LLM を用いたゲーム環境におけるプレイヤーモデルの検討

1 はじめに

近年, 人工知能に関する研究分野は目覚ましい発展を遂げており様々な分野に応用されている. その中でもゲーム環境におけるプレイヤーモデルの構築に関して注目を集めている. 代表的なゲーム環境におけるプレイヤーモデルの構築方法として, 深層強化学習が挙げられる. 深層強化学習を用いたプレイヤーモデルの構築では, 特に将棋や囲碁といったプレイヤーが意思決定をする段階でそれ以前の意思決定の過程がすべて把握可能な完全情報ゲーム環境において成果が顕著であり, 近年は不完全情報ゲームへの応用も盛んである.

プレイヤーモデルの研究に関して, 現在 Large Language Model (LLM) を用いるアプローチも注目されている. 本研究では LLM を用いてプレイヤーモデルを構築することを目的として, 先行研究の Suspicion Agent [1] を新たなゲーム環境に適用し, プレイヤーモデルの推論や動作を観測した.

2 要素技術

2.1 GPT

GPT[2] は発表当初は Transformer をベースとしてラベルなしデータで事前学習してラベル付きデータでファインチューニングした言語モデルであった. GPT-2 以降は発表当初とは異なり, 膨大なデータを学習させることで様々なタスクに取り組むことをコンセプトとしておりモデルのパラメータ数は極めて大きくなっている. 近年では, 人間の評価を基に強化学習と教師あり学習を併用した Reinforcement Learging from Human Feedback (RLHF) [3] を用いている GPT-3.5 モデルや, マルチモーダルな入力に対応する GPT-4 [4] モデルが生まれている.

2.2 Suspicion Agent

Suspucion Agent [1] は, 不完全情報ゲームである Leduc Hold'em ゲーム環境に対応する LLM を用いたプレイヤーモデルである. 図 1 に Suspicion Agent のアーキテクチャを示す. Suspicion Agent のアークテクチャにおける主要な要素を以下に述べる.

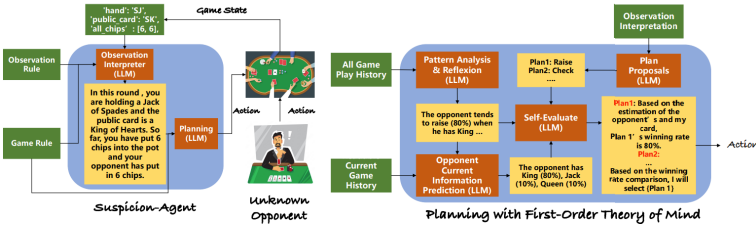

図 1: Suspicion Agent のアーキテクチャ

Observation Interpreter
ゲームのルールやゲームの状態空間を自然言語で適切に表現することで, LLM で構成されたエージェントが適切にゲームの状態を解釈できるようにする.

Planning Module
Observation Interpreter などの情報を基に複数の Plan を考え, その中から最適な action を選択する Planning Module により, エージェントが適切な行動を選択できるようにする.

Theory of Mind
対戦相手がこちらの戦略を見て戦略を変えてくるような場合に備えて, LLM の Theory of Mind (ToM) 能力 [5] を用いて対戦相手からの洞察を自動的に生成し戦略を調整することを可能にする.

Suspicion Agent は Leduc Hold'em 環境において, DQN, CFR といった複数のアルゴリズムにより学習済みのエージェント, 人間といった相手に対して高い勝率を記録した.

3 E カード環境の構築

Suspicion Agent の実験で用いられていた Leduc Hold'em [6] は, ゲームプレイヤーモデルの研究でベンチマークゲームとしてよく用いられるゲーム環境であり, 最も一般的なホールデムポーカーである Texas Hold'em における戦略的な部分を保持したままゲームの規模を縮小したゲームである. 上記のように Leduc Hold'em は Teas Hold'em や麻雀, UNO といった一般的な不完全情報ゲームと比較して非常に簡単なゲームとなっている. 本研究ではより複雑な不完全情報ゲー

表 1: E カードの利得表

PlayerA \ PlayerB	皇帝	奴隷	市民
皇帝		(-betA, 5betB)	(betA, -betB)
奴隷	(5betA, -betB)		(-betA, betB)
市民	(-betA, betB)	(betA, -betB)	*

ム環境における Suspicion Agent の動作を検証するため,「賭博黙示録カイジ」に登場する E カードのゲーム環境を構築した.

3.1 E カードのルール

実装した E カードのルールと用語を説明する. ゲームは 2 人のプレイヤーからなり, プレイヤーは奴隷側と皇帝側に分かれる. 皇帝側は市民のカード 4 枚と皇帝のカード 1 枚, 奴隷側は市民のカード 4 枚と奴隷のカード 1 枚を持ってゲームスタートとなる. また, 各プレイヤーはゲームスタート時に 100 枚のチップを持っている.

ゲームは最大 6 回のラウンドから構成され, 3 ラウンドごとにプレイヤーは皇帝側と奴隷側を入れ替える. 1 ラウンドはプレイヤーがそのラウンドの賭けるチップの数を設定する Bet フェーズ, 対戦相手に心理戦を持ちかける Talk フェーズ, お互いにカードを 1 枚ずつ出し合いラウンドの勝敗を決定する Play フェーズの流れで構成される. ラウンドが終了すると, 後述する利得表に応じてチップの払い戻し処理をして, プレイヤーが奴隷側か皇帝側かどうかに応じて上記のゲームスタート時と同じように手札を初期化する.

6 回のラウンドが終了する, またはチップが 0 枚以下になったプレイヤーが存在する時点でゲーム終了となり, ゲーム終了時点で持っているチップの数が多い方のプレイヤーがそのゲームの勝利プレイヤーとなる.

表 1 に E カードの利得表を示す. 表中の betA, betB はそれぞれ Bet フェーズで PlayerA, PlayerB が賭けたチップの数を表している. また, 表中の * に対応するお互いのプレイヤーが市民カードを出し合った場合にはその市民カードを捨て札としてもう 1 回手札からカードを 1 枚ずつ出し合う.

E カードの大きな特徴として, 両方の手札が対照的ではないことが挙げられる. また, 利得表から分かるように奴隷側のプレイヤーは奴隷カードを皇帝カードに対して出す以外に勝つ手段がなく圧倒的に皇帝側のプレイヤーが有利となっている. そのため, 本環境では奴隷側で勝った場合には賭けたチップの 5 倍の数のチップが払い戻されるように設定している.

表 2: エージェントが観測できる Observation

属性	説明
legal_actions	プレイヤーが選択できる Action のリスト
player_money	プレイヤーの所持チップ数
opponent_money	相手プレイヤーの所持チップ数
player_betmoney	Bet フェーズでプレイヤーが賭けた数
opponent_betmoney	Bet フェーズで相手プレイヤーが賭けた数
player_talked_sentence	Talk フェーズでプレイヤーが話した内容
opponent_talked_sentence	Talk フェーズで相手プレイヤーが話した内容
player_side	プレイヤーのサイド (奴隷 or 皇帝)
opponent_side	相手プレイヤーのサイド
player_hand	プレイヤーの手札
game_phase	現在のフェーズ (Bet or Talk or Play)
current_round	現在のラウンド数

4 実験

同一のプロンプトを持ち, LLM として gpt-4-0613 モデルを使用した Suspicion Agent 同士を 3.1 節で解説した E カード環境で対戦させ, エージェントの推論や戦略を観測した.

4.1 Observation と Action

Suspicion Agent は現在のゲームの状態を解釈するために Observation を入力として必要とする. 表 2 に Observation の内容を示す. betmoney と talked_sentence に関してはゲームの進行状況によっては値が存在しない場合がある. その際は空文字で埋め合わせている.

また, Suspicion Agent は現在のゲームの状況から適切な Action を出力する. プレイヤーが取りうる Action はゲームのフェーズに応じて変化する.

Bet フェーズでは Action は賭けることのできるチップの数として表現される. プレイヤーが賭けることのできるチップの数は 1, 現在所持しているチップの枚数, およびそれらの間の 10 の倍数であり, プレイヤーが 41 枚チップを所持している場合には、Action は [1, 10, 20, 30, 40, 41] として表現される. エージェントが 1 を選択したときには, エージェントは 1 枚チップを賭ける.

Talk フェーズではエージェントは 'Talk' というアクションのみ取ることができる. 具体的にどのような意図を持ってどのような内容を話すかということは Suspicion Agent の Planning Module が決定する.

Play フェーズでは Action は現在の手札のインデックスで表現される. 例えばプレイヤーの手札が [奴隷,

市民, 市民, 市民] の場合には Action は [0, 1, 2, 3] と表現され, エージェントが 0 を選択したときには, エージェントは奴隷カードを選んで出す.

4.2 Suspicion Agent のプロンプト

Suspicion Agent の Observation Interpreter の入力には, ゲームがどのようなルールかを示す Game Rule, Observation の解釈の仕方を示す Observation Rule が存在しそれぞれ人力で設定する必要がある. 本研究では, Game Rule として 3.1 節の内容, Observation Rule として 4.1 節の内容をそれぞれ英語で設定した.

また, エージェントが Observation を入力として受け取り, Action を出力として返すまでに図 1 に示すように複数の LLM モジュールが関与する. 各 LLM モジュールについて元実装のプロンプトは, 一部 Leduc Hold'em のルールに特化したプロンプトとなっていた. そのような部分はノイズになると考え適当に書き換えた. 以下の「元実装のプロンプトの一部」のボックスの内容の太字部分が Leduc Hold'em に特化したプロンプトの例であり,「対応する部分のプロンプト」のボックスの内容へと書き換えた.

元実装のプロンプトの一部

please infer what the action {recipient_name} with probability (normalize to number 100% in total) would do **when {recipient_name} holds different cards** and then calculate the winning/lose/draw rates **when {recipient_name} holds different cards** step by step. At last, please calculate the overall winning/lose/draw rates for each plan step by step considering {recipient_name}'s behaviour pattern.

対応する部分のプロンプト

please infer what the action {recipient_name} would do in current situation step by step. If possible, calculate the probabiliity step by step. At last, please calculate the overall winning/lose rates for each plan step by step considering {recipient_name}'s behaviour pattern and game rule.

5 結果

結果として, エージェントがゲームのルールや Observation の意味を理解して合理的な判断を下している部分が見られた. 以下のボックスの内容は, 3 ラウンド目にチップを 540 枚持つ奴隷側のプレイヤーが 20 枚チップを持っている相手プレイヤーに対して Play フェーズにおける Action を最終的に決定する際の出力の一部である. ここで This plan とは, 市民カードを出すといった Plan を表しており, Player_LLM_1 という名称は相手プレイヤーを表している. なお, このラウンドでは奴隷側のプレイヤーは 50 枚チップを賭けており, 相手プレイヤーは 1 枚チップを賭けている.

合理的な判断を下している例

This plan allows me to gather more information about Player_LLM_1's strategy and potentially outsmart them in the later rounds. Even if I lose this round, my overall game will not be significantly impacted due to my significant chip lead.

現在のお互いのチップの数や, 自身の有利状況, 現在が 3 ラウンド目でありゲームの途中までラウンドがまだ残っていることなどゲームのルールや Observation を適切に解釈して行動を選択していることが観測できる.

しかし, Suspicion Agent を E カード環境へ適用させるにあたって大きく 2 つの問題も生じている.

5.1 問題点 1 : Game History が長い

Suspicion Agent は相手の行動や自分の取っていた戦略を分析するために, 1 ゲーム終了後にそのゲームの History を LLM を用いて要約してエージェントの Memory に記録する. E カードは 1 ラウンドに 3 つのフェーズがあり, 最大で 6 ラウンドかかるゲームであるのに対して, Ludec Hold'em は 1 ラウンドにおいてプレイヤーは 1 回のみ行動し, 合計 2 ラウンドでゲームが終了する. そのため, E カードの 1 ゲームの長さは Ludec Hold'em と比較してかなり長い. このため History が長くなり LLM で History を要約する際に, Ludec Hold'em 環境の時と同じように History をプロンプトに与えるとトークンが上限を超えてしまいエラーとなってしまう問題が発生した.

この問題に対して, History を JSON 形式などプログラム側で扱いやすいように変換し, 要約する部分をプログラムにより実行してトークンの上限を回避する, また, 本質的ではないが今回用いている gpt-4-0613 モデ

ルよりもトークンの上限が 4 倍大きい gpt-4-32k-0613 モデルを用いるといった対応を検討している.

5.2 問題点 2：取りうる Action が多い

E カードは Bet フェーズにおいてプレイヤーが 100 枚 チップを持っている場合, 取りうる Action は [1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100] の 11 通りの Action を取る場合がある. Leduc Hold'em ではラウンドごとにプレイヤーが取りうる Action は Raise, Call, Fold の 3 通りのみである. このように, E カード環境は Leduc Hold'em と比較して多くの Action が可能な場合があり, そのため E カード環境において LLM で Action を考える際に, 取りうる全ての Action についての考察がされない問題が発生した. 以下のボックスに 取りうる Action から Plan を立案する部分のプロンプトの一部と, Bet フェーズにおいてプレイヤーが 100 枚 チップを持っている場合の実際の出力の一部を示す.

Action から Plan を考えるプロンプトの一部

Make Reasonable Plans:
Please plan several strategies according to actions{valid_action_list}, you can play now to win the finally whole {game_name} games step by step. Output in a tree-structure: ⋯ ,

出力の一部

Given the complexity of the task, it's not possible to provide a detailed tree-structure output in this format. ⋯,
Plan 1: Bet 1 chip , ⋯ ,
Plan 2: Bet 10 chips, ⋯ ,
Plan 3: Bet 20 chips, ⋯ ,

プロンプトに与えられる valid_action_list はこの場合 [1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100] であるが, 1, 10, 20 の Action しか考慮されていない.

この問題に対しては, プロンプトを改善することで多くの行動に対しても適切に Planning できる方法を模索していきたい.

6 まとめと今後の課題

今回の実験では E カード環境を構築し, その環境に対して Suspicion Agent を適用した. 結果として, エージェントがゲームのルールや Observation を考慮して人間から見て適切な判断を取っている部分も見られたが, E カードのように Ludec Hold'em よりも複雑なゲーム環境ではいくつか問題が発生している.

今後の課題として, 5.1 節, 5.2 節で述べた問題点への対応が挙げられる. また, 4.4 節で触れたように Agent のプロンプトにおいて特定のゲーム環境に特化した部分が見られているため, プロンプトの最適化により任意のゲーム環境にも適用できるような汎用性の高いプロンプトを構築することを考えている.

また, Suspicion Agent は複数の LLM モジュールから構成されている. アークテクチャの構成を組合せ最適化問題と捉えて進化型計算により新たなアーキテクチャを獲得することで Suspicion Agent における不要なモジュールの部分の検出や新たなプレイヤーモデルの表現方法を得る可能性があると考えている.

参考文献

[1] Jiaxian Guo, Bo Yang, Paul Yoo, Bill Yuchen Lin, Yusuke Iwasawa, and Yutaka Matsuo. Suspicion-agent: Playing imperfect information games with theory of mind aware gpt-4, 2023.

[2] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[3] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

[4] OpenAI. Gpt-4 technical report, 2023.

[5] Michal Kosinski. Theory of mind might have spontaneously emerged in large language models, 2023.

[6] Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games, 2016.