CoLLMLight 專案程式碼說明文件

1. 專案概述

CoLLMLight 是一個基於大型語言模型（LLM）的網絡級交通信號控制系統。該系統通過協作式 LLM 代理框架來優化城市交通流量，減少擁堵。

1.1 主要特點

- 協作式 LLM 代理框架
- 結構化時空圖表示
- 複雜度感知推理機制
- 基於模擬的微調策略

2. 系統架構

2.1 目錄結構

```
.
├── models/              # 各種交通控制代理模型
├── framework/          # 核心框架實現
├── utils/             # 工具函數
├── data/              # 數據文件
└── media/             # 媒體資源
```

2.2 核心組件

2.2.1 模型組件 (models/)

- `CoLLMLightAgent.py` : 主要的 LLM 代理實現
- `colight_agent.py` : CoLight 算法實現
- `mplight_agent.py` : MPLight 算法實現
- `presslight_one.py` : PressLight 算法實現
- `maxpressure_agent.py` : MaxPressure 算法實現
- `network_agent.py` : 網絡代理基礎類
- `chatgpt.py` : ChatGPT 接口實現

2.2.2 框架組件 (framework/)

- `CoLLMlight.py` : 核心框架實現
- `FTSample.py` : 微調數據採樣實現

3. 主要功能模塊

3.1 交通控制代理

系統實現了多種交通控制策略：

1. **CoLLMLight**
   - 基於 LLM 的協作式控制
   - 動態複雜度感知推理
   - 鄰近路口協調
2. **傳統方法**
   - 固定時間控制 (FixedTime)
   - 最大壓力控制 (MaxPressure)
   - 隨機控制 (Random)
3. **深度學習方法**

- CoLight
- MPLight
- PressLight
- AttendLight

3.2 訓練與優化

- 模擬驅動的數據收集
- 環境反饋整合
- 輕量級 LLM 微調
- 時空圖構建與更新

4. 使用說明

4.1 環境要求

- Python >= 3.9
- TensorFlow-CPU = 2.8.0
- CityFlow
- 其他依賴：pandas, numpy, wandb, transformers, vllm, lmdeploy

4.2 運行方式

1. 部署 LLM 服務器： `shell lmdeploy serve api_server YOUR_LLM_PATH --tp=YOUR_GPU_NUM`
2. 運行 CoLLMLight：
   `shell python run_CoLLMlight.py --model_path=YOUR_LLM_PATH --dataset='newyork_28x7' --traffic_file='anon_28_7_newyork_real_doubl`

5. 性能特點

- 網絡級優化能力
- 適應性強
- 計算效率高
- 擴展性好
- 魯棒性強

6. 實驗場景

系統支持多種實驗場景：

- 合成數據測試
- 真實世界數據測試
- 不同交通流量條件
- 不同路網規模
- 不同時間段

7. 開發指南

7.1 添加新的控制代理

1. 在 `models/` 目錄下創建新的代理類
2. 繼承 `agent.py` 中的基礎類
3. 實現必要的接口方法
4. 在主程序中註冊新代理

7.2 數據處理

- 使用 `utils/` 中的工具函數處理數據
- 遵循 CityFlow 的數據格式規範
- 確保數據的時空一致性

7.3 數據格式說明

7.3.1 路網配置文件 (roadnet_*.json)

路網配置文件定義了交通網絡的基本結構，包含以下主要部分：

```JavaScript
{
    "intersections": [
        {
            "id": "intersection_1",
            "point": {"x": 0, "y": 0},
            "width": 10,
            "roads": ["road_1", "road_2", "road_3", "road_4"],
            "trafficLight": {
                "lightphases": [
                    {
                        "phase": ["NSG", "EWR"],
                        "time": 30
                    },
                    {
                        "phase": ["NSR", "EWG"],
                        "time": 30
                    }
                ]
            }
        }
    ],
    "roads": [
        {
            "id": "road_1",
            "points": [
                {"x": -100, "y": 0},
                {"x": 0, "y": 0}
            ],
            "lanes": 3,
            "startIntersection": "intersection_0",
            "endIntersection": "intersection_1"
        }
    ]
}
```

7.3.2 交通流量文件 (anon_*.json)

交通流量文件定義了車輛的行駛計劃：

```JavaScript
{
    "flow": [
        {
            "vehicle": {
                "length": 5.0,
                "width": 2.0,
                "maxPosAcc": 2.0,
                "maxNegAcc": 4.5,
                "usualPosAcc": 2.0,
                "usualNegAcc": 4.5,
                "minGap": 2.5,
                "maxSpeed": 11.111,
                "headwayTime": 2
            },
            "route": ["road_1", "road_2"],
            "interval": 3.0,
            "startTime": 0,
            "endTime": 3600
        }
    ]
}
```

7.3.3 狀態數據格式

系統運行時的狀態數據格式：

```JavaScript
{
    "time": 3600,
    "vehicles": {
        "vehicle_1": {
            "speed": 10.0,
            "position": {"x": 100, "y": 200},
            "road": "road_1",
            "lane": 1
        }
    },
    "intersections": {
        "intersection_1": {
            "phase": 0,
            "waiting_vehicles": 5
        }
    }
}
```

7.3.4 微調數據格式 (FinetuneData/*.json)

微調數據文件定義了用於訓練模型的問答對：

```JavaScript
[
  {
    "instruction": "你是一位交通號誌專家，負責管理一個四向交叉路口。你的主要任務是評估目前的協調程度，並實施合適的信號選擇策略。目標在於優化交通流與安全
    "input": "## 背景說明\n[路口配置和交通狀況的詳細描述]\n\n## 數據\n### 歷史觀測\n[包含多個時間點的交通數據表格]\n\n### 當前觀測\n[當前時間點的
    "output": "```json\n{\n  \"phase1\": {\n    \"thought_process\": \"思考過程描述\",\n    \"answer\": \"Simple\"\n  },\n  \"phase
  }
]
```

微調數據的主要組成部分：

1. **指令（instruction）**
   - 定義交通號誌專家的角色和任務
   - 設定優化目標和考慮因素
2. **輸入（input）**
   - 背景說明：路口配置和基本規則
   - 歷史觀測數據：包含多個時間點的詳細交通數據
   - 當前觀測數據：當前時間點的交通狀況
   - 信號優先順序：各種信號方案的優先級
3. **輸出（output）**
   - Phase 1：協調情境分析
     - 思考過程描述
     - 情境分類（無協調/簡單協調/複雜協調）
   - Phase 2：信號選擇策略
     - 決策過程描述
     - 具體信號選擇（ETWT/NTST/ELWL/NLSL）

主要數據文件：

- `syntrain_reasoning_tuning.json` ：推理能力訓練數據.
- `syntrain_refine.json` ：優化後的訓練數據.
- `example.json` 和 `example2.json` ：示例數據.

8. 注意事項

1. 運行環境需求
   - Linux 系統（推薦 Ubuntu）
   - 足夠的 GPU 資源
   - 穩定的網絡連接
2. 性能優化
   - 適當調整批次大小

- 優化模型參數
- 監控資源使用

3. 數據安全

- 定期備份實驗數據
- 保護模型檢查點
- 注意數據隱私