ロールプレイングゲームの画面情報分析による選択可能テキストの抽出

狩野竜示 [1]　森友亮 [1]　荒牧岳志 [1]
[1] 株式会社スクウェア・エニックス
{kanryuji,moriyus,taramaki}@square-enix.com

概要

本研究では，マルチモーダル大規模言語モデルを用いてロールプレイングゲーム（RPG）の画面から，選択対象のテキストを抽出する手法を提案し，その精度を検証した．テキスト情報を多く使用するビデオゲームのジャンルである RPG の自動プレイを機械学習モデルが行うには，画面に表示されたテキスト情報の処理能力が必要となる．本研究では，RPG の自動プレイに必要なサブタスクの一つとして，選択肢の提示された画面から，選択可能なテキストを行列形式で抽出する手法を提案する．また，データベースとの照合によるフィードバック手法を導入し，精度が向上することを確認した．

1　はじめに

ビデオゲームを自動プレイさせるタスクは，強化学習モデルの問題解決能力を計るタスクとして長らく研究題材とされてきた [1]．学術分野においてこうした研究はモデルの能力の検証といった色彩が強いが，ゲーム産業分野において，自動プレイは，テストプレイ，Quality Assurance（QA）, 難易度自動調整, さらには新しいゲームデザインの提唱など様々な応用範囲が期待できる．

ロールプレイングゲーム（RPG）は，ビデオゲームの一つの大きなジャンルであるが，従来の強化学習研究の対象にされることは少なかった．その理由として，RPG が高度なテキスト処理を必要とする点にある．近年マルチモーダル大規模言語モデル（LLM）の隆盛と，そのエージェントとしての応用の研究が盛んになるにつれ，アクション RPG などのテキスト処理能力を必要とするビデオゲームの自動プレイを行う LLM エージェントの研究が見られるようになってきた [2, 3, 4].

RPG には，画面上に表示されるテキストを選択することにより，プレイヤーが次の行動を決定する場面が多く存在する．RPG における，テキスト選択画面の例を図 1，2 に示す．これらの例では，カーソルが白い手として表現されており，図 1 においては，ミスリルソードを，図 2 においては，ねぶくろを指している．カーソルで選択されたテキストを選択 (この例では A ボタンの押下) することで，武器の装備や，アイテムの使用といった行動をプレイヤーは取ることができる．これらの操作を自動化することは，RPG の自動プレイにおける重要なサブタスクである．

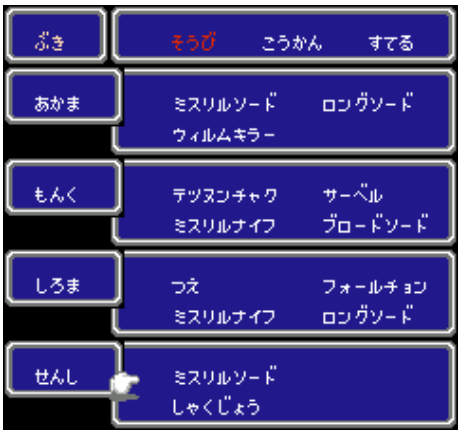

図 1　所持武器画面 © SQUARE ENIX

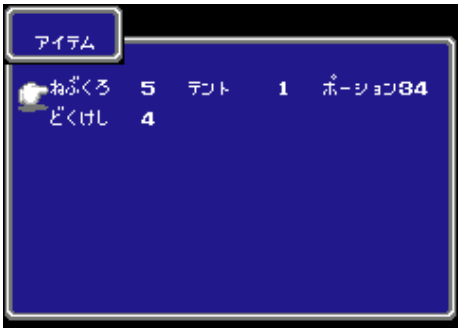

図 2　所持アイテム画面 © SQUARE ENIX

本研究では，ゲームの画面からゲーム画面上に記されたテキストの内，ユーザーが選択可能なテキス

トを抽出する手法を提案し，その精度を検証する．また，テキストを抽出する際，そのテキストの選択に至るためのボタン操作が把握可能な形で抽出するために, 選択が可能なテキストを行列形式で抽出する．手法としては，まず選択可能なテキストの行数と列数を推定し，推定した行数と列数，及びゲーム画面をマルチモーダル LLM に入力し，選択可能なテキストを行列形式で出力する．

本研究の貢献は以下である．

1. ゲーム画面から選択可能なテキストを抽出する手法を提案した．
2. 上記手法の認識精度の検証を，RPG であるファイナルファンタジーにおける武器防具及び，アイテム表示画面において行った．

2 関連研究

ゲームは従来より強化学習の研究対象とされており，囲碁，将棋などのボードゲームへの応用研究が広く知られている [5, 6]．ビデオゲームにおいては，アクションゲームやシューティングゲーム，パズルゲームが研究対象となっている [1]．

RPG のように自然言語理解を要するゲームジャンルは，従来の強化学習研究の対象とはあまりされなかったが，近年では，マルチモーダル LLM をエージェントとして応用する研究が盛んになるにつれ，テキストを利用したビデオゲームの自動プレイの研究が見られるようになってきた．Cradle [3] は Red Dead Redemption 2[1] を対象とした実験を行っており，テキストを含むゲーム画面の情報用いて次に行うべき行動を決定している．また Minecraft [2] を用いた研究として，Voyager [2] は，ゲームの情報をテキストとして入力し，次にとるべき行動をあらかじめ用意された関数群の中から選択している．Black Myth: Wukong [3] 用いた研究として，VARP [4] は，人のゲームプレイログを用いて，複数のキー操作を関数として定義する手法を用いている．

ゲームを自動プレイするエージェントの実装には，状態と行動を定義する必要がある．状態の計算のためには，ゲーム内の情報を取得する必要があるが，それには大別して以下の 2 種類の方法がある．

- API などを用いて直接取得する方法
- ゲーム画面の画像から取得する方法

前者には PokeLLMon [7] や Voyager [2] などの Minecraft の研究例があり，後者の例として，Cradle [3] や VARP [4] が存在する．一般にゲームの画面のスクリーンショット一枚からでは，ゲームの情報を十分に取得することができないため，複数の画像を使用することが殆どである．しかし，必要な画像の枚数や最適な取得頻度を決定するのは困難であり，かつ計算資源的な制約から，モデルに入力できる画像の量には限りがある．このため，アプローチとしての難易度は，後者の方が高い．ただし，API などを通じて，ゲームとエージェントが相互作用できる環境を用意することは，通常のゲーム開発工程には存在しない余分な工数のかかる作業であり，汎用的なエージェントの開発のためには，後者の技術が不可欠である．本研究でも，ゲーム画面の画像からゲームの情報を取得するアプローチを取る．

ビデオゲームにおけるエージェントの状態の定義方法は大別して 2 種類あることを上述したが，行動の定義方法も同様に以下の 2 種類に大別できる．

- キーボードやコントローラーのボタン押下を 1 つの行動とする方法 (例： 1 回 A ボタンを押す)
- 一連のキーボードあるいはコントローラーの操作の組み合わせ (行動列) を 1 つの行動とみなす方法

前者の問題は，計算量が膨大になってしまう点にある．後者の行動列が n 回のボタン操作から成ると仮定すると，前者のボタン押下を行動とみなす手法は，ゲーム環境への行動の伝達と画像情報の取得の回数が n 倍になるため，単純計算で約 n 倍の時間がかかる．更に大きな問題として，前者の定義方法では，ボタンの種類の数のべき乗の行動の組み合わせが存在するため，探索空間が膨大になり，現実的に探索することが不可能になってしまうという問題が存在する．そのため，複数のボタン操作の組み合わせを 1 つの行動単位として使用する仕組みが複雑なエージェント操作には不可欠である．

Minecraft には，公開された API が存在し，キャラクターの操作やゲームの情報の取得をプログラム上の関数を通じて行うことができる．Voyager [2] では，これらの API の関数を更に組み合わせたものを関数化する仕組みを構築している．Wukong Black Myth [4] の例では，人のプレイヤーのプレイログから，複数のボタン操作を関数化した．

1） Red Dead Redemption 2 は Rockstar Games, Inc. の登録商標です．
2） Minecraft は Mojang Studios の登録商標です．
3） Black Myth: Wukong は Game Science の登録商標です．

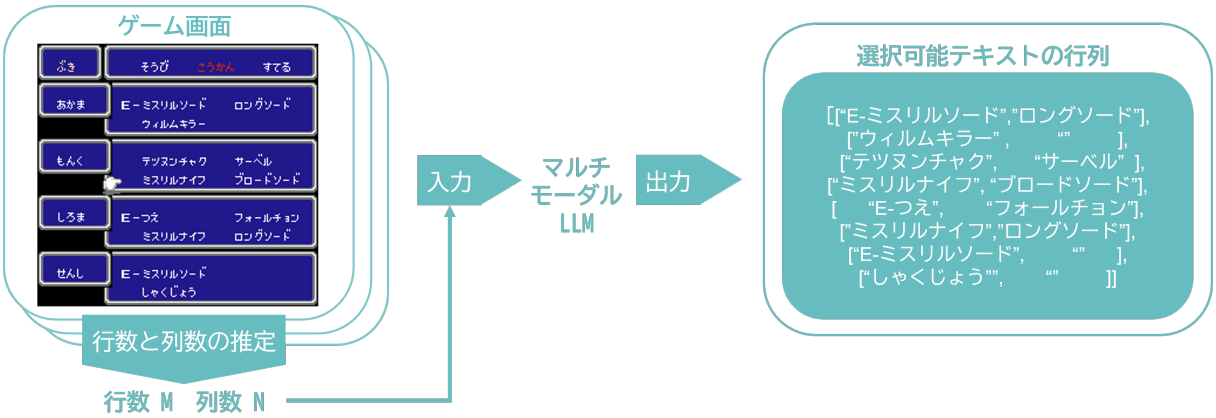

図 3 タスクの模式図 © SQUARE ENIX

3 タスク設定

本研究では，RPG の画面において表示されたテキストの内，カーソルで選択可能な文字列を行列形式で抽出するタスクを設定する．ビデオゲームの選択画面は，座標を指定してボタン押下をする形式のものもあるが，コントローラーを使用するゲームにおいては，コントローラーのボタン入力を用いて離散的に選択対象を決定するものが多い．ここでは特に，「上」「下」「左」「右」の 4 方向の操作のボタン群を持つ十字キーについて考える．また，選択可能なテキストは行列形式で表現可能である．例えば，図 1 において，画面上部にある「そうび」，「こうかん」，「すてる」は 1x3 の行列として表現できる．また，その下部にある「ミスリルソード」などの武器名が並んだ箇所は画面上では 2x2 の行列が 4 つ縦に並んでいるが，十字キーで各長方形の間を移動できるため，十字キーで移動可能な範囲は，8x2 の行列として表現できる．また, 図 1 では，空白部分の箇所もカーソルの選択対象に入る．

本研究では，以下の 2 つのタスクにおいて，性能を検証した．

- タスク 1．所持武器防具画面の認識
- タスク 2．所持アイテム画面の認識

両タスクの模式図を図 3 に示す．まず，カーソル移動によって，行数と列数の推定を行い，選択可能な対象のテキストを行列形式で，マルチモーダル LLM に出力させる．タスク 1．では，抽出対象の行列の行数と列数は常に 8x2 であり，武器や防具を所持していない箇所にもカーソルを移動可能である．評価データ数は 14 である，タスク 2．では，所持していない箇所にカーソルを移動することは出来ず，所持しているアイテムの数によって，行数と列数が変化する．図 2 の例は，2x3 である．評価データ数は 9 である．評価の方法として，以下の 2 種類を行う．

A) 出力行列の行数と列数が指定したものに合致する割合

B) 推定した行列の各要素に含まれる文字列の内，正解の文字列と完全一致する割合

4 手法

行数と列数の推定 本研究では，十字キーの操作による画面遷移の情報を用いて，選択対象行列の行数と列数を推定する．具体的には, 各画面において，決定ボタンを押した際のカーソルの最初の位置を選択対象行列の一行目一列目とする．そこで，「下」ボタンを連続して押下し，画面遷移が起こらなくなるまでに押下できた「下」ボタンの回数に 1 を足したものを行数とする．ここでは，既に取得した画像と類似度が一定値以上高くなる場合を，画面遷移が起こらないと定義する．同様に「右」ボタンを連続して押下し，列数を取得する．

使用モデルと比較実験の条件 行列出力を行うマルチモーダル LLM として，Qwen2-VL-72B-Instruct-GPTQ-Int4B [4] [8], 及び GPT-4o [9] を用いた．GPT-4o のバージョンは 2024-08-06 を用いた．

比較のため，カーソルの当たりうる箇所全ての画像を入力した場合と，一つのみ画像を入力した場合の性能比較を行った．

更に，武器や防具，アイテムのデータベースの名称と抽出した文字列を照合し, 名称が合わない場合には修正を行う手法も検証した．具体的には，マル

4) https://huggingface.co/Qwen/Qwen2-VL-72B-Instruct-GPTQ-Int4

チモーダル LLM からの出力の行列の各要素のテキストの内，データベース上に存在しない文字列をデータベース (DB) 上に存在する名称との類似度をレーベンシュタイン距離によって算出し，閾値以下のものに対しては，置換するプロンプトを，閾値以上のものに対しては，誤った箇所を抽出しているというプロンプトを作成し，最初に入力した結果とマルチモーダル LLM の出力に加え，再度マルチモーダル LLM に入力し，行列を出力させた．

また，異なる画像での正解行列を一つ入力プロンプトに加える，one-shot の in-context learning [10] も実験として行った．この時，プロンプトに入力する画像としては，対象が武器の表示画面である場合は，防具の表示画面を使用し，対象が防具の表示画面である場合は，武器の表示画面を使用した．また，対象がアイテムの表示画面である場合は，異なるアイテムの表示画面を用いた．

プロンプト　プロンプトはマルチモーダル LLM を問わず，英語で記述した．詳細は A.1 に記載する．

ゲーム画面の画像の詳細　本研究で利用した画面のスクリーンショットは，240x224 の画像であるが，OCR の精度向上のため，マルチモーダル LLM に入力する画像の解像度は 2 倍に拡張した．

5　結果と考察

表 1 と表 2 に，武器防具画面での行数列数の合致率と，OCR の精度を記す．DB はデータベース参照によるフィードバック，one-shot は DB のフィードバックに加え，入出力の具体例を一つプロンプトに加えた結果を示している．なお，DB 及び one-shot は複数画像入力の結果である.DB や one-shot を加えるに従い，各精度が向上している．行数と列数の精度が 1.0 になっていない条件があるが，これはプロンプトで行数と列数を指定した場合でも，指定の行列が出力できていない例があることを示している．また，武器防具画面において，全体としての OCR 精度は最も高い値でも未だ 3 割程度であり，低い値となっている．これは，低解像度でのフォント形式が特殊であることに加え，8x2 という大きな行列のどの位置にどのテキストが書かれているかを正確に把握することが難しいことを示している．これに反し，アイテム画面に対しての性能は最大で 5 割に達した．これは，アイテムにおいては，行数列数が武器防具と比較して少ないため，難易度が下がることを示唆している．

表 1　武器防具画面の選択行列抽出精度

モデル	条件	各推定精度 行数列数	OCR
GPT-4o	1 枚画像	0.50	0.22
GPT-4o	複数画像	0.93	0.16
GPT-4o	+DB	1.00	0.29
GPT-4o	+one-shot	1.00	0.31
Qwen2-VL-72B	1 枚画像	0.43	0.05
Qwen2-VL-72B	複数画像	0.86	0.06
Qwen2-VL-72B	+DB	0.57	0.04
Qwen2-VL-72B	+one-shot	0.71	0.21

表 2　アイテム画面の選択行列抽出精度

モデル	条件	各推定精度 行数列数	OCR
GPT-4o	1 枚画像	0.88	0.50
GPT-4o	複数画像	1.00	0.26
GPT-4o	+DB	1.00	0.34
GPT-4o	+one-shot	1.00	0.41
Qwen2-VL-72B	1 枚画像	0.00	0.00
Qwen2-VL-72B	複数画像	1.00	0.00
Qwen2-VL-72B	+DB	0.75	0.00
Qwen2-VL-72B	+one-shot	0.88	0.28

6　今後の課題

本研究では RPG を対象として，ゲーム画面における選択対象のテキストを取得する手法を提案し，その精度検証を行った．前提として，ゲーム画面における選択対象のテキストは行列形式で表現できると仮定し，まず十字キーの移動による行数と列数の推定を行い, 各行列の要素となるテキストをマルチモーダル LLM にて抽出した．

ただし，ゲーム画面においては，十字キーの移動によって，離散的に選択肢を選択できるとは限らず，カーソルやタッチパッドを使った連続的な座標を指定するゲームも存在する．これらについては，テキスト領域 (Bounding Box) を抽出する手法 [11, 12] が有効であると考えられるため，今後の課題としたい．また，カーソルの抽出に関しては，SAM2 [13] のような Segmentation 技術が応用可能と考えられる．今回行ったタスクはビデオゲームの自動プレイという目的の一部分のタスクである．今後本研究で培った技術を応用し，自動プレイへの応用を目指す．

参考文献

[1] Volodymyr Mnih, Koray Kavukcuoglu, et al. Playing Atari with Deep Reinforcement Learning. **NIPS Deep Learning Workshop 2013**, 2013.

[2] Guanzhi Wang, Yuqi Xie, et al. Voyager: An Open-Ended Embodied Agent with Large Language Models. **arXiv preprint arXiv: Arxiv-2305.16291**, 2023.

[3] Weihao Tan, Wentao Zhang, et al. Cradle: Empowering Foundation Agents Towards General Computer Control. **arXiv preprint arXiv:2403.03186**, 2024.

[4] Peng Chen, Pi Bu, Jun Song, et al. Can VLMs Play Action Role-Playing Games? Take Black Myth Wukong as a Study Case. **arXiv preprint arXiv:2409.12889**, 2024.

[5] David Silver, Julian Schrittwieser, Karen Simonyan, et al. Mastering the game of go without human knowledge. **Nature**, Vol. 550, No. 7676, pp. 354–359, 2017.

[6] David Silver, Thomas Hubert, Julian Schrittwieser, Antonoglou, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. **Science**, Vol. 362, No. 6419, pp. 1140–1144, 2018.

[7] Sihao Hu, Tiansheng Huang, and Ling Liu. PokeLLMon: A Human-Parity Agent for Pokemon Battles with Large Language Models. **arXiv preprint arXiv:2402.01118**, 2024.

[8] Jinze Bai, Shuai Bai, Shusheng Yang, et al. Qwen-VL: A Versatile Vision-Language Model for Understanding, Localization, Text Reading, and Beyond. **arXiv preprint arXiv:2308.12966**, 2023.

[9] OpenAI, et al. GPT-4o System Card. **arXiv preprint arXiv:2410.21276**, 2024.

[10] Tom B. Brown, Benjamin Mann, Nick Ryder, et al. Language models are few-shot learners. In **Proceedings of the 34th International Conference on Neural Information Processing Systems**, 2020.

[11] Pan He, Weilin Huang, Tong He, Qile Zhu, Yu Qiao, and Xiaolin Li. Single Shot Text Detector with Regional Attention. In **2017 IEEE International Conference on Computer Vision (ICCV)**, pp. 3066–3074, 2017.

[12] Youngmin Baek, Bado Lee, Dongyoon Han, et al. Character Region Awareness for Text Detection. In **2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**, pp. 9357–9366, 2019.

[13] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, et al. SAM 2: Segment Anything in Images and Videos. **arXiv preprint arXiv:2408.00714**, 2024.

A　参考情報

A.1　プロンプト

プロンプトは以下のように英語で記載した．

> “You are a helpful and harmless assistant. You have vast knowledge of Japanese Retro RPG games. You should think by step by step. In the given image, a cursor is pointing a text. The cursor is a white hand pointing to the right with a finger. Transcribe the texts pointed by the cursors in the images in the form of a matrix with {**n_row**} rows and {**n_col**} columns in python format. The texts are written in Japanese Hiragana, Katakana, Alphabets, and Arabic numbers.”

n_row，**n_col** には推定した行数と列数が入力される．データベース上に存在する文字列との照合のフィードバック用のプロンプトは以下とした．

> “The following terms are not in the database.
> {**list_not_in_db**}
> Do not transcribe that part of the image. The following terms might be mistranscription of the similar terms in the database.
> {**extracted_term**} -> {**db_term**}.
> Replace them with the correct terms.”

list_not_in_db は，抽出した行列の各要素に含まれる各テキストの内，武器防具アイテムデータベースの各名称とのレーベンシュタイン距離が閾値以下になるものが一つも無いものを指す．**extracted_term** と，**db_term** はそれぞれ，抽出した行列のテキストと，そのテキストとレーベンシュタイン距離が閾値以下になったデータベース内の名称の中で最も値が低いものを指す．上記プロンプトでは簡易的に一つのみを載せているが，実際には，**extracted_term** と，**db_term** は 複数存在しうる．