状态和对象关系
        # ...代码省略
```

Listing 4: 虚拟环境动作执行示例

3.5 集成系统

ToMCollaborationSystem 类将所有模块整合为完整系统，实现端到端的协作流程。

3.5.1 当前实现状态

- 完成了 ToMCollaborationSystem 类的基本实现
- 实现了组件初始化和配置管理
- 开发了完整的” 观察-推理-规划-执行” 工作流程
- 添加了任务完成状态评估功能
- 实现了模拟模型回退机制，提高系统鲁棒性

3.5.2 代码示例

以下代码展示了系统如何处理完整的协作流程:

```python
def run_complete_demo(self, demo_data):
    """运行完整的演示-推理-执行流程"""
    results = {}

    try:
        # 重置系统
        self.reset()

        # 1. 观察演示并推断任务
        task_goal = self.observe_and_infer_task(demo_data)
        results["task_goal"] = task_goal

        # 2. 推断代理信念
        agent_beliefs = self.infer_agent_beliefs()
        results["agent_beliefs"] = agent_beliefs

        # 3. 生成协作计划
        action_plan = self.plan_collaboration()
        results["action_plan"] = action_plan

        # 4. 执行计划
        execution_status = self.execute_plan(action_plan)
        results["execution_status"] = execution_status

        # 5. 收集最终状态
        final_observation = self.environment.get_observation()
        results["final_state"] = final_observation

        return results
    except Exception as e:
        print(f"运行过程中出错: {str(e)}")
        results["error"] = str(e)
        return results
```

Listing 5: 系统协作流程示例

4 功能演示与测试结果

4.1 系统运行环境

当前系统已在以下环境中进行了测试:

- 操作系统：Windows 10
- Python 版本：3.8+
- 核心依赖：PyTorch, Transformers, NumPy

4.2 测试场景

我们设计了一个简单的清理餐具测试场景，包含以下要素:

- 环境：包含桌子、水槽、柜台等位置
- 对象：杯子、盘子、海绵和抹布等物品
- 代理：机器人和人类
- 任务：清理脏餐具

4.3 运行结果

系统在模拟环境中成功执行了简单的清理任务。以下是运行日志摘录:

```
初始化ToM协作系统组件...
使用模拟模型进行初始化...
初始化VirtualHome-Social环境...
系统初始化完成

观察演示并推断任务...
模拟演示处理...
任务推断完成: 清理餐具

推断代理信念状态...
模拟推断代理 human 的信念...
已推断代理 human 的信念状态

生成协作计划...
模拟生成协作计划...
生成了 5 个动作的协作计划

开始执行协作计划...
执行动作: robot move ['table']
执行动作: robot pick ['cup_1']
执行动作: robot move ['sink']
执行动作: robot place ['cup_1', 'sink']
执行动作: human clean ['cup_1']

最终环境状态:
=== 虚拟环境状态 ===
时间: 9.0s
代理:
```

```
- robot 位于 sink
- human 位于 living_room
对象:
- cup_1: type=cup, 位于sink, 干净
- cup_2: type=cup, 位于table, 脏
- plate_1: type=plate, 位于table, 脏
- sponge_1: type=sponge, 位于sink, 干净
- cloth_1: type=cloth, 位于counter, 干净
```

Listing 6: 系统运行日志

通过日志可以观察到，系统成功完成了：

1. 任务目标识别：正确推断出” 清理餐具” 任务
2. 协作规划：生成了包含 5 个动作的协作计划
3. 计划执行：机器人成功移动、拿取物品并与人类协作完成了一个杯子的清洁

尽管系统只清理了一个杯子（还有其他餐具未清理），但这验证了系统的基本功能和工作流程。

5 未来工作计划

5.1 短期计划（1-2 个月）

1. **完善模块功能**：
   - 加强任务目标推理的泛化性，支持更复杂的任务表示
   - 完善 ToM 推理的动态更新机制，提高意图推断准确率
   - 扩展协作规划能力，处理更复杂的多步骤任务
2. **开发评估框架**：
   - 完成实验评估模块的全部功能
   - 实现基线系统对比功能
   - 设计并实现消融实验
3. **系统集成优化**：
   - 优化组件间的接口和数据流
   - 改进异常处理机制
   - 增强日志和监控功能

5.2 中期计划（3-6 个月）

1. **集成真实预训练模型**：
   - 选择并集成适合的视觉语言模型用于感知模块
   - 针对 ToM 任务微调 LLM 模型
   - 优化模型推理效率

2. **环境扩展**：
    - 构建更丰富的虚拟测试场景库
    - 添加更复杂的社交互动规则
    - 考虑不确定性和部分可观察性
3. **对比实验**：
    - 与 WAH 基线系统进行对比
    - 与其他协作方法进行性能比较
    - 分析 ToM 对协作效率的提升程度

5.3 长期计划（6 个月以上）

1. **实体机器人集成**：
    - 设计硬件接口，支持系统部署到实体机器人
    - 适配感知输入处理，处理实际传感器数据
    - 进行实际环境测试
2. **多代理协作扩展**：
    - 扩展系统支持多个机器人和人类的协作场景
    - 实现代理间的动态角色分配
    - 增强群体 ToM 推理能力
3. **应用场景研究**：
    - 家庭服务机器人应用测试
    - 工业协作场景适配
    - 评估在特殊人群（如老人、儿童）辅助中的应用潜力

6 风险与挑战

项目实施过程中可能面临以下挑战：

1. **ToM 建模复杂度**：准确推断人类复杂意图仍是一项挑战，尤其在隐含意图和错误信念场景中。
2. **LLM 幻觉问题**：LLM 可能生成不符合实际情况的推断结果，需要设计适当的约束和验证机制。
3. **实时性要求**：在实体机器人部署中，ToM 推理和规划需在有限时间内完成，对系统效率提出挑战。
4. **泛化能力**：系统需要泛化到未见过的任务和环境，这需要更丰富的训练数据和更强大的学习框架。

7 结论

本项目探索了将心理理论（ToM）与大型语言模型（LLM）结合应用于机器人协作系统的新方法。截至目前，我们已经实现了一个完整的纯软件系统架构，包括感知与目标推理、ToM 推断、协作规划和虚拟环境模拟等核心功能模块。

初步测试表明，系统能够成功理解简单任务目标，生成合理的协作计划，并在虚拟环境中执行基本协作行为。尽管当前系统使用了模拟模型实现，但已经验证了整体架构的可行性和工作流程的正确性。

未来工作将聚焦于提升各模块功能，集成真实预训练模型，开展系统对比实验，以及探索在实体机器人上的部署可能性。我们期望这一研究能为未来更智能、更自然的人机协作方式提供新的思路和技术支持。

References

[1] Puig, X., Ra, K., Boben, M., Li, J., Wang, T., Fidler, S., & Torralba, A. (2018). *Watch-and-help: A challenge for social perception and human-AI collaboration*. arXiv preprint arXiv:1812.04707.

[2] Rabinowitz, N. C., Perbet, F., Song, H. F., Zhang, C., Eslami, S. M., & Botvinick, M. (2018). *Machine theory of mind*. In International conference on machine learning (pp. 4218-4227).

[3] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). *Language models are few-shot learners*. Advances in neural information processing systems, 33, 1877-1901.