Báo cáo chuyên sâu: ALFWorld & PPO Agent

---

1. Giới thiệu & Cấu trúc codebase

1.1. Tổng quan

ALFWorld là một nền tảng nghiên cứu AI cho các tác vụ phiêu lưu dạng văn bản và embodied (3D), kết hợp hai môi trường: - **ALFRED (AI2-THOR):** Môi trường embodied 3D, agent tương tác vật lý với thế giới ảo. - **TextWorld:** Môi trường phiêu lưu dạng text, agent nhập lệnh và nhận phản hồi dạng văn bản.

ALFWorld cho phép chuyển đổi nhiệm vụ giữa hai môi trường, giúp nghiên cứu học tăng cường, học chuyển đổi, và khả năng hiểu ngôn ngữ tự nhiên của agent.

1.2. Cấu trúc thư mục chính

- **alfworld/**: Mã nguồn chính (agent, môi trường, module phụ trợ, dữ liệu, sinh dữ liệu).
- **scripts/**: Script huấn luyện, đánh giá, sinh dữ liệu, chơi thử nghiệm.
- **training/**: Checkpoint mô hình đã huấn luyện.
- **configs/**: File cấu hình YAML cho huấn luyện/đánh giá.
- **media/**: Hình ảnh minh họa, tài liệu.
- **eval_results.csv**: Kết quả đánh giá agent.
- **README.md, README_REPORT.md**: Tài liệu hướng dẫn, báo cáo tổng hợp.

Ví dụ cấu trúc alfworld/agents/:

```
alfworld/agents/
    agent/          # Định nghĩa các agent (PPO, DQN, DAgger, ...)
    controller/     # Agent điều khiển môi trường embodied (THOR)
    expert/         # Agent handcoded, oracle
    environment/    # Lớp môi trường (TextWorld, ThorEnv)
    modules/        # Module neural, buffer, memory, layers
    utils/          # Hàm tiện ích
```

---

2. Phân tích chi tiết pipeline code (Đào sâu)

2.1. Tổng quan pipeline

Pipeline của ALFWorld gồm các bước chính: 1. **Sinh dữ liệu & chuẩn bị môi trường** 2. **Khởi tạo agent và môi trường** 3. **Huấn luyện agent** 4. **Đánh giá agent** 5. **Lưu checkpoint, log, kết quả**

2.2. Luồng dữ liệu và các class then chốt

- **Sinh dữ liệu:**

  - Sử dụng các script như `augment_trajectories.py`, `generate_trajectories.py` để sinh ra các file `traj_data.json`, PDDL, layout, …
  - Dữ liệu này mô tả trạng thái ban đầu, kế hoạch expert, các hành động hợp lệ.
- **Khởi tạo môi trường:**
  - Class then chốt: `AlfredTWEnv` (TextWorld), `AlfredThorEnv` (THOR)
  - Các class này wrap lại môi trường gốc, cung cấp API reset, step, trả về quan sát, lệnh hợp lệ, reward, …
- **Khởi tạo agent:**
  - Agent được khởi tạo từ các class trong `alfworld/agents/agent/` (PPO, DQN, DAgger, …)
  - Agent nhận config, nạp checkpoint nếu có.
- **Huấn luyện:**
  - Agent tương tác với môi trường qua vòng lặp: quan sát → sinh lệnh → step → nhận reward → lưu transition.
  - Buffer lưu lại các transition để tính advantage (PPO), Q-value (DQN), hoặc dùng cho imitation (DAgger).
  - Định kỳ, agent cập nhật policy/network dựa trên dữ liệu trong buffer.
- **Đánh giá:**
  - Chạy agent trên tập nhiệm vụ mới, ghi lại reward, success rate, số bước.
  - Kết quả ghi vào eval_results.csv, log, checkpoint.

2.3. Ví dụ flow code huấn luyện PPO

```
# Khởi tạo môi trường và agent
config = generic.load_config()
agent = TextPPOAgent(config)
env = AlfredTWEnv(config, train_eval="train").init_env(batch_size=1)

for episode in range(num_episodes):
    obs, infos = env.reset()
    done = False
    while not done:
        action_candidates = infos["admissible_commands"]
        action, _, _, _, _ = agent.select_action(obs[0], task_desc,
action_candidates, None)
        obs, reward, done, infos = env.step([action])
        # Lưu transition, update policy định kỳ
```

2.4. Phân tích các hàm then chốt

- **select_action:** Nhận quan sát, nhiệm vụ, danh sách lệnh hợp lệ, trả về action tối ưu theo policy hiện tại.
- **preprocess_observation, preprocess_action_candidates:** Chuẩn hóa text đầu vào, sinh vector embedding.
- **update:** Cập nhật policy network theo thuật toán PPO/DQN/DAgger.
- **reset, step (env):** Khởi tạo lại môi trường, thực hiện một hành động.

2.5. Flow dữ liệu thực tế

- Dữ liệu từ file json/pddl → môi trường → agent nhận text → agent sinh lệnh → môi trường trả về trạng thái mới, reward → agent lưu lại → update policy.
- Đánh giá: agent tự động chơi, log lại từng bước, ghi kết quả.

2.6. Đặc điểm nổi bật pipeline ALFWorld

- **Modular:** Dễ thay thế agent, môi trường, reward, buffer.
- **Hỗ trợ nhiều thuật toán:** PPO, DQN, DAgger, Oracle, Handcoded.
- **Tích hợp sinh dữ liệu, augment, kiểm tra replay.**
- **Có thể mở rộng sang môi trường embodied (THOR) nếu cài đặt thành công.**

---

3. Chi tiết về các loại agent (Đào sâu)

3.1. PPO Agent

- **Thuật toán:** Proximal Policy Optimization (PPO) là một thuật toán policy gradient hiện đại, cân bằng giữa hiệu quả cập nhật và sự ổn định.
- **Cơ chế:**
  - Agent sinh xác suất chọn từng lệnh hợp lệ dựa trên quan sát và nhiệm vụ.
  - Sử dụng buffer để lưu các transition (obs, action, reward, done, value, …).
  - Tính advantage (GAE), cập nhật policy bằng loss PPO (clip, entropy regularization).
- **Ưu điểm:**
  - Ổn định, dễ tune, hiệu quả với không gian hành động rời rạc lớn như text.
  - Có thể tận dụng reward shaping, curriculum learning.
- **Nhược điểm:**
  - Cần nhiều episode để đạt hiệu quả cao.
  - Dễ bị kẹt ở local optimum nếu reward quá thưa.

Ví dụ code update PPO:

```
def update(self):
    # Tính advantage, returns
    self.finish_path(last_value=0)
    # Lấy batch từ buffer
    obs, actions, returns, advantages = ...
    # Tính loss PPO
    loss = ... # policy loss + value loss - entropy
    loss.backward()
    self.optimizer.step()
```

3.2. DQN Agent

- **Thuật toán:** Deep Q-Network, học Q-value cho từng action.
- **Cơ chế:**
  - Agent dự đoán Q-value cho từng lệnh hợp lệ, chọn action greedy hoặc epsilon-greedy.

  - Replay buffer lưu transition, cập nhật Q-network qua Bellman update.
- **Ưu điểm:**
  - Hiệu quả với không gian hành động rời rạc.
  - Có thể kết hợp với exploration bonus.
- **Nhược điểm:**
  - Dễ bị overestimation, cần tune kỹ replay buffer, target network.
  - Không phù hợp với không gian hành động quá lớn hoặc liên tục.

3.3. DAgger Agent

- **Thuật toán:** Dataset Aggregation (Imitation Learning).
- **Cơ chế:**
  - Agent học từ expert (oracle), thu thập dữ liệu từ cả expert và policy hiện tại.
  - Cập nhật policy để bắt chước expert.
- **Ưu điểm:**
  - Học nhanh, hiệu quả khi có expert tốt.
  - Giảm exploration cần thiết.
- **Nhược điểm:**
  - Phụ thuộc vào chất lượng expert.
  - Không tự khám phá được các chiến lược mới.

3.4. VisionDAgger Agent

- **Kết hợp quan sát hình ảnh (frame) với text.**
- **Phù hợp với môi trường embodied (THOR).**

3.5. Oracle, Handcoded Agent

- **Oracle:** Sử dụng thông tin hoàn hảo từ môi trường, chỉ dùng để đánh giá upper bound.
- **Handcoded:** Dựa trên heuristic, rule-based, không học.
- **Ưu điểm:**
  - Đánh giá giới hạn tối đa của môi trường.
  - Làm baseline cho các agent học.
- **Nhược điểm:**
  - Không tổng quát hóa, không thích nghi với nhiệm vụ mới.

3.6. So sánh các agent

Agent	Ưu điểm	Nhược điểm
PPO	Ổn định, tổng quát tốt	Cần nhiều episode, tune khó
DQN	Đơn giản, hiệu quả action rời rạc	Dễ overfit, không hợp action lớn
DAgger	Học nhanh, ít cần explore	Phụ thuộc expert, không tự khám phá
Oracle	Upper bound, baseline	Không thực tế, không học

Agent	Ưu điểm	Nhược điểm
Handcoded	Đơn giản, dễ debug	Không tổng quát, rule-based

---

4. Dữ liệu & config

4.1. Dữ liệu

- **PDDL, traj_data.json:** Định nghĩa nhiệm vụ, trạng thái ban đầu, kế hoạch expert.
- **Layout, objects:** Định nghĩa cấu trúc phòng, vật thể (alfworld/gen/layouts/).
- **Checkpoint .pt:** Trạng thái mô hình đã huấn luyện (training/).

Ví dụ file traj_data.json:

```
{
  "scene": {"scene_num": 401, ...},
  "plan": {"low_actions": [...], ...},
  ...
}
```

4.2. Config

- **base_config.yaml, eval_config.yaml:** Cấu hình huấn luyện, đánh giá (batch size, learning rate, reward, …).
- **alfred.pddl, alfred.twl2:** Định nghĩa logic nhiệm vụ, grammar sinh text.

Ví dụ đoạn config:

```
general:
  use_cuda: false
  save_path: training/
  training_method: ppo
```

---

5. Phân tích log, kết quả, và các script hỗ trợ

5.1. Kết quả đánh giá

- **eval_results.csv:** Ghi lại reward trung bình, success rate, số bước trung bình cho từng checkpoint.
- **Log huấn luyện:** In ra reward, loss, số bước, checkpoint định kỳ.

Ví dụ eval_results.csv:

Checkpoint	Reward TB	Success rate	Steps TB
test_ep0.pt	0.00	0.00	29.0
test_ep5000.pt	0.18	0.06	27.7
test_ep10000.pt	0.63	0.28	26.5
test_ep15000.pt	0.78	0.35	26.5

Checkpoint	Reward TB	Success rate	Steps TB
test_ep20000.pt	0.80	0.40	26.0

5.2. Script hỗ trợ

- **augment, replay, check:** Đảm bảo dữ liệu hợp lệ, augment trajectory, kiểm tra khả năng replay.
- **utils:** Hàm tiện ích cho xử lý text, sinh lệnh hợp lệ, quản lý buffer, …

---

6. Các vấn đề thực tế khi build, cài đặt, chạy thử

6.1. Lỗi thường gặp

- **Lỗi thư viện:** AI2-THOR, maskrcnn, vision… yêu cầu nhiều dependency hệ thống, dễ xung đột trên WSL2/Linux.
- **Lỗi bộ nhớ:** Chạy mô hình lớn, load checkpoint có thể gây bus error, out-of-memory.
- **Lỗi CUDA/GPU:** Không tương thích driver, thiếu GPU, hoặc cài bản CUDA không phù hợp.

6.2. Kinh nghiệm thực tế

- **Nên bắt đầu với TextWorld:** Dễ cài đặt, debug, huấn luyện nhanh.
- **Cẩn thận khi chuyển sang THOR:** Cần chuẩn bị môi trường hệ điều hành, driver, GPU, Unity3D.
- **Nên sử dụng Docker hoặc máy ảo chuẩn hóa:** Giảm thiểu xung đột thư viện.
- **Đọc kỹ log, kiểm tra từng bước:** Đặc biệt khi augment dữ liệu, replay, hoặc debug agent.

---

7. Kết luận & đề xuất mở rộng

- Đã xây dựng thành công agent PPO cho ALFWorld, reward và success rate tăng ổn định.
- Hạn chế: success rate chưa cao do môi trường phức tạp, reward sparse.
- Đề xuất: kết hợp thêm imitation learning, reward shaping, curriculum learning.

---

8. Tài liệu tham khảo

- Mã nguồn dự án alfworld
- ALFWorld: Aligning Text and Embodied Environments for Interactive Learning
- PPO: Proximal Policy Optimization Algorithms (Schulman et al., 2017)