環境依存情報を利用しない大規模言語モデルによるコンピュータータスク自動化手法

笹川慶人　河原大輔
早稲田大学理工学術院
{kate@fuji., dkw@}waseda.jp

概要

大規模言語モデル (LLM) を用いたエージェントによってさまざまなコンピュータータスクを自律的に実行する手法が提案されている。先行研究で提案された手法の多くは HTML のコードなど環境に依存した情報を用いており、それらを用いることのできない環境では使用できず、汎用性に欠ける。本研究では環境に依存した情報を用いずにタスクを実行する手法を提案する。具体的には、コンピューター画面のスクリーンショットを画面上のテキストとユーザーインターフェースの要素の位置情報に変換し、その情報からマウスやキーボード操作のコマンドを LLM に生成させる。結果としてコンピューター画面の画像の入力のみでタスクを実行できるようになり、また、高いタスク成功率を示した。

1　はじめに

大規模言語モデル (LLM) はさまざまな分野で応用されており、自律的にコンピュータータスクを実行するエージェントとしても利用される。このようなエージェントは人間から与えられるタスクと現在のコンピューターの状態に基づいて、次に実行するべきアクションを予測し、タスクを完了する。既存の研究の多くは HTML のコードなど環境に依存した情報を LLM に入力することで現在のコンピューターの状態を認識させ、次に実行するべきアクションを予測させている [1, 2, 3]。そのような手法は特定の環境でしか適用することができない。例えば、HTML のコードは web ブラウザー以外のソフトウェアでは利用できないことが多い。

この問題に対処するため、本研究では特定の環境でしか利用することのできない情報を使わずに、コンピューター画面のスクリーンショットの内容を描写したテキストのみを使用してコンピュータータスクを実行する手法を提案する。スクリーンショットの処理には光学的文字認識 (OCR) モデルや物体検出モデルを使い、スクリーンショット上のテキストやユーザーインターフェース (UI) の要素の名称とその位置情報を示したバウンディングボックスに変換する。出力はマウスやキーボード操作のコマンドとなる。モデルの評価にはさまざまなコンピュータータスクからなる MiniWoB++ベンチマーク [4] を用いる。コンピューター画面のスクリーンショットのみを利用しているため他の環境でも使える手法であり、結果としてタスク成功率 9 割を達成した。

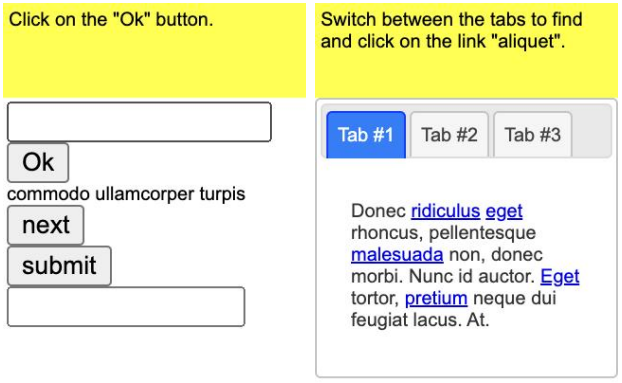

図 1　MiniWoB++のタスク例. 左:click-button, 右:click-tab-2.

2　関連研究

2.1　コンピュータータスクの自動化

web ブラウザーを使った日常的なタスクを集めたプラットフォームである World of Bits (WoB) [5] の一つに MiniWoB ベンチマークがある。このベンチマークにさらに難しいタスクや全く新しいタスクなどが導入された MiniWoB++ベンチマークがあり、コンピュータータスクのベンチマークとしてよく用いられている。MiniWoB++ベンチマークには、ボタンをクリックするタスクやテキストを入力するタスクなどの単純なタスクから複数回の画面の遷移を必

要とするような難しいタスクまでさまざまなタスクがある。それぞれのタスクにおいて seed 値を設定することでランダムにタスク環境が生成される。

このようなコンピュータータスクを自動で実行するモデルを構築する手法として強化学習を用いる手法、LLM を fine-tuning する手法、LLM の zero-shot や few-shot による手法がある。強化学習を用いる手法として CC-Net [6] や Pix2Act [7] があるが、モデルを学習するのに人間によるデモンストレーションデータが大量に必要である。LLM を fine-tuning する手法として WebGUM [8] や WebN-T5 [9] があるが、モデルを学習するコストがかかる。

これらに対して、LLM の zero-shot、few-shot による手法は必要なデモンストレーションデータの数が少なく、パラメータを更新する必要がないという利点がある。LLM の zero-shot による手法 [10] は人間によるデモンストレーションデータは全く使わないが、難しいタスクにおいて few-shot の手法より成功率が低い。LLM の few-shot による手法である RCI [1] では LLM に実行する予定のアクションのプランを再帰的に修正させることで精度を向上させている。AdaPlanner [2] は環境からのフィードバックによってプランを改善することにより精度を向上させている。Synapse [3] ではタスクの類似度によって LLM に対する入出力例 (exemplar) を選んで入力して精度を向上させている。これらの few-shot による手法はタスク成功率は高いが、HTML のコードを用いており、それが利用できる環境でのタスクにしか適用することができない。我々の提案手法は HTML のコードを用いていないため、さまざまな環境におけるタスクに適用することができる。

2.2 大規模言語モデルのマルチモーダル化

HTML のコードを使わずに画像を使用してコンピュータータスクを解くために、LLM に画像を認識させる必要がある。その手法として LLM とコンピュータービジョンモデル (CV モデル) を直接結合する手法がある。この手法では画像とテキストのアライメントをとる必要があり、BLIP-2 [11] では LLM と CLIP [12] の ViT [13] を Q-Former というモジュールで結合し、画像とキャプションペアのデータで学習してアライメントをとっている。BLIP-2 をマルチモーダル instruction tuning したモデルとして InstructBLIP [14] があり、性能が向上している。LLaVA [15] では LLM と CLIP の ViT を線形層で結合し、マルチモーダル instruction tuning することで画像とテキストのアライメントをとっている。

また、CV モデルの出力をテキストとして直接 LLM に入力する手法がある。MM-REACT [16] では ChatGPT[1)] を画像のキャプションモデルや OCR モデル等の CV モデルと組み合わせて画像に関する質問に答えられるようにした。また、LLaVA では画像のキャプションとバウンディングボックスを GPT-4 に入力することでマルチモーダル instruction tuning 用のデータを生成し、学習に利用している。

3 提案手法

3.1 概要

本研究では強力な LLM の推論能力を利用するために、コンピューター画面のスクリーンショットをテキストで表現して LLM に入力するという手法を採用する。図 2 に提案手法の概要を示す。まず、コンピューター画面のスクリーンショットを撮って、用意した CV モデルに入力し、CV モデルの出力、タスクごとに作成した exemplar、タスクの指示、今までの実行したアクション履歴を LLM に入力し、次のアクションを出力させる。この処理をタスクが完了するまで繰り返す。なお、画像を入力としてアクションを決定するモデルとしては 2.2 節で述べたマルチモーダルモデルを使う手法も考えられるが、zero-shot、few-shot で利用する場合はタスク成功率が低く、また、fine-tuning をする場合は学習データの用意やパラメータの更新などのコストがかかってしまうため採用しない。

3.2 テキストによる画像内容の描写方法

LLM にコンピューター画面のスクリーンショットを認識させるために、OCR と物体検出によってスクリーンショット上のテキストと UI の要素を検出し、その名称と位置を表すバウンディングボックスを入力する。OCR モデルは EasyOCR ライブラリ[2)] を利用して実装する。物体検出モデルとしては YOLOv8 を学習し利用する。YOLOv8 を学習するために、MiniWoB++ベンチマークのタスクのスクリーンショットにおいて、UI の要素の位置をアノテーションしたデータセットを構築する。まず、それぞれのタスクにおいて 10 種類の seed 値を設定して人

1) https://openai.com/chatgpt
2) https://github.com/JaidedAI/EasyOCR

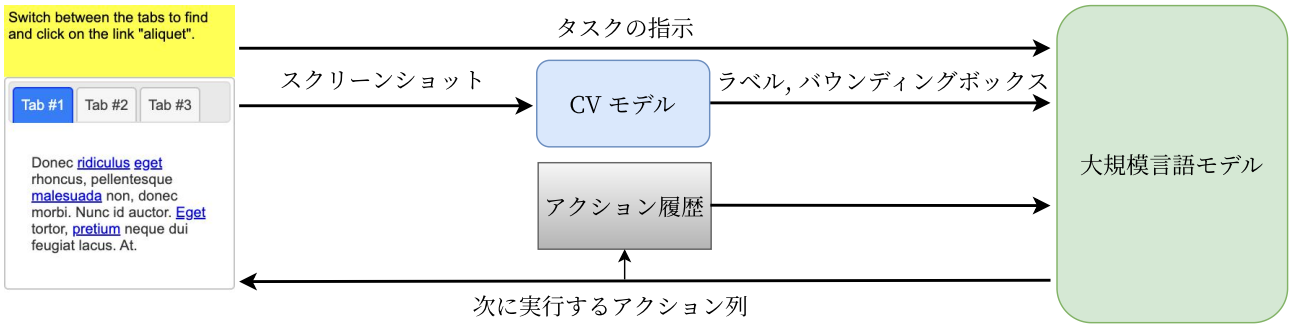

図 2　提案手法の概要

手でタスクを解き、画面の遷移が起こるごとにスクリーンショットを撮ることによって画像を収集する。ここで、テキストとしてタスクの指示が与えられるため、スクリーンショットのタスクの指示がかかれている黄色の部分はカットする。画像の解像度はデフォルトの 2 倍の解像度である 320x320 とした。収集した画像に対して UI の要素の位置とクラスをアノテーションした。クラスラベルをつけにくい UI の要素に関しては “other object” というラベルをつけ、後処理でキャプションを生成する。構築したデータセットのクラス数は 40、データ数は約 2.8k となった。このデータセットを用いて Ultralytics ライブラリ[3] を利用して YOLOv8 を学習した。

LLM のコンピューター画面の認識精度を上げるため、OCR と物体検出の結果に対して後処理を行う。まず、テキストを含む UI の要素は OCR の結果と組み合わせる。text button などの中に語句を含むオブジェクトは、‘Submit’ というテキストがオブジェクト内に検出された場合、‘Submit’ button などのように統合する。text area などの中に文や文章を含むオブジェクトは、中のテキストがわかるように、text area containing the text ‘text’ のようにする。また、“other object” というラベルがつけられたオブジェクトは InstructBLIP によってキャプションを生成してラベルとして利用する。

3.3　LLM へのプロンプト

タスク成功率を上げるため、LLM へのプロンプトとして ReAct [17] プロンプトを利用する。ReAct プロンプトは思考の出力、アクション生成および実行結果の観察をテキストで組み合わせ、タスク成功率を上げるというものである。具体的には Auto-GPT[4] のプロンプトに変更を加えたものを利用する。LLM に対する指示を記述するシステムプロンプトで、LLM の役割や制約、実行可能なアクション、バウンディングボックスの表現方法、出力形式、その他補助的な情報等を指示する。ユーザープロンプトはタスクの指示、過去に実行したアクション履歴、CV モデルの出力による現在のコンピューター画面情報とする。ここで、CV モデルの出力のバウンディングボックスは y 座標によってソートする。出力は思考過程と次に実行するアクション列となる。具体的な入出力例を付録 C に示す。

4　実験

4.1　実験設定

実験環境として、スクリーンショットを撮って座標を指定したアクションを実行することのできる MiniWob++ベンチマークを使用する。このベンチマークのタスクのうち 74 タスクにおいて評価を行う。実験に使用しないタスクは付録 A に示す。それぞれのタスクごとに exemplar を作成し、one-shot で入力する。LLM は OpenAI が API として提供している gpt-3.5-turbo、gpt-3.5-turbo-16k、gpt-4 を利用する[5]。gpt-3.5-turbo では、タスク完了までの画面の遷移回数が最小となるようにアクションを実行しても context size におさまらないタスクでは評価を行わない。評価指標としては、タスク環境から返される報酬が正の値であった割合で定義される成功率を用いる。ただし、次のアクションが決定されない、モデルの context size を超える、設定された max steps を超えるなどという事象が起きた場合は失敗として扱う。タスクごとに、gpt-3.5-turbo、gpt-3.5-turbo-16k では seed 値を 0-49 として 50 回実験し、gpt-4 では seed 値を 0-19 として 20 回実験する。その他の詳細設定は付録 B に示す。

3）　https://github.com/ultralytics/ultralytics
4）　https://github.com/Significant-Gravitas/AutoGPT
5）　https://platform.openai.com/docs/api-reference

表 1 全体の成功率とタスク数．本手法以外の結果は [1, 2, 3] より引用.

モデル	成功率の平均	タスク数
Ours (gpt-3.5-turbo)	0.874	60
Ours (gpt-3.5-turbo-16k)	0.824	74
Ours (gpt-4)	0.907	74
RCI (gpt-3.5-turbo)	0.906	54
RCI (gpt-4)	0.940	54
Adaplanner	0.929	53
Synapse	0.992	64

表 2 100 個の失敗したタスクの失敗原因

原因	gpt-3.5-turbo-16k	gpt-4
CV モデルの推論	21	42
LLM の推論	70	49
その他	9	9

4.2 実験結果

本手法と先行研究の few-shot の手法とを比較した結果を表 1 に示す。表において、成功率の平均はタスクごとの成功率の平均値であり、タスク数はそれぞれの手法において評価が行われたタスクの数である。画像を入力としていること、context size の大きいモデルを使用していること、先行研究の手法ではサポートされていなかったドラッグなどの操作をアクションに追加したことなどの要因から実行できるタスク数が増えた。 ただし、先行研究の手法では HTML のコードを利用しているため、評価する意味のないタスクにおいては評価を行っていないということが考えられる。また、本手法の gpt-3.5-turbo-16k の結果と gpt-4 の結果を比較すると、推論能力の高い gpt-4 の方がタスク成功率は高くなっていた。さらに、gpt-3.5-turbo と gpt-3.5-turbo-16k の結果を比較すると、gpt-3.5-turbo の方が解けるタスク数が減っているため、本手法はより大きな context size が必要であることがわかった。

4.3 結果の分析

gpt-3.5-turbo-16k と gpt-4 それぞれについて、失敗したタスクから 100 個ランダムにサンプリングして、その失敗原因を分析した。その結果を表 2 に示す。gpt-3.5-turbo-16k と gpt-4 を比較すると gpt-4 では LLM の推論による失敗が少なかった。CV モデルの推論の原因例として OCR モデルや物体検出モデルの推論が間違っていた場合や、キャプションモデルがわかりやすいキャプションをつけられなかった場合などが挙げられる。LLM の推論が原因で失敗したタスク例として、スクロールして目的のオブジェクトを探すタスクやオブジェクトの数を数えるタスク、オブジェクトの位置関係が重要で、バウンディングボックスによって正確に位置を認識しなければならないタスクなどがあった。その他の原因としてはオブジェクトをドラッグで入れ替えるタスクは判定が厳しく、失敗することが多かった。

表 3 タスクを解くのに必要なステップ数ごとの成功率

ステップ数	gpt-3.5-turbo-16k	gpt-4
1 ステップ	0.867	0.913
n(≥ 1) ステップ	0.723	0.893

表 3 にタスクを解くのに必要なステップ数ごとのタスク成功率を示す。1 ステップタスクは最初のスクリーンショットの情報のみからタスクを完了するためのアクションを全て生成できるタスクであり、n ステップタスクはアクションを実行した後の画面の変化を観測し、再びアクションを生成する必要があるタスクである。n ステップタスクの成功したタスクにおける平均ステップ数は gpt-3.5-turbo-16k では 3.024、gpt-4 では 3.089 であった。また、gpt-4 では n ステップタスクでもあまり成功率が下がっておらず、複数回の画面の遷移を経るタスクでも正しく推論ができていた。

gpt-3.5-turbo、gpt-3.5-turbo-16k 特有の失敗原因として、exemplar に影響されて失敗するというものがあった。例えば、図 1 の click-tab-2 というタスクにおいて、exemplar で “Tab #2” に目的のリンクがあった場合、“Tab #2” に目的のリンクがなかったとしても、“Tab #2” の別のリンクをクリックしてしまうという現象が起こった。gpt-4 では指定されたオブジェクト名と全く同じオブジェクト名でないとクリックしないという事象が見られた。

5 おわりに

本論文では画像のみを用いて座標を指定したマウスやキーボードのアクションを生成する手法を提案した。この手法は HTML のコードを利用せずタスクを解くことができるため、物体検出モデルを構築すれば、web ブラウザー以外のソフトウェアでも使える環境に依存しない手法となっている。今後はマルチモーダルモデルと本手法を組み合わせたい。また、より現実的な環境で実験を行いたい。

謝辞

本研究は SB Intuitions 株式会社と早稲田大学の共同研究により実施した。

参考文献

[1] Geunwoo Kim, Pierre Baldi, and Stephen Marcus McAleer. Language models can solve computer tasks. In **Thirty-seventh Conference on Neural Information Processing Systems**, 2023.

[2] Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. Adaplanner: Adaptive planning from feedback with language models. In **Thirty-seventh Conference on Neural Information Processing Systems**, 2023.

[3] Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In **NeurIPS 2023 Foundation Models for Decision Making Workshop**, 2023.

[4] Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. In **International Conference on Learning Representations**, 2018.

[5] Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In Doina Precup and Yee Whye Teh, editors, **Proceedings of the 34th International Conference on Machine Learning**, Vol. 70 of **Proceedings of Machine Learning Research**, pp. 3135–3144. PMLR, 06–11 Aug 2017.

[6] Peter C Humphreys, David Raposo, Tobias Pohlen, Gregory Thornton, Rachita Chhaparia, Alistair Muldal, Josh Abramson, Petko Georgiev, Adam Santoro, and Timothy Lillicrap. A data-driven approach for learning to control computers. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, **Proceedings of the 39th International Conference on Machine Learning**, Vol. 162 of **Proceedings of Machine Learning Research**, pp. 9466–9482. PMLR, 17–23 Jul 2022.

[7] Peter Shaw, Mandar Joshi, James Cohan, Jonathan Berant, Panupong Pasupat, Hexiang Hu, Urvashi Khandelwal, Kenton Lee, and Kristina Toutanova. From pixels to UI actions: Learning to follow instructions via graphical user interfaces. In **Thirty-seventh Conference on Neural Information Processing Systems**, 2023.

[8] Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust, Shixiang Shane Gu, and Izzeddin Gur. Multimodal web navigation with instruction-finetuned foundation models. arXiv, 2023. abs/2305.11854.

[9] Izzeddin Gur, Ofir Nachum, Yingjie Miao, Mustafa Safdari, Austin Huang, Aakanksha Chowdhery, Sharan Narang, Noah Fiedel, and Aleksandra Faust. Understanding HTML with large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, **Findings of the Association for Computational Linguistics: EMNLP 2023**, pp. 2803–2821, Singapore, December 2023. Association for Computational Linguistics.

[10] Tao Li, Gang Li, Zhiwei Deng, Bryan Wang, and Yang Li. A zero-shot language agent for computer control with structured reflection. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, **Findings of the Association for Computational Linguistics: EMNLP 2023**, pp. 11261–11274, Singapore, December 2023. Association for Computational Linguistics.

[11] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In **ICML**, 2023.

[12] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, **Proceedings of the 38th International Conference on Machine Learning**, Vol. 139 of **Proceedings of Machine Learning Research**, pp. 8748–8763. PMLR, 18–24 Jul 2021.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In **International Conference on Learning Representations**, 2021.

[14] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. InstructBLIP: Towards general-purpose vision-language models with instruction tuning. In **Thirty-seventh Conference on Neural Information Processing Systems**, 2023.

[15] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In **Thirty-seventh Conference on Neural Information Processing Systems**, 2023.

[16] Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. MM-REACT: Prompting chatgpt for multimodal reasoning and action. arXiv, 2023. abs/2303.11381.

[17] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In **International Conference on Learning Representations (ICLR)**, 2023.

A　実験の対象から除外したタスク

先行研究で除外していた debug tasks （choose-date-easy、choose-date-medium、click-tab-2-easy、click-tab-2-medium、click-test-transfer、email-inbox-delete、email-inbox-forward、email-inbox-important、email-inbox-noscroll、email-inbox-reply、email-inbox-star-reply)、即時的反応が必要なタスク (chase-circle、moving-items、simon-says)、selenium の仕様上できない動作が必要なタスク（choose-list、click-scroll-list、highlight-text、highlight-text-2、text-editor)、テキストによる指示が不明確なタスク (click-menu-2、use-colorwheel-2)、バウンディングボックスの情報からアクションを生成するという実験設定に沿わないタスク (grid-coordinate、guess-number、use-slider、use-slider-2、use-spinner) や exemplar を作成できなかったタスク（book-flight、drag-cube、drag-sort-numbers、number-checkboxes) も除外する。

B　実験の詳細設定

本実験で実行可能なコマンドは、マウスカーソルを動かす MOVE_COORDS、指定された座標をクリックまたはダブルクリックする CLICK_COORDS、DBLCLICK_COORDS、ドラッグの操作を実行する MOUSEDOWN_COORDS、MOUSEUP_COORDS、スクロールを実行する SCROLL_UP_COORDS、SCROLL_DOWN_COORDS、指定されたキーを押す PRESS_KEY、指定されたテキストをタイプする TYPE_TEXT がある。PRESS_KEY で使用できるキーボードのキーは"<Enter>"、"<Backspace>"、"C-a"、"C-c"、"C-x"、"C-v"のいずれかとした。OpenAI の API としては、gpt-3.5-turbo-0613、gpt-3.5-turbo-16k-0613、gpt-4-0613 を使用した。LLM の最大の completion の回数である max steps は 10 回に設定し、出力の max tokens は drag-shapes、choose-date は 1024、click-checkboxes-large、click-checkboxes-transfer は 1500、その他のタスクは 512 とした。また、completion のパラメーターは temperature を 0、その他の max tokens 以外のパラメーターはデフォルトの値を用いた。さらに、モデルの推論に時間がかかるため、タスクの制限時間はデフォルトの 100 倍の時間に設定した。

C　入出力例

図 1 における click-button タスクの入出力例を表 4、表 5 に示す。

表 4　click-button タスクにおける入力例

```
Task:
Click on the "Ok" button.

Observation 1:
OCR:
'commodo ullamcorper turpis': [3, 88,
260, 116]

Object Detector:
textbox: [4, 4, 283, 47]
'Ok' button: [4, 46, 69, 90]
'next' button: [4, 113, 86, 156]
'submit' button: [3, 155, 114, 198]
textbox: [3, 199, 256, 242]
```

表 5　click-button タスクにおける出力例

```
{
  "thoughts": {
    "text": "I need to click on the 'Ok'
button.",
   "reasoning": "Based on the Observation
1, the 'Ok' button is represented by ''Ok'
button: [4, 46, 69, 90]'.",
    "plan": "- Click on the center of the
'Ok' button.",
     "criticism": "I must click on the
center of the object.",
    "speak": "I will click on the center
of the 'Ok' button."
  },
  "commands": [
    {
      "name": "CLICK_COORDS",
      "args": {
        "coords": [36, 68]
      }
    }
  ]
}
```