

2025年度(令和7年度) 博士論文

LLMとRoBERTaに基づくテキスト
ゲームエージェントの実装と性能向上

2026年

鳥取大学大学院 工学研究科 工学専攻
自然言語処理研究室

D23T5002A ZHUO BINGGANG

概要

テキストゲーム（Text Game）エージェントの研究は、強化学習と自然言語処理の交差領域に位置し[1]、自然言語による記述を理解しつつ自律的にタスクを遂行できるエージェントの実現が重要な課題である。本研究は、文字によって構成されたゲーム環境においてタスクを自律的に解決し、タスク達成率を向上させるエージェントの設計と性能向上を目的とする。

本研究では、実験環境として2つの公開データセット TWC (TextWorld Commonsense の略称、タスクが比較的単純で小規模) および FTWP (First TextWorld Problems、タスクが複雑かつ大規模) を採用した。これらは難易度とルールが大きく異なる2種のテキストゲーム環境を代表するものである。環境特性の違いに応じて、我々は2種類のエージェント実装方式を提案する。

第一の方式は、大規模言語モデル（LLM）に基づき、精緻に設計したプロンプトによって行動を誘導するものである。この方式は実装が簡潔であり、基礎的アプローチとして適している。また、TWCのタスクは比較的容易であるため、LLMが十分に対応できると予想された。実験においても、LLMベースのエージェントは TWC で良好な性能を示した。しかし、この手法をより複雑な FTWP 環境に適用した場合、性能は大きく低下した。これは、プロンプト設計のみで微調整を行わない LLM が FTWP の複雑性を対応するには不十分であることに起因する。さらに、データセットが大規模であるため LLM の推論コストを無視できない問題もある。

これらの問題に対処するため、我々は第二の方式としてローカルモデルの微調整に基づくエージェントを提案する。FTWP データセットは大規模であるため、ローカルモデルを訓練するための十分なデータが存在する。しかし、ローカルモデルには多様な選択肢が存在するため、どのモデルを採用すべきかが課題となる。テキストゲームエージェントに最も重要なのは言語理解能力であることから、我々は「文章の自動強調表示（Automatic Document Emphasis）」タスクにおける性能を指標としてモデル選択を行った。このタスクは、正確な強調に文脈理解が不可欠であるため、言語理解能力の評価に適している。実験の結果、最も高い性能を示した RoBERTa[2] を微調整エージェ

ントの基盤モデルとして採用した。

以上より、我々は (1) LLM ベース、(2) ローカルモデル微調整ベースという 2 種のテキストゲームエージェントを構築し、それを単純環境および複雑環境に対応させた。さらにエージェント性能を向上させるため、以下の工夫を取り入れた。

LLM ベースのエージェントでは、TWC の特徴を踏まえ、chain-of-thought[12]・one-shot prompting[13]・出力テンプレート [14] など、プロンプト設計における重要手法を統合したプロンプトを構築した。加えて、ゲームからのフィードバックを明確化して LLM の行動選択を改善する feedback augmentation と名付けた新たなプロンプト技術を提案した。

ローカルモデルを微調整したエージェントでは、2 つの拡張モジュールを導入した。ひとつは誤り行動分類器である。FTWP には cookbook と呼ばれる明示的なタスク指示書が存在し、これに違反する行動はゲームの即時終了を引き起こす。我々は、訓練データの解析を通じて誤り行動データセットを自動構築し、誤り行動を識別する分類器を学習した。分類器により誤り行動と判定された行動の選択確率を低減することで、エージェントの失敗を防止する。

もうひとつの拡張モジュールはナビゲータ (Navigator) である。FTWP では膨大な数の部屋と物品が存在するため、物品の位置情報を記憶・参照できる補助ツールが有効である。ナビゲータは物品と部屋の対応を記録し、「物品ナビゲーション指令」をエージェントに提供する。エージェントがこの指令を選択すると、ナビゲータが対象物品の存在する部屋へ誘導する。この拡張により、エージェントの探索効率が大幅に向上了。

実験の結果、LLM と最適化プロンプトに基づくエージェントは、小規模データセット TWCにおいて easy: 1.00, medium: 1.00, hard: 0.70 のタスク達成率を示し、従来最良のエージェント (0.96, 0.85, 0.57) を上回った。

また、RoBERTa と 2 つの拡張モジュールを用いたエージェントは、FTWP テストセットにおいて 0.968 のタスク達成率を記録し、従来最良のエージェント (0.880) を大幅に上回った。

本研究は、二種類のエージェント実装戦略の性能を検証しただけでなく、テキストゲームエージェントの開発における方法論も提示する。すなわち、小規模データセットに対しては LLM の強力な事前知識を活用したエージェントを設計し、その有効性を検証する。一方で、LLM が性能面やコスト面で不十分な場合には、RoBERTa のような言語理解が得意なローカルモデルを用いたエージェントを実装する。さらに、その基盤の上で特定のゲーム環境に適した拡張モジュールを導入することで、より高い性

能を目指す。このような方法論は、今後のテキストゲーム研究に新たな枠組みを提供するものである。

目 次

第1章 はじめに	1
第2章 先行研究	5
2.1 データセットとベンチマーク	5
2.2 テキストゲームエージェント	6
2.3 プロンプトエンジニアリングと大規模言語モデル	8
2.4 行動消去	9
2.5 模倣学習と因果混同	9
2.6 ゲームとナビゲーター	10
第3章 LLMによるTWCゲームの効率的解法	11
3.1 TWCゲームの紹介	12
3.2 手法	16
3.2.1 プロンプト	16
3.2.2 フィードバック増強	18
3.2.3 APIでプロンプトを使用	19
3.3 実験	20
3.3.1 データセット	20
3.3.2 実験設定	20
3.3.3 ベースライン	21
3.4 実験結果	21
3.4.1 従来手法との比較	21
3.4.2 アブレーション実験と結果	22
3.5 考察	24
3.5.1 新しく作ったゲーム環境におけるGPT-4の性能	24
3.5.2 より広範な応用シナリオ	25
3.5.3 例の選択の重要性	25

3.5.4 ケーススタディ	26
3.6 結言	30
第4章 より複雑なゲームへ:RoBERTaの性能検証	31
4.1 タスク紹介:日本語文章の自動強調付与	31
4.2 関連研究	32
4.2.1 RoBERTa	32
4.2.2 タイトル情報	33
4.2.3 CRF	34
4.3 手法	34
4.3.1 RoBERTaにおけるタイトル情報の導入	35
4.3.2 RoBERTa-CRF	36
4.3.3 日本語データセット向けの事前トークン化処理	37
4.4 実験	37
4.4.1 データセット	37
4.4.2 指標	38
4.4.3 モデルの実装	38
4.4.4 ベースライン	38
4.4.5 パラメータ設定	39
4.4.6 アブレーション実験	39
4.5 実験結果	40
4.6 考察	41
4.6.1 人手実験とLLM実験	41
4.6.2 空タイトル実験	47
4.7 結言	49
第5章 誤り行動分類器でFTWPエージェントの性能向上	50
5.1 FTWPゲームの紹介	51
5.2 手法	52
5.2.1 ベースライン手法	53
5.2.2 誤り行動の定義	54
5.2.3 トレーニングデータセット構築	55
5.2.4 分類器統合	56

5.3 実験	58
5.3.1 データセット	58
5.3.2 評価指標	59
5.3.3 実装詳細	59
5.4 実験結果	60
5.5 考察	61
5.5.1 失敗パターンの分析	61
5.5.2 UCB1との併用の意義	62
5.5.3 LLMの性能	63
5.6 結言	73
第6章 ナビゲータによるFTWPエージェントの性能向上	74
6.1 先行研究とその問題点	74
6.2 手法	75
6.2.1 ナビゲータ	75
6.2.2 トレーニングセットの修正	78
6.2.3 ナビゲータとエージェントの統合	80
6.3 実験	83
6.3.1 データセット	83
6.3.2 評価指標	83
6.3.3 実装詳細	83
6.4 実験結果	84
6.5 考察	85
6.5.1 テスト時にナビゲータを除去	85
6.5.2 大規模言語モデルの性能	86
6.6 結言	91
第7章 結論	93
7.1 研究の総括	93
7.2 今後の課題	95
付録A 研究業績	103
A.1 学術雑誌論文	103

A.2 国際会議論文	103
A.3 学会研究会論文	104

表 目 次

3.1	実験結果	22
3.2	アブレーション実験の結果	24
3.3	例の選択の重要性	26
4.1	自動強調付与実験結果. ▲: 有意水準 0.05 で他のすべてのモデルと比較して統計的に有意であることを示す.	40
4.2	人手実験と LLM 実験結果. 最初の文を強調: 各段落の最初の文のみを強調する手法.	45
4.3	空タイトル実験の結果. ▲: 有意水準 0.05 で統計的に有意である.	48
5.1	誤り行動分類器を適用する前後の正規化スコア (→: ベースライン → 提案手法). ▲は有意差を示す.	60
5.2	検証セットおよびテストセットにおける性能比較. ▲は CogniTextworldAgent に対する統計的有意性を示す.	61
5.3	テストセットにおける誤り行動分類器適用前後の失敗回数 (→: 適用前 → 適用後). Agent1 から 3 は初期値の異なるエージェントを表す.	61
5.4	極端な分類器を用い, 潜在的に報酬のあるすべての行動を誤分類した場合の正規化スコア (→: 行動除去 → UCB1 との併用). エージェント 1 から 3 は初期値の異なる設定である.	62
5.5	ランダムに選択した 10 本の FTWP テストゲームにおける性能比較.	64
6.1	FTWP データセットにおける性能比較.“+ Navigator”は提案手法, ▲は導入による改善が統計的に有意であることを示す.“- Navigator”はテスト時にナビゲータを外したアブレーション.	84
6.2	ランダムに選んだ 10 個のテストゲームにおける LLM と学習済みエージェントの性能.	88

図 目 次

4.1	自動強調付与手法の構造.	35
4.2	空タイトル実験の図示. 左下は通常の入力であり, 左上はタイトル情報を得ていない入力(空タイトル)である. 右はモデルの出力であり, 下線は強調する箇所である.	48
5.1	クリックブック内容と候補行動で誤り判断	54
5.2	上部はベースラインの CogniTextworldAgent アーキテクチャ, 下部は提案手法を示す.	58
6.1	3つの部屋とそれらに配置されたアイテムからなる FTWP ゲーム環境の空間構造	77
6.2	左側はナビゲータを持たないベースラインエージェント, 右側はナビゲータを有効にしたエージェントを示す. 右側のようにナビゲータが有効な場合, <i>examine cookbook</i> コマンド(ステップ 3)の実行後にナビゲーションコマンドが候補行動リストへ追加され, 以前に寝室で観察された <i>apple</i> に向けてエージェントを導くことができる.	79
6.3	我々の最高性能エージェント CogniTextworldAgent + Action Classifier + Navigator のアーキテクチャ概要	82

第1章 はじめに

テキストゲーム（インタラクティブフィクションとも呼ばれる）は、自然言語処理における挑戦的な課題のひとつである。これらのゲームでは、プレイヤーの行動およびゲームエンジンからの応答がいずれも自然言語で記述される。言語理解はテキストゲームエージェントにとって重要な課題であり、例えば料理を作るゲームにおいて「slice onion」と「chop onion」の違いを理解できなければ、タスクの失敗につながりうる。さらに、高得点を獲得するには、言語理解だけでなく、長期記憶、計画、常識推論といった複雑な能力も必要である。

しかし、テキストゲームエージェントの研究は、画像ゲーム分野に比べてはるかに少ない。その重要な理由のひとつは、適切な難易度を持つデータセットが不足していることである。テキストゲームはもともと人間プレイヤー向けに設計されており、その代表例として「Zork」が挙げられる。Zorkに関する多くの研究[8, 9, 10]は、現行の人工知能エージェントにとって依然として難易度が高すぎることを示している。したがって、困難なゲームに直接取り組むよりも、容易または中程度の難易度の環境でエージェントの性能を段階的に向上させる方が望ましい。この要求に応えるため、CôtéらはTextWorld[11]を提案した。TextWorldは、テキストゲーム生成のための標準化されたインターフェースを提供し、ユーザが手作業でゲームを構築することも、生成メカニズムを用いて自動的にゲームを生成することも可能である。

TextWorld フレームワークにより、多様なテキストゲームデータセットが構築されており、本研究ではTextWorld Commonsense(TWC)およびFirst TextWorld Problems(FTWP)を利用する。TWC[6]はIBM Researchにより構築され、エージェントの常識利用能力を評価することを目的としている。TWCのゲームでは、散乱した物品を適切な位置に戻すことが課題となる。一方、FTWPはMicrosoft Researchにより構築され、大規模なテキストゲームエージェントの評価基盤を提供することを目指している。¹ FTWPゲームでは、料理を作ることが課題で、タスクは多段階である。エージェントはまずクックブックを探し、内容を確認した上で次の行動を判断する必要がある。

¹データセット詳細: <https://competitions.codalab.org/competitions/21557>.

ある。二つのデータセットは規模が大きく異なり、TWC は 45 のゲームを含むのに対し、FTWP は 5,176 のゲームを含み、テキストゲーム分野における最大規模のデータセットとなっている。

本論文では、これら二種類の異なるデータセットに対して、それぞれ異なるエージェント実装戦略を採用する。

TWC では、大規模言語モデル (LLM) とプロンプトエンジニアリングに基づくエージェントを実装した。LLM は、大量のテキストデータに対して自己教師あり学習を行うことで訓練された言語モデルであり、近年の自然言語処理における大きな進展を代表するものである。LLM は多くのタスクで人間専門家に匹敵する性能を示す。プロンプトエンジニアリングとは、精巧に設計されたプロンプトを用いて LLM のタスク解決能力を高める技術である。本研究では、Chain-of-Thought[12], few-shot[13], 出力テンプレート [14] といった複数のプロンプト技術を採用し、性能向上に有効であることを確認した。さらに本研究では「フィードバック増強」という新たなプロンプト技術を提案し、性能を一層高めることに成功した。実験の結果、提案エージェントは TWC テストセットで (easy: 1.00, medium: 1.00, hard: 0.70) のタスク達成率を示し、従来最良エージェント (easy: 0.96, medium: 0.85, hard: 0.57) を上回った。

LLM を用いてテキストゲームエージェントを実装することには、主に二つの利点がある。第一に、学習データが少ない状況下でも、LLM の高い汎化能力によりエージェントが優れた性能を発揮できる点である。第二に、実装が比較的シンプルである点である。しかし、FTWP のようなより複雑かつ大規模なテキストゲーム環境では、LLM ベースのエージェントはコスト面および性能面で十分な成果を得にくい。こうした環境に対応するためには、LLM より小規模な言語モデルをローカルで訓練することが必要である。ローカルにおいて訓練可能な言語モデルには多様な選択肢が存在する。本研究では、テキストゲームエージェントの実装に最も適したモデルを選定するために、いくつかの代表的な言語モデルの性能を評価した。テキストゲームエージェントにとって言語理解は不可欠な能力であるため、モデルの言語理解能力を十分に反映できるタスクとして「自動強調付与」を評価基準に設定した。具体的には、日本語文章を対象とした自動強調付与タスクにおいて、fastText[3], BERT[4], RoBERTa[2], CRF[5] といった言語モデルを比較評価した。その結果、transformer ベースのモデル (BERT および RoBERTa) は他のモデルを大きく上回る性能を示し、さらに transformer モデル間においても RoBERTa が常に優れた結果を示した。この知見に基づき、我々は後続のテキストゲームエージェントの実装において一貫して RoBERTa を採用することと

した。

それゆえ、FTWPにおいて、我々はRoBERTaに基づくエージェント実装方法を採用した。CogniTextworldAgent[7]はFTWPコンテストでの優勝エージェントであり、BERTとUCB1アルゴリズム[15]で実装されている。BERTは事前学習済みのtransformerモデルの一種であり、様々な言語理解タスクで高い性能を示している。UCB1は多腕バンディット問題を解くための古典的アルゴリズムであり、行動の探索と活用のバランスを提供する。提案手法では、CogniTextworldAgentを基盤としつつ、前述の理由により再実装においてはBERTをより高性能なRoBERTaに置き換えた。

性能をさらに向上させるために、我々はCogniTextworldAgentに二つの拡張モジュールを導入した。一つは誤り行動分類器(Wrong Action Classifier)であり、クックブックの指示と矛盾する行動(例:cookbookが「slice onion」を要求している際に「chop onion with knife」を実行する)を検出し、こういう「誤り行動」のスコアを減点して選択確率を下げる。もう一つはナビゲータ(Navigator)であり、エージェントに「navigate to red onion」のようなアイテムナビゲーション行動を提供する。ナビゲータはエージェントを目的のアイテムが存在する部屋まで移動させ、マップ記録やアイテム追跡を担当することで、エージェントはハイレベルの行動決定に集中できる。

実験の結果、提案エージェント(CogniTextworldAgentに誤り行動分類器とナビゲータを導入したもの)は、FTWPテストセットで0.968のタスク達成率を示し、従来最良エージェントの0.880を大幅に上回った。

本研究は、二種類のエージェント実装戦略の性能を検証しただけでなく、テキストゲームエージェントの開発方法論も提示する。すなわち、小規模データセットに対してはLLMの強力な事前知識を活用するエージェントを実装し、その性能を検証する。LLMが性能面やコスト面で不十分な場合には、RoBERTaのような事前学習済みTransformerモデルを基盤とするエージェントを実装し、特定のゲーム環境に応じた拡張モジュールを開発して性能を向上させる。この方法論は、今後のテキストゲーム研究に新たな枠組みを提供する。

本論文の主要な貢献は以下のとおりである:

- TWCデータセットにおいて、Chain-of-Thought, one-shot, 出力テンプレート, フィードバック強化といったプロンプト技術を組み合わせたLLMベースのエージェントを提案・実装し、その優れた性能を確認した。
- 我々は「自動強調付与」タスクにより複数の小規模言語モデルの言語理解能力を

評価し，RoBERTa の優位性を確認した. その原因で，以降の研究では一貫して RoBERTa を基盤モデルとして採用した.

- FTWP データセットにおいて，先行研究の CogniTextworldAgent が誤り行動に 鈍感である課題に対し，誤り行動分類器を開発した. 誤り行動分類器により， クックブック違反によるゲーム中断を大幅に減らし，正規化スコアを 0.880 から 0.912 に向上させた.
- 同じ FTWP データセットにおいて，ナビゲータモジュールを設計・実装し，誤り行動分類器と併用することで 0.968 のタスク達成率を示し，FTWP における 性能記録を更新した.
- 方法論的観点から，テキストゲームエージェント実装における LLM と中小規模 事前学習モデルの補完関係を整理し，効率的かつ高性能なエージェント設計の方 向性を示した.

第2章 先行研究

2.1 データセットとベンチマーク

テキストゲームエージェント分野の研究は、画像ゲーム分野と比べて著しく少ない。その主要な理由の一つとして、テキストゲーム分野には適切な難易度を備えたデータセットが不足している点が挙げられる。テキストゲームは本来、人間プレイヤー向けに設計されており、その中でも最も著名な作品に「Zork」がある。Zorkに関する多数の研究[8, 9, 10]により、このゲームが依然として既存の人工知能エージェントにとって極めて困難な課題であることが示されている。こうした高難度のゲームで直接研究を行うのではなく、まずは容易から中程度の難易度をもつゲーム環境でエージェントの性能を段階的に向上させる方が望ましい。

Côtéらが開発したTextWorld[11]は、テキストゲーム研究のための生成ツールおよび標準化されたインターフェースを提供した。これにより、多様なテキストゲームデータセットが構築されており、以下では代表的なものを紹介する。

ScienceWorld[16]は、エージェントの科学的推論能力を評価する目的で設計されたテキストゲームデータセットである。小学校理科のカリキュラムを題材とし、熱力学、電気回路、化学反応、生物のライフサイクルなどをシミュレートする複数の物理プロセスエンジンを備えている。データセットは「物質の状態変化」「温度測定」「電気伝導」「摩擦」「植物と受粉」「メンデル遺伝」など10のトピックに基づく30種類のタスクで構成され、それぞれに多数のバリエーションが存在する。総タスク数は7200を超え、学習・開発・テスト用に分割されている。特にテストセットには学習・開発セットに登場しない物質が含まれるため、エージェントの汎化性能を厳しく評価できる。また、模倣学習のために各タスクには人手で作成されたウォーカスルーも提供されている。ScienceWorldの特徴は、単なる正解の提示ではなく、一連の行動を通じて答えを検証する必要がある点にあり、言語モデルや強化学習手法の科学推論能力を評価する上で重要なベンチマークとなっている。ただし、ScienceWorldは科学分野に特化しているため、同データセットで高い性能を示したとしても、エージェントが十分な汎化

能力を備えているとは限らないという課題がある。

TextWorld Commonsense[6](TWC) は、強化学習エージェントに常識知識の活用を促すことを目的としたテキストゲーム環境である。TWC データセットは TextWorld 上に構築されており、エージェントには部屋を整理し、物品を適切な場所に配置することが求められる。TWC のゲームは easy・medium・hard の 3 段階の難易度に分類され、各タスク(物品とその所属場所の対応関係)の妥当性は人手により検証されている。本研究においても、TWC は主要なデータセットの一つとして利用される。

FTWP は Microsoft Research によって構築されたテキストゲームデータセットであり、その目的はテキストゲームエージェントの評価に適した大規模なテストベッドを提供することである。¹ FTWP におけるエージェントのタスクは、クックブックに基づいて料理を作ることである。これは多段階のタスクであり、エージェントはまずクックブックを発見し、その内容を確認しなければ次の行動を決定できない(例えば、特定の食材を集めたり、特定の調理器具で処理したりする必要がある)。FTWP は 5,176 の独立したゲームを含み、訓練セット(4,440 個)、検証セット(222 個)、テストセット(514 個)の 3 分割で構成されている。これはテキストゲーム分野で最大規模のデータセットである。FTWP は本研究においても主要なデータセットの一つであり、Microsoft Research は 2019 年に同データセットを基盤とした公開コンテストを開催した。このコンテストには世界各地から 16 チームが参加し、優勝した CogniTextworldAgent は本研究における重要な基盤ともなっている。

2.2 テキストゲームエージェント

テキストゲームは、自然言語処理と強化学習の交差点に位置する挑戦的な研究分野として広く認識されている。環境からのフィードバックが完全に自然言語で提示されるため、高いスコアを獲得するには、テキストゲームエージェントは言語理解、推論、長期計画といった複数の能力を同時に備える必要がある。

初期の研究では、エージェントは主に回帰型ニューラルネットワーク(RNN)に基づく強化学習手法に依拠していた。代表的な手法として LSTM-DQN[17] があり、このモデルは隠れ状態を利用して履歴情報を保持することで、一定の長期計画能力を実現した。しかし、テキストゲームの高い複雑性により、これらの手法では行動と環境状態の間に存在する複雑な依存関係を十分に捉えることが難しかった。

¹データセット入手先: <https://competitions.codalab.org/competitions/21557>.

この課題に対処するため、深層強化学習と行動評価ネットワーク (Deep Reinforcement Relevance Network, DRRN) の統合が提案された。DRRN[18] は、環境状態と行動をそれぞれ別にエンコードし、cosine 類似度を用いて行動のスコアを算出する手法である。このアプローチは、従来の RNN ベース手法と比べて優れた汎化性能を示した。

その後、知識グラフがテキストゲームエージェントに導入され始めた。代表的な例として KG-A2C[19] があり、これは知識グラフを動的に構築し、エージェントが実体関係を活用して探索を行えるようにした。複雑な物品間の相互作用や因果推論を含むタスクにおいて、KG-A2C は従来のシーケンスマネジメントモデルを上回る探索能力とタスク達成率を示している。

TWC 環境における代表的な高性能エージェントは TWC-agent[6] と SceneIT[20] である。TWC-agent は DRRN アーキテクチャを基盤に、ConceptNet から常識知識を取得することで、常識推論が要求される TWC 環境に適応した。一方、SceneIT はテキストに加えて画像情報を補助的に導入し、マルチモーダル情報がエージェント性能の向上に寄与することを実証した。

FTWP コンテストにおいて最高性能を示したのは CogniTextworldAgent である。CogniTextworldAgent[7] は、BERT と UCB1 アルゴリズムを組み合わせ、模倣学習 (Behavioral Cloning, BC) に基づくフレームワークを実装した。BERT[4] は事前学習型 Transformer モデルの一種であり、言語理解タスクにおいて高い性能を発揮する。また、UCB1[15] は古典的な多腕バンディットアルゴリズムであり、頻繁に選択される行動にペナルティを課すことで、探索と利用のバランスを実現する。

第 2 位のエージェントである LeDeepChef[21] は、より伝統的な手法を採用している。具体的には、GloVe[22] (静的単語埋め込み) を用いて情報をエンコードし、強化学習フレームワークである Actor-Critic[23] を適用した。ただし、GloVe のような静的埋め込みには多義性 (polysemy) の問題があり、文脈に応じて単語の意味が変化する場合に対応できない。一方、BERT のような Transformer モデルは文脈依存的に単語表現を生成できるため、多義性を緩和し、柔軟かつ精緻なモデリングを可能にする。この点が、CogniTextworldAgent が LeDeepChef を大きく上回った主な要因の一つであると考えられる。

2.3 プロンプトエンジニアリングと大規模言語モデル

大規模言語モデル (Large Language Model, LLM) は、膨大なテキストデータを学習することで、人間のように自然言語を理解し生成する能力を獲得した人工知能システムである。これらのモデルは質問応答、文章生成、要約、翻訳など多様なタスクに応用可能であり、その性能は近年飛躍的に向上している。

LLM 研究における「プロンプトエンジニアリング」とは、モデルから望ましい出力を得るために入力文(プロンプト)を戦略的に設計する手法を指す。適切なプロンプト設計によって、同一のモデルであっても出力の正確性や一貫性が大きく変化することが知られている。具体的には、指示を明確かつ具体的に与える、モデルに役割を設定する、段階的な推論を促すといった工夫が有効である。このように、プロンプトエンジニアリングは LLM の実運用に不可欠な技術となりつつある。以下に、代表的かつ影響力の大きい研究成果を挙げる。

- **Few-Shot Prompting:** GPT-3 の論文 [13] で提案された手法で、プロンプト内に少数の例を提示することで、モデルにタスクの形式や意図を理解させる。ゼロショットに比べ安定した性能を示す。
- **Chain-of-Thought(CoT):** Wei ら [12] が提案した方法で、推論過程を逐次的に言語化させることにより、数学的推論や論理的推論タスクにおける精度を大幅に向上させる。
- **Self-Consistency:** Wang ら [24] による手法で、単一の推論結果に依存せず、複数の CoT 出力を生成した上で、多数決的に最も一貫性のある解答を選択することで安定性を高める。
- **Instruction Tuning:** Wei ら [25] は、大量の指示文を事前学習に組み込むことで、モデルの指示追従性を向上させる Instruction Tuning を提案した。

これらの研究は、単なるプロンプト設計の工夫にとどまらず、人間がモデルに推論や思考を促す方法論として体系化されつつある。さらに近年では、プログラムによる自動プロンプト最適化や、他の AI を用いたメタ・プロンプトエンジニアリングなど、新たな方向性も活発に模索されている。

2.4 行動消去

行動消去 (Action Elimination) は, Zahavy ら [8] によって提案された手法であり, ニューラルネットワークを用いてテキストゲーム環境における無効な行動を検出し, 行動空間から除外することを目的とする. このアプローチは「Zork」や「Grid World」などのゲームにおいて学習効率を向上させ, 複雑なテキストゲームにおける行動空間削減の有効性を実証した. その理論的・実証的基盤は, 後に Huang と Ontañón [26] の研究によってさらに補強されている.

しかし, 強制的な行動消去にはリスクが伴う. すなわち, 誤判定によってタスク達成に不可欠な行動が永久に排除されてしまうと, エージェントの成功率が著しく低下する可能性がある. そこで本研究では, 探索の安全性とモデルの安定性を両立させるため, 行動消去の枠組みに基づき「ソフトペナルティ」メカニズムを提案する. 本手法では, 疑わしい無効行動を完全に削除するのではなく, 行動選択時の優先度を下げる, あるいは負の報酬を付与することでその選択確率を抑制する. この戦略により, 誤判定による性能劣化を回避しつつ無効行動の影響を効果的に軽減でき, 理論的にはテキストゲームにおける高不確実性な探索環境への適応性が向上すると期待される.

2.5 模倣学習と因果混同

模倣学習は強化学習エージェントの訓練における有効なアプローチであり, さまざまな領域で広く利用されている. この手法は, 与えられた状態における専門家の行動を予測する識別モデルを学習することで, 強化学習を教師あり学習問題として定式化するものである. しかしながら, Haan ら [27] が指摘するように, 模倣学習には「因果混同」と呼ばれる重要な課題が存在する. すなわち, エージェントが専門家の行動を表面的に模倣するだけでは, その行動の背後にある根本的な意図を正しく捉えることができず, 結果として汎化性能を大きく損なう可能性がある.

この問題は, テキストベースゲームにおいて特に顕著に表れる. 例えば「go north」といった移動コマンドは, 必ずしも一意的な意図に対応するわけではない. それは, 寝室に入って赤いタマネギを取る場合, バスルームに向かう場合, あるいは未知の領域を探索する場合など, 複数の異なる意図に結び付けられる可能性がある. このように, 移動コマンドそのものは明確な意図を含意しないため, 因果混同が生じ, エージェントの性能低下を引き起こす恐れがある.

この課題に対処するための一つの方法として、移動コマンドのシーケンスをアイテムナビゲーションコマンド (item navigation command) へと再定式化する手法が考えられる。例えば、[”go west”, ”go south”] といった行動列は「navigate to red onion」と書き換えることができる。アイテムナビゲーションコマンドは「赤いタマネギを見つける」という意図を明示的に表現するため、暗黙的な意図推論の必要性を排除し、因果混同の影響を軽減する効果が期待できる。

2.6 ゲームとナビゲーター

ナビゲーションはコンピュータゲームにおける古典的な課題であり [28], 幅優先探索 (BFS), A*, ダイクストラ法といった確立されたアルゴリズムが、長らく NPC(Non-Player Character) の行動実装に広く用いられてきた [29]。近年のビジョンベースのゲームエージェント研究においては、しばしばナビゲーターが既に利用可能な前提となっている。例えば、Zhang らによる Minecraft の研究 [30] では、彼らの LLM ベースのエージェントは API を呼び出すだけで目的地まで自動的に移動できた。これは Mineflayer ライブラリにより提供されており、内部では A*アルゴリズムが実装されている。経路探索とアイテム追跡の双方を管理するこのナビゲーターは、Zhang らのエージェントに大きな支援を与えたものの、その寄与については明示的に分析されていない。

一方、研究例の少ないテキストゲーム領域では、最大規模のデータセットである FTWPにおいてさえ、公開されたナビゲーター実装は存在していない。仮にナビゲーターを設計できたとしても、LLM はそれを自律的に呼び出すことが可能かもしれないが、LLM より小規模なエージェントにどのように効果的に利用させるかは未解決の問題である。本論文では、まさにこの二つの課題に取り組むことを主要な貢献とする。

第3章 LLMによるTWCゲームの効率的解法

従来のテキストゲームエージェントは、主に強化学習技術に依拠して実装されてきた。しかし、強化学習には学習過程が困難であるという根本的な課題が存在する。近年の大規模言語モデル(LLM)の急速な発展[31]を踏まえ、本研究ではGPT-4[32]を活用したテキストゲーム解法を提案する。本手法の中核をなすのは、慎重に設計されたプロンプトであり、Chain-of-Thought(CoT), few-shotプロンプティング、出力テンプレートなど、プロンプトエンジニアリングにおける主要な成果を統合している。

まず、CoTは特定のプロンプト表現(例:「think step by step」)を通じてLLMに段階的推論を促し、より一貫性のある出力を導く手法である[12]。また、few-shotプロンプティング[13]は、プロンプト内に少数の例を提示することでタスク理解を補助し、従来のzero-shot設定に比べて出力の安定性を向上させる。さらに、Parkら[14]に着想を得た「出力テンプレート」は、「<fill in>」といった標準化表現を用いることでモデル出力の形式を統一し、後続の自動処理を容易にする。これらの要素を組み合わせることで、プロンプト設計の柔軟性と汎用性を高めることが可能となる。

加えて、本研究では新たに「フィードバック増強」と呼ぶプロンプト技術を導入する。これは、ゲームから得られる報酬・罰のフィードバック情報を明示化し、LLMが環境の状況をより適切に把握できるよう支援する手法である。従来研究では環境フィードバックの活用が十分に検討されてこなかったが、本手法によりエージェントの推論過程に外部情報を統合することが可能となる。

提案手法の有効性を検証するため、TWC(TextWorld Commonsense)データセットを用いて評価を行った。実験の結果、提案エージェントはTWCテストセットで(easy: 1.00, medium: 1.00, hard: 0.70)のタスク達成率を示し、従来最良エージェント(easy: 0.96, medium: 0.85, hard: 0.57)を上回った。これにより、LLMと適切に設計されたプロンプトを組み合わせることで、強化学習に依存せずとも高い性能を実現できることが明らかとなった。

3.1 TWC ゲームの紹介

TWC は IBM Research が構築したテキストゲームデータセットで、エージェントの常識の理解と活用を重視している。TWC ゲームでの人工知能エージェントのタスクは、部屋を整理することであり、具体的には特定の物体をその属する場所に戻すことである。たとえば、使ったティッシュペーパーはゴミ箱に入れるべきであり、汚れた服は洗濯機に入れるべきである。物体が特定の位置に属するかどうかの判断基準は常識であり、これらの物体-位置のペアの妥当性は人間の評価者によって確認されている。

データセットは GitHub で見つけることができる。¹ このデータセットは訓練セット、検証セット、テストセットに分割されており、各セットには、easy, medium, hard レベルのそれぞれに五つのゲームが含まれている。レベルは主に整理すべきアイテムの数と部屋の数で異なる。最も簡単なゲームは一つのアイテムと一つの部屋を含み、最も困難なゲームは七つのアイテムと二つの相互接続された部屋からなる。エージェントは「*go east*」のようなコマンドを使って部屋間を移動できる。

理解を助けるために、簡単な TWC ゲームの流れを示す。以下はゲームエンジンからの出力とプレイヤーからの入力である。

Welcome to TextWorld! You find yourself in a messy house. Many things are not in their usual location. Let's clean up this place. After you'll have done, this little house is going to be spick and span! Look for anything that is out of place and put it away in its proper location.

-= Laundry Room =- You've entered a laundry room.

An open washing machine is right there by you. The washing machine is empty! This is the worst thing that could possibly happen, ever! You can see a clothes drier. The clothes drier is empty! This is the worst thing that could possibly happen, ever! You can make out an opened laundry basket. The laundry basket is empty, what a horrible day! Oh, great. Here's a suspended shelf. On the suspended shelf you can see a wet azure dress. You make out a bench. On the bench you can make out a dirty yellow T-shirt and a white cap. You make out a work table. You shudder, but continue examining the work table. The work table is usual. The work table appears to be empty.

¹<https://github.com/IBM/commonsense-rl/tree/master/games/twc>

There is an exit to the east.

> inventory

You are carrying: a dirty yellow dress

> insert dirty yellow dress into washing machine

You put the dirty yellow dress into the washing machine.

Your score has just gone up by one point.

> take dirty yellow T-shirt from bench

You take the dirty yellow T-shirt from the bench.

> insert dirty yellow T-shirt into washing machine

You put the dirty yellow T-shirt into the washing machine.

Your score has just gone up by one point.

> take white cap from bench

You take the white cap from the bench.

> go east

You've entered a corridor.

You can see a shoe cabinet. The shoe cabinet contains a dirty checkered shirt. You can make out a key holder. The key holder is ordinary. Unfortunately, there isn't a thing on it. What, you think everything in TextWorld should have stuff on it? You see a gleam over in a corner, where you can see an umbrella stand. But the thing is empty. You can see a hat rack. Looks like someone's already been here and taken everything off it, though. You smell an intriguing smell, and follow it to a coat hanger. But there isn't a thing on it.

There is an exit to the west. Don't worry, there is no door.

There is a dirty cardigan and a dirty maroon dress on the floor.

> put white cap on hat rack

You put the white cap on the hat rack.

Your score has just gone up by one point.

...

> insert dirty checkered shirt into washing machine

You put the dirty checkered shirt into the washing machine.

Your score has just gone up by one point.

*** The End ***

このゲームはhard レベルであり、部屋は2つ、片付けるべきアイテムが7つ存在する。紙幅の都合上、プレイ過程の中間部分は省略するが、重要な行動——すなわちインベントリの確認、アイテムの取得、移動、配置——についてはすべて示している。

ゲームエンジンの初期出力(環境の説明)から、プレイヤーがランドリールームにいることが分かる。まずプレイヤーは「inventory」コマンドを実行し、汚れた黄色のドレスを所持していることを確認した。その後「insert dirty yellow dress into washing machine」コマンドを実行し、ドレスを洗濯機に入れた。これにより、ゲームエンジンから1ポイントの報酬が与えられる。

続いて、プレイヤーは長椅子から汚れた黄色のTシャツを拾い上げ、「insert dirty yellow T-shirt into washing machine」コマンドを実行して洗濯機に入れた。これによって、さらに1ポイントの報酬が与えられる。

次に、プレイヤーは長椅子から白い帽子を手に入れたが、ランドリールームには帽子を置く場所がなかった。そのため「go east」コマンドを使い、別の部屋へ移動した。ゲームエンジンの出力から、このときプレイヤーが廊下に移動したことが分かる。廊下には靴箱があり、その中には汚れた格子柄のシャツがあった。これは整理すべきアイテムの一つだが、プレイヤーはそれを後回しにし、まず白い帽子を帽子掛けに置いた。これにより、再び1ポイントの報酬が与えられる。

その後も同様に、プレイヤーは整理対象のアイテムを探し出し、適切な場所に配置していった。最終的に、すべてのアイテムを正しい場所に置いた時点でゲームは終了する。

ゲームの終了条件は2つある。1つ目はすべてのタスクを完了することであり、2つ目は実行したステップ数が事前に定められた上限を超えることである。本研究では、この上限を20ステップに設定した。MurugesanらのTWCに関する研究[6]では、ベースラインの行動回数の上限を50に設定しているが、本研究では以下の2つの理由から20に制限した。第一に、限られた条件下でGPT-4の効率性を高めることを目的としている点である。第二に、ステップ数を制限することで、API呼び出しの回数を減らし、行動履歴を簡潔に保つことができ、結果としてAPI利用コストの削減につながる点である。さらに、先行研究[6]によれば、人間が困難レベルのゲームを完了する際の平均ステップ数は15である。したがって、20ステップという制限は実用的かつ合理的であると考えられる。

エージェントの実装を容易にするために、ゲームエンジンから以下の情報を請求できる：

- **Action Feedback:** 行動が実行されると、エージェントは環境からフィードバックを受け取る。モデルが行動で報酬を得た場合、その情報もフィードバックに含まれる。例えば、行動とフィードバックのペアは次のようになる：「*insert dirty yellow T-shirt into washing machine* → *You put the dirty yellow T-shirt into the washing machine. Your score has just gone up by one point*」。この例では、「*insert dirty yellow T-shirt into washing machine*」が行動で、矢印の後の部分がゲームエンジンからのフィードバックである。
- **Environment Description:** 現在の環境のテキスト記述。
- **Inventory:** 現在保有しているアイテムのリスト。
- **Available Action List:** エージェントが現在利用可能な行動のリスト。

3.2 手法

3.2.1 プロンプト

本研究は GPT-4 に基づくテキストゲームタスクを解く手法を提案する。丁寧に設計されたプロンプトが本研究の主な貢献である。このプロンプトには、chain-of-thought(COT), few-shot prompting, output template, そして我々が提案したフィードバック増強というプロンプトテクニックが使われている。プロンプトの具体例は以下のとおりである（スペースを確保するため省略される部分がある）。

Task: You are an experienced text game player, your goal is to put things in their proper locations and improve your score.

Example walkthrough: Action 0: insert dirty yellow dress into washing machine → You put the dirty yellow dress into the washing machine. Your score has just gone up by one point. Right position. Action 1: take dirty yellow T-shirt from bench → You take the dirty yellow T-shirt from the bench. Action 2: insert dirty yellow T-shirt into washing machine → You put the dirty yellow T-shirt into the washing machine. Your score has just gone up by one point. Right position... Action 14: insert dirty maroon dress into washing machine → You put the dirty maroon dress into the washing machine. Your score has just gone up by one point. Right position.

Action history: Action 0: examine suspended shelf → The suspended shelf is reliable. Action 1: take dirty singlet → You pick up the dirty singlet from the ground.

Inventory: You are carrying: a dirty singlet

Current environment: -= Laundry Room =- You've entered a laundry room. You begin to take stock of what's here. You can see a washing machine. Empty! What kind of nightmare TextWorld is this? You make out an opened clothes drier. The clothes drier is empty! What a waste of a day! You can see a laundry basket. Hmm... what else, what else? The laundry basket is empty! What a waste of a day! You can see a suspended shelf. The suspended shelf is ordinary. But oh no! there's nothing on this piece

of garbage. You move on, clearly infuriated by TextWorld. You scan the room, seeing a bench. But the thing hasn't got anything on it...

Action you can take:

- * close clothes drier
- * drop dirty singlet
- * examine bench
- * insert dirty singlet into clothes drier
- * look
- * put dirty singlet on bench

...

Question: To put things in their proper locations and improve your score, what should you do? Think step by step then choose 'one' action from above list.

Consideration: <fill in>

Next action: <fill in>

「Task」とはタスクの説明で、固定されている文字列である。「Example walkthrough」とはLLMに提供する one-shot の例であり、基本的にはトレーニングセットにある hard レベルのゲームの一つにおける人間プレイヤーが行った行動の履歴である。部屋間の物品の移動を示すために、2つの部屋からなるゲーム環境が意図的に選ばれた。² 「Action history」とは行動履歴で、行動の回数とともに長くなり、「アクション → アクション フィードバック」の形式でテキスト化される。アクション(行動)はLLMの応答から抽出されているが、アクションフィードバックは基本的にゲーム環境から取得した内容である。ただし、フィードバック増強の原因で、物品の配置アクションに応じるフィードバックには、「Right position」などの特定なテキスト内容が追加されているので、その点に注意してほしい。フィードバック増強テクニックの詳細は3.2.2節で述べる。「Current Environment」とは現在の環境のテキストの説明である。「Action you can take」とはゲーム環境から取得した可能な行動のリストである。「Question」、「Consideration」お

²<https://github.com/IBM/commonsense-rl/blob/master/games/twc/hard/train/tw-iqa-cleanup-objects7-take6-rooms2-train-aEOOFxjEcxEI9Xo.ulx>

より「Next action」は「Task」と同様に固定された文字列であり、モデルの出力を構造化するためのテンプレートである。上記のプロンプトに対するGPT-4の出力は以下のとおりである。

Consideration: To properly place the dirty singlet, it should be put somewhere related to laundry processing. Inserting it into the washing machine seems like the most appropriate step since typically, dirty clothes need to be washed before they are dried or placed in a laundry basket.

Next action: insert dirty singlet into washing machine

思考過程を観察すると、モデルが状況に対して合理的な判断を下したことがわかる。

3.2.2 フィードバック増強

TWCゲーム環境から取得したフィードバック、特に物品の配置アクションに対するフィードバックが曖昧であり、エージェントが正確な判断を下すのを妨げる。そのフィードバックを明確にするために、我々はフィードバック増強というプロンプトテクニックを提案する。

例を挙げると、物品を誤って配置した場合、「insert clean white polo shirt into chest of drawers → You put the clean white polo shirt into the chest of drawers」のようなアクション-フィードバックペアが得られる。このフィードバックから、アクションが実行されたことが分かるが、アクションの結果、物品(ポロシャツ)がその場所(引き出し)に置くことが正しいかどうかは不明である。

一方、物品が正しく配置された場合、「insert clean white polo shirt into wardrobe → You put the clean white polo shirt into the wardrobe. Your score has just gone up by one point」のようなアクション-フィードバックペアが得られる。物品を誤って配置した場合のフィードバックと比較すると、スコアの増加が物品が正しく配置されたことを示唆していることがわかる。微調整されたモデル、あるいはfew-shot設定のLLMは、スコア増加の有無に基づいてアクションの結果を正しく判断することができるが、zero-shot設定では、このアクションフィードバックは明らかに曖昧である。

これらの要因を考慮し、我々はフィードバック増強というプロンプトテクニックを提案する。フィードバック増強のアイデアはシンプルで、物品の配置アクションに対

するフィードバックの末尾に説明文を追加し、アクションの結果を明らかにすることだけである。

具体的には、物品が正しく配置された場合、フィードバックに「**Right position**」をヒントとして追加する。例は以下のとおりである。

Insert clean white polo shirt into wardrobe → You put the clean white polo shirt into the wardrobe. Your score has just gone up by one point. **Right position.**

誤って配置する場合、「**Wrong position, you should put it somewhere else, maybe the other room**」をフィードバックに追加する。例は以下である。

Insert clean white polo shirt into chest of drawers → You put the clean white polo shirt into the chest of drawers. **Wrong position, you should put it somewhere else, maybe the other room.**

この例では、シャツは引き出しではなくワードローブに配置するべきであるため、アクションは間違っている。「**maybe the other room**」の部分はエージェントに部屋を跨いで探索することを促す。その後、正常な処理の流れに従い、増強されたアクション-フィードバックペアは行動履歴に保存され、エージェントの記憶となる。

3.2.3 APIでプロンプトを使用

本節では、APIでのプロンプトの使用方法について説明する。我々はOpenAI APIを使用してGPT-4(具体的なモデル名はgpt-4-1106-preview)およびGPT-3.5(モデル名はgpt-3.5-turbo-0613)を呼び出す。Pythonコードは次のとおりである。

```
1 from openai import OpenAI
2 client = OpenAI()
3 completion = client.chat.completions.create(
4     model='gpt-4-1106-preview',
5     messages=[
6         {"role": "system", "content": system_msg},
7         {"role": "user", "content": user_msg}
8     ]
9 )
10 )
```

ここで、*system_msg*と*user_msg*は文字列変数であり、それぞれ以下のテキスト内容を含んでいる。

- *system-msg*: タスク (Task), ゲームの流れの例 (Example walkthrough), 行動履歴 (Action history), インベントリ (Inventory), 現在の環境 (Current environment)
- *user-msg*: 可能な行動リスト (Available Action List), 出力テンプレート (Question, Consideration, Next action)

3.3 実験

3.3.1 データセット

手法の性能を評価するために我々が使用するデータセットは TWC であり, このデータセットは GitHub から入手できる.³ TWC データセットには, easy, medium, hard の 3 つの難易度があり, 難易度ごとに 5 つのゲームが含まれている. 各ゲームで, 人工知能エージェントのタスクは部屋を整理することであり, 具体的には, 一部の物品をその属すべき場所に戻すことである. たとえば, 使用済みのティッシュペーパーはゴミ箱に, 汚れた衣類は洗濯機に入れるべきである. 物品が特定の位置に属するかどうかを判断する基準は常識であり, これらの物品-場所のペアの妥当性は, 人間の評価者によって確認されている [6]. ゲームの難易度は整理する必要のある物品と部屋の数で区別される. easy レベルの環境では, 整理する必要のある物品は 1 個であり, 部屋も 1 つだけである. medium レベルでは, 物品の数は 2~3 個で, 部屋の数は 1 つである. hard レベルでは, 物品の数は 6~7 個で, 部屋の数は 1~2 つである. 2 つの部屋がある場合, 部屋がドアでつながり, エージェントは「go east」のような行動で部屋間の移動ができる.

3.3.2 実験設定

実験設定は以下のとおりである.

- **LLM**: GPT-4 を使用する. 具体的なモデル名は「gpt-4-1106-preview」である. GPT-4 を呼び出すために OpenAI API を使用し, 具体的なコードは 3.2.3 節で説明した. GPT-3.5 を使う場合の性能は 3.4.2 節のアプリケーション実験で検証する.
- **微調整**: LLM の微調整は不要である.

³<https://github.com/IBM/commonsense-rl/tree/master/games/twc>

- **行動回数上限:** ゲームごとにエージェントが取ることができる行動の上限を 20 と規定する.
- **重複実験:** 重複実験をしない. すべてのスコアは 1 回の実行結果である.
- **評価指標:** 評価指標は TWC 論文 [6] と同様に正規化スコア (normalized score) を主とし, 「エージェントが達成したスコア」 ÷ 「最大可能なスコア」で計算する. 各タスクの達成に 1 ポイントが付与されるため, 正規化スコアはエージェントのタスク達成率を意味する. その他, 平均ステップ数も示す. 平均ステップ数とはエージェントがゲームごとに取ったアクション (行動) の数の平均であり, 少ないほどエージェントの効率が良いと考えられる.
- **プロンプトに含まれるサンプルの数:** サンプルは一つである (one-shot). サンプルなし (zero-shot) の場合の性能は 3.4.2 節のアブレーション実験で検証する.

3.3.3 ベースライン

本研究で使用されるベースラインには, LSTM-A2C, DRRN, KG-A2C, および TWC Agent(常識知識は Numberbatch で導入)が含まれており, これらのベースラインの特徴について 2.2 節で紹介した. ベースラインのスコアは TWC の論文から直接取得し, 再実装およびテストを行っていないことに注意してほしい.

3.4 実験結果

3.4.1 従来手法との比較

実験結果は表 3.1 に示されている. 各難易度レベルには 5 つのゲームが含まれている. 「Score」とは normalized scores であり, ゲームで達成したスコアを最大達成可能スコアで割ることによって計算される. 1.0 の normalized score はエージェントがすべてのタスクを遂行したことを意味する. 「Step」とはエージェントがゲームごとに取ったアクションの数の平均であり, 少ないほどエージェントの効率が良いと考えられる. 表の示すとおり, すべての難易度レベルにおける提案手法の性能が一番良いであることが分かる. なお, すべてのベースライン手法はトレーニングセットで 100 エピソードのトレーニングを行ったが, 提案手法には LLM の微調整が不要である.

	Easy		Medium		Hard	
	Score	Steps	Score	Steps	Score	Steps
LSTM-A2C	0.86	17.6	0.74	38.0	0.54	49.2
DRRN	0.81	18.9	0.73	33.4	0.44	46.2
KG-A2C	0.85	17.7	0.72	37.2	0.46	49.4
TWC-Agent	0.96	11.8	0.85	27.1	0.57	44.2
Our approach	1.00	2.2	1.00	3.6	0.70	18.8

表 3.1: 実験結果

3.4.2 アブレーション実験と結果

異なるプロンプトテクニックが提案手法に与える影響を検証するために、プロンプトテクニックを提案手法から一つずつ除去し、その性能をテストする。

“-GPT4”: 提案手法は GPT-4[32] に基づいており、GPT-4 を他の大規模言語モデルに置き換えた場合の性能変化を検証したい。本実験では、GPT-4 を GPT-3.5 に置き換える、この手法を “-GPT4” と名付ける。出力テンプレート (*Question*, *Consideration*, および *Next Action*) の存在により、GPT-4 の応答は構造化されており、アクションを応答テキストから自動的に抽出することができる。しかし、GPT-3.5 の応答はほとんどテンプレートに従わず、有効なアクションを抽出するためには追加のグラウンディング技術が必要となる。本研究では、GPT-3.5 の応答テキストからアクションを人手で抽出する。人間アシスタントのタスクは、モデルの応答からテキストの一部(上記の例では「*Next action*」の後の部分)をコピーし、それをゲームのインターフェースに貼り付けることである。

“-COT”: 提案手法にある出力テンプレートは以下のとおりである。

Question: To put things in their proper locations and improve your score, what should you do? Think step by step then choose 'one' action from above list.

Consideration: <fill in>

Next action: <fill in>

「Think step by step」と「Consideration: <fill in>」などの提示が Chain-of-Thought (COT)[12] を参考にして作ったものである。プロンプトからこれらの提示を削除する

と性能が低下するかどうかを検証したい。それゆえ、出力テンプレートを以下のように変更する。

Question: To put things in their proper locations and improve your score, what should you do? Choose 'one' action from above list.

Next action: <fill in>

このプロンプトに基づく手法を “-COT” と名付ける。

“-one-shot”: Brown ら [13] は、プロンプトの中に例を提示することで、LLM の性能が向上することを発見した。我々の提案手法は one-shot 設定に基づいている。本実験では、one-shot 設定の有効性をテストするために、プロンプトから「Example walkthrough」の部分を削除し、このプロンプトに基づく手法を “-one-shot” と名付る。この設定は zero-shot とも考えられる。

“-FA”: フィードバック増強由来の性能向上を示すために、フィードバック増強の原因で追加された説明文を削除し、この手法を “-FA” (-Feedback Augmentation) と名付ける。例えば、以下のように物品が誤った場所に置かれた場合の増強されたアクションフィードバックペアがある。“-FA” の手法では、太字で示された説明文が削除される。

Insert clean white polo shirt into chest of drawers → You put the clean white polo shirt into the chest of drawers. **Wrong position, you should put it somewhere else, maybe the other room.**

実験結果を表 3.2 に示す。まず、GPT-4 が提案手法にとって最も重要な部分であることが明らかである。GPT-4 を GPT-3.5 に置き換えた後、hard レベルのゲームでの normalized score が 0.70 から 0.18 に低下した。性能の他、GPT-3.5 の多くの出力はテンプレートに従わず、手入れが必要である。次に、フィードバック増強が提案手法にとって重要なテクニックであることが分かる。フィードバック増強を除去すると、hard レベルのゲームでの normalized score が 0.70 から 0.52 に低下した。最後に、Chain-of-Thought と one-shot 設定もモデルの性能向上に役立つことが分かる。

	Easy		Medium		Hard	
	Score	Steps	Score	Steps	Score	Steps
TWC-Agent	0.96	11.8	0.85	27.1	0.57	44.2
Our approach	1.00	2.2	1.00	3.6	0.70	18.8
-GPT4	1.00	7.2	0.58	15.2	0.18	20.0
-COT	1.00	2.2	1.00	3.8	0.61	18.8
-one-shot	1.00	2.2	1.00	3.8	0.64	18.8
-FA	1.00	2.2	1.00	3.8	0.52	18.8

表 3.2: アブレーション実験の結果

3.5 考察

3.5.1 新しく作ったゲーム環境における GPT-4 の性能

GPT-4 はインターネットデータでトレーニングされており、TWC データセットは GitHub で公開されている。そのため、GPT-4 がトレーニング中に TWC のテストデータセットをすでに見たことがあるかどうかが不明である。GPT-4 のトレーニングデータセットに TWC のテストデータセットが含まれていないことを証明できないが、我々は新しく作った環境で GPT-4 の性能をテストすることができる。我々は TextWorld[11] を使ってゲーム環境を実装する。このゲーム環境には、2 つの部屋 (kitchen と bedroom) と以下の「物品-場所」ペアが含まれている。

used noodle cup → garbage can

chili oil → kitchen pantry

computer monitor → writing desk

a bag of cookies → kitchen pantry

a family photo → bedroom cabinet

game software → writing desk

apple core → garbage can

すべての物品と場所が TWC データセット (train, dev, および test) に存在しないことを確認した。環境を初期化する時、物品をランダムに配置し、エージェントを寝室に置く。エージェントの行動回数上限を 20 に設定する。実験の結果、提案手法は family

photo 以外の物品をすべて正しく戻し, 正規化スコアは $6 / 7 = 0.86$ であった. このゲーム環境は本研究で作ったものであり, 肝心な知識(物品-場所ペア)はインターネット上に存在しない. このゲーム環境での GPT-4 の性能は, GPT-4 が常識に基づくテキストゲームタスクを解決する能力を持っていることを示唆している.

3.5.2 より広範な応用シナリオ

本研究は, LLM を用いてテキストゲームを解くことに注目している. 我々は TWC データセットだけで実験を行ったが, 提案手法は他のテキストゲームにも容易に適用できる. 新しいゲーム環境に適用するために, 最初のステップはタスク説明を変更することである. 例えば, Zork という有名なテキストゲームで手法を使う場合, タスク説明を「あなたはトレジャーハンターで, タスクは宝物を集めてスコアを上げることである」に変更することが考えられる. タスク説明を変更した後, フィードバック増強戦略も変更しないと行けない. アクションとフィードバックは環境ごとに異なるため, 新しい環境には新しいフィードバック増強戦略を設計する必要がある. 例えば, エージェントが不合理な行動を示したときに適切な説明文を追加することが考えられる. フィードバック増強戦略は環境に応じて再設計する必要があるが, フィードバック増強のアイデア自体はシンプルで効果的であることが, 我々の実験結果で示されている.

タスク説明とフィードバック増強戦略を変更した後, エージェントは新しい環境で行動することができる. しかし, テキストゲームの中で, TWC のゲーム流れは比較的短く, hard レベルの環境でも 20 ステップで全てのタスクを完了させることができる. ゲームの流れは長くなると, 行動履歴が長くなり, LLM の処理効率が低下し, コストも増加する. この問題に対処するために, 大規模言語モデルの要約機能を通じて行動履歴を圧縮することが考えられるが, この方向性についての探求は今後の研究テーマとする.

3.5.3 例の選択の重要性

One-shot プロンプティングの主な問題の一つは, 例の選択がモデルの性能を大きく影響することである. 3.2.1 節で述べたように, 部屋間の物品の移動を示すために, 2 つの部屋を含むゲーム環境が意図的に選ばれた. 例の選択がモデルの性能に与える影響を調査するために, 本節ではその例をより簡単な例に置き換え, モデルの性能の変

	Easy		Medium		Hard	
	Score	Steps	Score	Steps	Score	Steps
TWC-Agent	0.96	11.8	0.85	27.1	0.57	44.2
Our approach	1.00	2.2	1.00	3.6	0.70	18.8
-one-shot	1.00	2.2	1.00	3.8	0.64	18.8
Easy one-shot	1.00	2.4	1.00	3.6	0.64	18.8

表 3.3: 例の選択の重要性

化を観察する。この新しい例は、人間のプレイヤーが easy レベルのゲーム環境で行った行動の履歴である。具体的な内容は以下のとおりである。

Example walkthrough: Action 0: take dirty gray underpants from work table → You take the dirty gray underpants from the work table. Action 1: insert dirty gray underpants into washing machine → You put the dirty gray underpants into the washing machine. Your score has just gone up by one point. Right position.

表 3.3 の示すとおり、簡単な例を使用した後、hard レベルの環境におけるエージェントの性能 (Easy one-shot) は 0.70 から 0.64 に低下し、zero-shot 設定 (-one-shot) の性能とほぼ同等である。例の選択がモデルの性能に大きく影響することが分かる。

3.5.4 ケーススタディ

Easy や medium レベルのゲームでは、提案手法は常に最少のステップで全てのタスクを完了する。しかし、hard レベルのゲームでは、エージェントが**不適切な行動**を行うことがある。これらの行動は主に、特定の物品が現在の部屋に属していないことに起因する。例えば、以下のような hard レベルのゲームからの行動履歴がある。

Action 0: take wet white jumper from patio chair → You take the wet white jumper from the patio chair.

Action 1: put wet white jumper on clothesline → You put the wet white jumper on the clothesline. Your score has just gone up by one point. Right position.

Action 2: **put milk on patio table** → You put the milk on the patio

table. Wrong position, you should put it somewhere else, maybe the other room.

太字で示されている部分が、エージェントが行った不適切な行動である。このゲーム環境では、牛乳は中庭のテーブルに置かれるべきではない。この行動を取った際のエージェントの考慮は以下のとおりである。

Consideration: The milk you are carrying is likely intended to be placed somewhere it can remain cool or be utilized. It is not appropriate for the BBQ, clothesline, or the patio chair. Among the options, the patio table seems to be the most logical place to put the milk, as it is a typical location to place food and beverages when you're spending time outdoors.

Next action: **put milk on patio table**

エージェントはバーベキュー、物干し、中庭の椅子が不適切な場所であると正しく判断しているが、別の部屋に冷蔵庫があることを認識していない。そのため、牛乳を中庭のテーブルに置くことにした。ゲーム環境からのフィードバックを受けた後のエージェント考慮は以下のとおりである。

Consideration: The milk is out of place; it should likely be put somewhere with refrigeration to prevent spoilage, such as a refrigerator, which could be inside the house.

Next action: take milk from patio table

エージェントは合理的な推論を行ったことが分かる。次の行動履歴の示すように、エージェントは後続のアクションで牛乳を正しい場所に戻すことができた。

Action 2: **put milk on patio table** → You put the milk on the patio table. Wrong position, you should put it somewhere else, maybe the other room.

Action 3: **take milk from patio table** → You take the milk from the patio table.

Action 4: **go west** → -= Kitchen =-You find yourself in a kitchen.

Action 5: **insert milk into fridge** → You put the milk into the fridge.

Your score has just gone up by one point. Right position.

しかし、何度か失敗した後に、モデルが他のアイテムに注意を移す場合もある。例えば次のケース(hard レベルの 2 番目のゲーム)では、モデルは 2 回試みた後に白いポロシャツを諦めている:

Action 6: **put clean white polo shirt on sofa** → You put the clean white polo shirt on the sofa. Wrong position, you should put it somewhere else, maybe the other room.

Action 7: take clean white polo shirt from sofa → You take the clean white polo shirt from the sofa.

Action 8: go west → -= Bedroom =-You find yourself in a bedroom.

Action 9: **insert clean white polo shirt into chest of drawers** → You put the clean white polo shirt into the chest of drawers. Wrong position, you should put it somewhere else, maybe the other room.

Action 10: go east → -= Livingroom =-You are in a livingroom. Action 11: take brown suit from end table → You take the brown suit from the end table.

さらに、稀ではあるが、GPT-4 は同じ誤った行動を 2 回繰り返すミスをすることもある。以下の例がその一つ:

Action 8: go south → -= Corridor =-Well, here we are in a corridor.

Action 9: take clean white socks → You pick up the clean white socks from the ground.

Action 10: **insert clean white socks into shoe cabinet** → You put the clean white socks into the shoe cabinet. Wrong position, you should put it somewhere else, maybe the other room.

Action 11: take clean white socks from shoe cabinet → You take the clean white socks from the shoe cabinet.

Action 12: **insert clean white socks into shoe cabinet** → You put the clean white socks into the shoe cabinet. Wrong position, you should put it somewhere else, maybe the other room.

次に、アクション 12(靴箱に靴下を繰り返し入れる)のときにモデルがどう考えていたかを見てみよう:

Consideration: In the current scenario, you are in a corridor with various items and furniture pieces such as a shoe cabinet, hat rack, key holder, and umbrella stand, all of which are empty. You're carrying a pair of clean white socks. The socks typically belong in a bedroom or a designated storage area for clothing, but considering that the shoe cabinet is mentioned to be empty and is a suitable location for storing such items, placing clean socks there seems reasonable. Although it was previously stated that the shoe cabinet was not the right place for the clean white socks, given the context within the game, it may be that different types of items have specific correct placements; the previous location could have been incorrect for the other item but might be correct for the clean white socks. Additionally, the placement of items is often based on logical organization within a living space.

Next action: **insert clean white socks into shoe cabinet**

このときモデルは、ゲーム環境が返すフィードバックの正確さを疑っているように見える。部屋の中で靴箱以外に靴下を置く適切な場所がないので、この疑いは理解できる。実際には靴下は別の部屋に置くべきであり、これはアイテムを部屋をまたいで移動させる難しさを改めて示している。

誤った行動以外にも、モデルの性能に悪影響を与える可能性のある行動のカテゴリとして、冗長な行動がある。例えば次のケース(同じくハードレベルの2回目のゲーム)では:

Action 0: insert clean plaid polo shirt into wardrobe → You put the clean plaid polo shirt into the wardrobe. Your score has just gone up by one point. Right position.

Action 1: **close wardrobe** → You close the wardrobe.

この例では、アクション 1(クローゼットを閉じる)が冗長な行動にあたる。クローゼットを閉じると、その後に他の服を拾っても「クローゼットに入れる」という選択肢が出なくなる。TWC のゲーム環境では、何かを閉じても(スコアが上がるなどの)メ

リットはなく、完全に無駄な行動となる。「閉じる必要はない」といったヒントを加えればモデルの性能はさらに上がるかもしれないが、それをするとデータセット固有の知識を導入してしまい、我々の手法の一般性を損なうことになる。

3.6 結言

本研究は GPT-4 でテキストゲームを解く手法を提案した。強化学習に基づく従来手法と比較して、提案手法の実装が簡単で、優れた性能を発揮する。TWC データセットでの実験の示すとおり、提案手法は学習済みの強化学習モデルを上回る性能を有する。具体的には、提案手法は easy と medium レベルのゲーム環境で全てのタスクを遂行し、hard レベルの環境においても、提案手法の性能 (0.70) が最良のベースライン TWC-Agent(0.57) を大きく上回った。今後、我々はより複雑なテキストゲーム環境での研究を進めたい。

第4章 より複雑なゲームへ:RoBERTaの性能検証

LLM(大規模言語モデル、数千万から数十億多数のパラメータを持つ人工ニューラルネットワークで構成されるコンピュータ言語モデル)を用いてテキストゲームエージェントを実装することには、主に二つの利点がある。第一に、学習データが少ない状況下でも、LLMの高い汎化能力によりエージェントが優れた性能を発揮できる点である。第二に、実装が比較的シンプルである点である。しかし、より複雑かつ大規模なテキストゲーム環境においては、LLMベースのエージェントのAPI使用コストを無視できず、そのような環境で十分な性能を確保することも難しい。こうした環境に対応するためには、ローカルモデルをファインチューニングすることが必要である。では、どのようなローカルモデルを訓練対象として選ぶべきだろうか。テキストゲームを解くためには高度な言語理解能力が求められることから、本章では、モデルの言語理解能力を十分に評価可能なタスクである「自動強調付与」において、複数のローカルモデルの性能を比較・評価する。

実験の結果、fastText[3]、BERT[33]、RoBERTa[2]、さらに従来手法であるCRFを比較したところ、RoBERTaが最も高い言語理解能力を示すことが確認された。この結果は、今後のテキストゲームエージェント研究において、RoBERTaを基盤モデルとして採用する合理的な理由となる。

4.1 タスク紹介: 日本語文章の自動強調付与

文章の強調表示は、ハイライトや太字などの様々な形式を含み、読者が文書をより良く理解するのに大いに役立つ。機械が強調を自動生成できれば、読者の理解を助けるだけでなく、簡潔で理解しやすい文書の作成に役立つ。それに対して、自動強調付与の目的は、文中の重要な部分を機械に強調させることである。自動強調付与の例を挙げる(lifehackerの記事):

入力(文):

バケーションは長ければ長いほど良い、といつつい思いがちですが、「Time」がオススメしているのは、できるだけ長い休暇をとることではなく、思い出に残る瞬間が常に続くような短い休暇をとること。

出力(強調付与された文):

バケーションは長ければ長いほど良い、といつつい思いがちですが、「Time」がオススメしているのは、できるだけ長い休暇をとることではなく、
思い出に残る瞬間が常に続くような短い休暇をとること。

実験のために我々はライフスタイルサイト「Lifehacker Japan」の2012年3月の記事331件を収集した。原文中の太字強調を正解ラベルとして使い、強調はトークン単位で付与される。データセットでは、記事ごとの平均文数は21文、平均トークン(形態素解析器 MeCab を使用)数は541、平均強調されるトークン数は98である。

4.2 関連研究

日本語文章の自動強調付与に類似したタスクとして、「視覚メディア内のテキストの強調選択」というタスクがあり、このタスクでは文章ではなくポスター上のテキストが強調される[34]。しかし、テキストのスタイル、テキストの長さ、タイトル情報の有無などの方面からこのタスクは文章の自動強調付与と大分異なっている。したがって、我々の研究と最も関連性のある研究は、村田らの研究[35]である。村田ら[35]は、条件付きランダムフィールド(CRF)を使用し、単語、品詞、TF-IDF、段落情報、タイトル情報など、慎重に選択された素性を組み合わせて自動強調付与を行った。村田らの手法の主要な課題は素性の品質であり、近年、ニューラルネットワークの急速的な発展を踏まえ、我々はtransformer モデルに基づいて手法の再設計を提案する。タスクの特性に基づいて、我々が最終的に設計したモデルは、3つの部分から構成される：事前学習済みのマスク言語モデルRoBERTa、タイトル情報入力、およびCRF層である。次に、節を分けて関連研究を紹介する。

4.2.1 RoBERTa

事前学習済みマスク言語モデルRoBERTaは、我々の研究手法の重要な部分である。マスク言語モデルとは自然言語処理分野における重要な進歩であり、その複雑な歴史

的発展を踏まえ、我々は3つの側面から RoBERTa に関する背景研究を紹介する。

- RoBERTa は、動的な単語埋め込みの手法の一種である。近年、単語埋め込みの手法は、静的な単語埋め込み方法から動的な単語埋め込み方法への移行を経てきた。より早期の静的な単語埋め込み手法、例えば Word2Vec[36] や GloVe[22] は、各単語に対して固定的な特徴ベクトルを出力する。それゆえ、複雑な意味や文脈情報を捉えるのが難しい。一方、動的な単語埋め込み手法では、単語のベクトルは常にコンテキストから計算され、多くの自然言語タスクでより高い性能を発揮している [37]。
- RoBERTa は Transformer モジュールに基づいている。Transformer の登場は大規模言語モデルの発展を大いに促進した。その進化した性能と高いトレーニング効率により、Transformer は長・短期記憶 (LSTM)[38] やゲート付き回帰型ユニット (GRU)[39] を取って代わり、大規模ニューラルネットワークの基本構成要素となっている。
- RoBERTa は BERT[33] をベースに改良されたアーキテクチャである。BERT の変種として、RoBERTa の改良には以下の内容が含まれる:(1) より多くのデータ、より大きなバッチサイズ、より長いトレーニング時間でモデルを予備訓練すること；(2) 次の文予測目標の削除；(3) より長いシーケンスでの訓練；(4) トレーニングデータに適用されるマスキングパターンの動的な変更。RoBERTa は様々なタスクで BERT より優れた性能を持つことが報告されている [2]。

4.2.2 タイトル情報

タイトル情報は、様々な自然言語処理タスクのパフォーマンス向上に役立つことが示されている。例えば、Narayan ら [40] は、タイトルを含む補足情報を用いることで、抽出型要約タスクのパフォーマンスが向上することを検証した。日本語文章の自動強調付与の分野では、村田ら [35] はタイトル情報を除去すると手法の性能が大幅に低下することを報告した。タイトル情報は重要であるが、マスク言語モデルがそれをどのように利用すべきかについて、一般的に受け入れられた解決策はない。本研究の貢献の1つは、事前学習済みマスク言語モデル上にタイトル情報を取り入れるための、シンプルで効果的なアプローチを提案することである。

4.2.3 CRF

CRF[5] は条件付き確率場の略称で、無向グラフにより表現される確率的グラフィカルモデルの一つであり、識別モデルである。文脈を考慮できる能力により、CRF は、シーケンスラベリング問題に特に適している。深層学習が一般的に使われる前には、CRF は多くの研究で人手で選んだ素性と組み合わせて用いられていた [41, 35]。その後、再帰型ニューラルネットワーク (RNN) の上にくっつける RNN + CRF というアーキテクチャが一般的になった [42, 43]。近年、事前学習済みマスク言語モデルの上に CRF を追加する手法が、以前の手法と比較してより高い性能を達成できることが示されている [44, 45]。このような歴史に踏まえて、本研究は日本語文章の自動強調付与タスクに事前学習済みマスク言語モデル + CRF のアーキテクチャを採用する。

4.3 手法

本章では、transformer モデルで自動強調付与を解く提案手法を説明する。提案手法は、事前学習済みマスク言語モデルの RoBERTa, タイトル情報, CRF レイヤーの 3 つの構成要素に分解できる。本節では、モデルアーキテクチャの詳細な説明を提供する。

モデルアーキテクチャは、図 4.1 の示すように、RoBERTa モデルとその上に置かれたトークンレベルの分類器、そして線形連鎖 CRF で構成される。その上に、RoBERTa がタイトル情報にアクセスできるように、我々は入力テキストにいくつかの変更を加える。

入力は 2 つの部分に分割できる：(1) 強調すべき文、(2) 文章のタイトル。それぞれの対応するトークンは図 4.1 で「*Text Tok_x*」、「*Title Tok_x*」と表される。言語モデルはトークンを入力として受け取るため、文を複数のトークンに分割する必要がある。そこで一部のトークンは特別な目的を持っており、例えば [CLS] とは classification(分類) の略であり、[CLS] トークンはいつもトークンシーケンスの先頭に置かれ、対応する埋め込みはよく文レベルの分類タスク(感情分析など)に使用される。[SEP] とは separation(分割) の略であり、入力内の異なる部分を分割するために一般的に使用され、提案手法では入力文とタイトルの分割に使用される。特殊トークンの説明を踏まえ、タイトル情報を次のように導入することができる：

$$[\text{CLS}] < \text{TEXT} > [\text{SEP}] < \text{TITLE} >$$

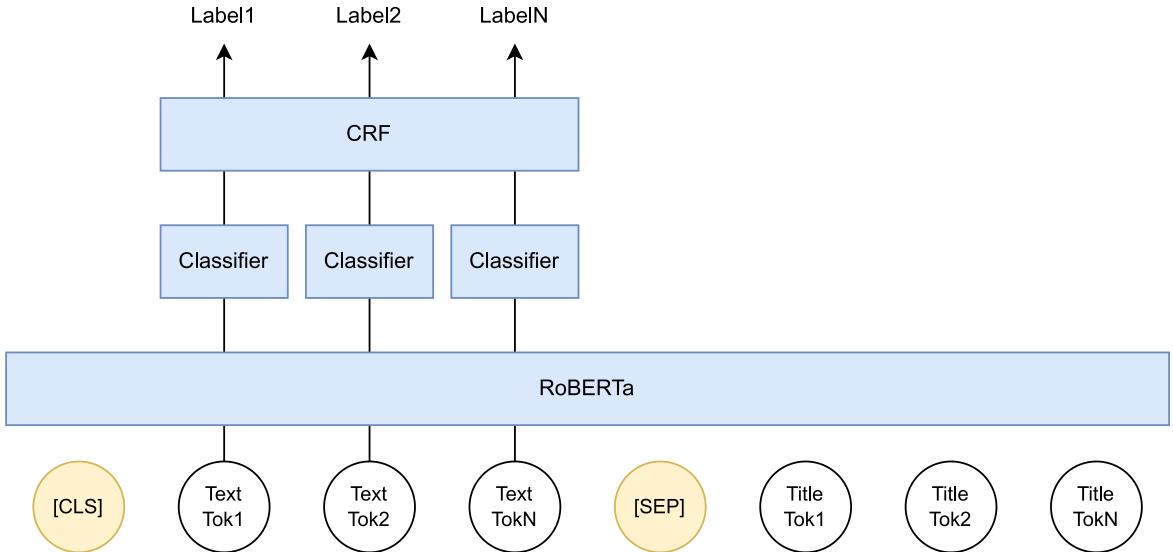


図 4.1: 自動強調付与手法の構造.

最後に、図 4.1 から、モデルの構成要素 (RoBERTa, 分類器, CRF) は異なる長さを持つことが観察できる。これは、モジュールが入力から統合される情報量を反映している。RoBERTa はすべてのトークンの情報を統合できる。分類器は 1 回につき一つのトークン情報しかアクセスできない。CRF は分類器からの出力情報をすべて統合できる。次に、アーキテクチャの各部分について詳細に説明する。

4.3.1 RoBERTa におけるタイトル情報の導入

タイトル情報を考慮することで、さまざまな自然言語処理タスクの性能が向上することが示されている。例えば、Narayan ら [40] は、補助情報(タイトルに限らない)を利用することで、抽出的要約タスクの性能を向上できることを示した。日本語文書の自動強調分野では、村田ら [35] がタイトル情報を削除すると手法の性能が著しく低下することを報告している。タイトル情報の重要性は認識されているが、事前学習済み transformer モデルがタイトル情報をどのように活用すべきかについての普遍的な解決策は存在しない。本研究の貢献のひとつは、transformer モデル上でタイトル情報を組み込むための、シンプルかつ効果的なアプローチを提案することである。

Transformer モデルは入力としてトークンのみを扱うことができるため、文章は処理される前に複数のトークンに分割される必要がある。Transformer モデルは認識可能なすべてのトークンからなる語彙を持ち、特別な役割を持ついくつかのトークンは「特殊トークン」と呼ばれる。RoBERTa モデルには以下の特殊トークンが含まれている。

[CLS]: 「分類 (classification)」の略である。学習時には、[CLS] トークンはトークン列の先頭に置かれる。実際の応用では、[CLS] トークンの対応する埋め込みは文章レベルの分類タスク（例：感情分析）によく使用される。

[SEP]: 「区切り (separation)」の略である。[SEP] トークンは、入力内の異なる部分を区切るためによく使用される。本研究では、[SEP] トークンを文章とタイトルを区切るために使用する。

[PAD]: 「パディング (padding)」の略である。バッチ学習の際、すべての入力は同じ長さである必要があり、[PAD] トークンは空いている位置を埋めるために使用される。

以上の説明に基づき、タイトル情報は以下のように導入される。

$$[CLS] < TEXT > [SEP] < TITLE >$$

ここで、<TEXT>は入力文に対応するトークンを、<TITLE>はタイトルに対応するトークンを表す。

4.3.2 RoBERTa-CRF

提案モデルは、RoBERTa の上にトークンレベルの分類器を置き、その後に線形連鎖CRF をつなげた構成になっている。入力が n トークンの場合、RoBERTa の出力は $n \times H$ で、ここで H は隠れ層のサイズを表す。RoBERTa-base では $H = 768$ になる。その後、分類器が出力をラベル空間にマッピングする、つまり $\mathbb{R}^H \mapsto \mathbb{R}^K$ となる。ここで K はラベルの数を表す。本研究ではラベルは強調ラベルと非強調ラベルの 2 種類しかないと想定し、 $K = 2$ となる。分類器の出力 $\mathbf{P} \in \mathbb{R}^{n \times K}$ はそのまま CRF 層に入力される。CRF のパラメータは、ラベル遷移行列 $\mathbf{A} \in \mathbb{R}^{K+2 \times K+2}$ で構成されている。行列 \mathbf{A} の要素 $A_{i,j}$ は、ラベル i からラベル j への遷移スコアを表す。なお、 \mathbf{A} にはシーケンスの開始タグと終了タグを表す 2 つの追加状態も含まれている。

次に、入力シーケンス $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$ と予測ラベルシーケンス $\mathbf{y} = (y_1, \dots, y_n)$, $y_i \in \{1, \dots, K\}$ に対して、シーケンスのスコアは次のように計算される。

$$s(\mathbf{X}, \mathbf{y}) = \sum_{i=0}^n A_{y_i, y_{i+1}} + \sum_{i=1}^n P_{i, y_i},$$

ここで y_0 と y_{n+1} はそれぞれ開始タグと終了タグを表す。提案モデルは、正しいラベルの対数尤度を最大化するように学習される。

$$\log(p(\mathbf{y}|\mathbf{X})) = s(\mathbf{X}, \mathbf{y}) - \log\left(\sum_{\tilde{\mathbf{y}} \in \mathbf{Y}_x} e^{s(\mathbf{X}, \tilde{\mathbf{y}})}\right) \quad (4.1)$$

ここで \mathbf{Y}_x は全ての可能なラベルシーケンスを表す。評価時には、Viterbi デコーディングを使って最も尤度の高いシーケンスを求める。以前の研究 [4, 44] と同様に、単語が複数のサブトークンに分割される場合は、最初のサブトークンに基づいて予測と損失計算を行う。

4.3.3 日本語データセット向けの事前トークン化処理

日本語の単語分割は英語より複雑である。BERT や RoBERTa などの異なるトークナイザーは、文を異なる単語列に分割する。トークンレベルの自動強調タスクでは、同じ強調でもスコアが変わることがあり、統一された評価基準を作るのが難しい。

この問題に対処するために、まず日本語単語分割ツール MeCab を使って、Lifehacker のウェブ記事を単語ごとに分解した。その後、元の強調に基づいて各単語にラベルを付け、統一されたデータセットを作成した。訓練やテストの過程では、異なるトランスフォーマーモデルがそれぞれのトークナイザーを使って、各単語を1つ以上のサブトークンに分割した。各単語に対応する最初のサブトークンが強調の必要性を決定する。こうして、異なる日本語トランスフォーマーモデル間で統一された評価基準を確立した。

4.4 実験

本節では、データセット、評価指標、モデルの実装、ベースラインと人手実験の設定について説明する。

4.4.1 データセット

村田ら [35] に従って、実験のために我々はライフスタイルサイト「Lifehacker Japan」の2012年3月の記事331件を収集した。提案手法の入力単位は文であり、原文中の太字強調を正解ラベルとして使う。データセットでは、記事ごとの平均文数は21文、平均トークン(形態素解析器 MeCab を使用)数は541、平均強調されるトークン数は98である。

実験では、モデルの性能を評価するために5分割交差検証を用いる。各分割では、訓練セットの末尾から500文をデブデータセットとして取り出す。人手実験では、実験量を抑えるために、データセットの最初の10記事をテストセットとして使用する。

4.4.2 指標

本研究は自動強調付与タスクをシーケンスラベリングタスクとして定式化したため、シーケンスラベリングの一般的な指標であるF1スコアを評価に用いる。F1スコアとは、適合率と再現率の調和平均である。ニューラルネットワークのランダム性による性能の変動を減らすために、すべてのスコアは、3回の実験の平均である。

4.4.3 モデルの実装

本研究で使用したディープラーニングフレームワークはPyTorchとHuggingFaceである。使用したRoBERTaモデルは、リンナ社がHuggingfaceリポジトリにアップロードした「japanese-roberta-base」の事前学習モデルである。BERT-baseモデルと同様に、RoBERTa-baseは12層の隠れ層、768次元のベクトル、12のアテンションヘッドで構成される。CRFレイヤーはPythonパッケージ「torch-crf」を用いて実装した。

4.4.4 ベースライン

本研究で使用した4つのベースラインモデルは以下のとおりである：

CRF: 村田ら[35]のモデルを表し、彼らは条件付きランダムフィールド(CRF)を使用し、単語、品詞、TF-IDF、段落情報、タイトル情報など、慎重に選択された素性を組み合わせて自動強調付与を行った。モデルのスコアは彼らの論文から直接参照した。

BiLSTM: 双方向LSTM(BiLSTM)とfastTextの組み合わせを表す。fastTextとはFacebook AI Researchによって開発された静的単語埋め込み手法である。日本語文章の自動強調付与タスクにおける静的単語埋め込みの性能を示すために、このモデルを実装する。具体的なモデルは、fastTextの公式サイトから訓練済みの300次元の日本語モデルである。そのため、BiLSTMの隠れ層サイズも300に設定する。

BERT: BERTの元の論文[4]は、シーケンスラベリングの問題に対するシンプルな解決策を提供した。我々はそれに基づいて実装を行った。BiLSTM+fastTextと比較することで、静的な単語埋め込みモデルと動的な単語埋め込みモデルの性能上の差を

観察できる。実装では、東北大学がHuggingface リポジトリにアップロードした「bert-based-japanese-whole-word-masking」の事前学習済みモデルをファインチューニングした。

RoBERTa: BERT とほぼ同じ方法で実装されたが、事前学習済みモデルはモデルがリンナ社がHuggingface リポジトリにアップロードした「japanese-roberta-base」である。

4.4.5 パラメータ設定

学習率: Transformer モデルを使うすべての手法の学習率は、Devlin らの研究 [4] で推奨されたハイパープラメータである $2e-5$ に設定する。BiLSTM の場合、パラメータ数は BERT や RoBERTa に比べてはるかに少ないため、 $1e-4$ というより大きな学習率を設定する。

訓練ステップ: Transformer モデルを使うすべての手法の訓練ステップは 3,000 に設定する。BiLSTM は収束が難しいため、訓練ステップ数を 30,000 に設定する。すべてのモデルでは、300 ステップごとにチェックポイントを保存し、評価時には開発データセットで最も高い性能を示したチェックポイントを選択し、対応するテストスコアを報告する。

バッチサイズ: すべてのモデルのバッチサイズは 16 に設定する。

オプティマイザ: AdamW.

重複回数: 3 回。ニューラルネットワークのランダム性の影響を緩和するために、すべての実験(人手実験を除く)を 3 回実行しスコアの平均を取る。

4.4.6 アブレーション実験

提案モデルは RoBERTa, タイトル情報, CRF の三つの構成要素に分解できるが、モデルにとって最も重要な部分は何か、すべての構成要素が不可欠なのかという疑問が生じる。この疑問に答えるために、構成要素を一つずつ取り除き、モデルのアブレーション実験を行った。アブレーション実験のために、以下の 3 つのアーキテクチャを設計した:

-CRF: CRF レイヤーを除去した提案モデルを表し、RoBERTa + Title と等しい。

-タイトル: タイトル情報の入力を除去した提案モデルを表し、RoBERTa + CRF と等しい。

-RoBERTa: 研究手法で事前学習済みマスク言語モデルRoBERTaをBERTに置き換えたものであり, BERT + Title + CRFと等しい。BERTの事前学習モデルは, 東北大学がHuggingfaceリポジトリにアップロードした「bert-based-japanese-whole-word-masking」である。

上記のすべてのモデルのパラメータは4.4.5節で説明したとおりである。

4.5 実験結果

提案手法, ベースライン及びアブレーション手法の性能を含める重複実験の結果を表4.1に示す。提案手法は平均F1スコア0.437を達成し, 最も高性能だったベースライン(スコア0.399)より0.038の改善を達成し, すべての手法の中で最高の性能を示した。

モデル	適合率	再現率	F1
ベースライン			
CRF	0.314	0.312	0.313
BiLSTM	0.277	0.324	0.297
BERT	0.376	0.372	0.362
RoBERTa	0.372	0.444	0.399
提案手法とアブレーション			
RoBERTa + Title + CRF	0.398	0.509	0.437[▲]
- CRF	0.414	0.453	0.423
- Title	0.360	0.476	0.401
- RoBERTa	0.387	0.459	0.409

表4.1: 自動強調付与実験結果. ▲: 有意水準0.05で他のすべてのモデルと比較して統計的に有意であることを示す。

アブレーション手法の性能から, タイトル情報が提案手法にとって最も重要な構成要素であることが分かる。タイトル情報を除去すると, 提案手法の性能が大幅に低下し, F1スコアが0.028低下した。最後に, 有意水準0.05で両側ウィルコクソンの符号付き順位検定を行い, 結果が統計的に有意であることを確認できた。

4.6 考察

4.6.1 人手実験とLLM実験

提案モデルの性能上の優位性を直感的に示すために、部分データセット（全データセットの最初の10ドキュメント）で人手とモデルの性能を比較する。近年、大規模言語モデル（LLM）への関心が自然言語処理分野で広まっていることを受けて、同じデータセットでChatGPTとGPT-4の性能も示す（ゼロショット）。

人手実験の場合、3人の被験者はすべて日本人母語話者である。被験者には、ランダムに選択した3つのサンプル文章が次のように提示される（強調された情報が含まれている）。

次の10記事を読んで、強調したい箇所を太字にしてください。宜しくお願ひします。

- 最初の一行はタイトルで、《》で囲みます。
- ■機能・特徴 のようなサブタイトルを強調しないでください。
- 返信は強調する箇所が分かるword、あるいはpdfファイルでお願いします。

サンプル3つ：

《Googleのページを開いていなくてもGoogleトークの使える拡張機能「Chat for Google」》

Chrome:Googleのチャット機能は、サクッとメッセージを送ったり、ビデオチャットをするのに便利ですが、GmailのようなGoogle系のページを開いている時にしか利用できません。そこでGoogleは、ブラウザと切り離された、どんなWebページを見ても関係なくGoogleチャットを見るとのできる公式拡張機能「Chat for Google」をリリースしました。

この拡張機能は少し前からあったのですが、『Chrome OS』のみ対応でした。現在では、他のプラットフォームのChromeでも使用可能です。デスク

トップからチャットを行ったり、通話したり、「Google Hangouts」を利用することもできます(これが筆者のお気に入り機能だとか)。「Google Voice」の通話機能に対応していないのが残念ですが、それでもやはり便利です。わざわざGmailのタブを切り替えるのにうんざりしていた人は、特にそう感じるでしょう。

Chat for Googleは、すべてのGoogle Chrome対応の無料拡張機能です。もちろん、デスクトップのGoogleトーク以上のものがほしい場合は、複数のプロトコルに対応しているインスタントメッセンジャークライアントをインストールしてもいいと思います。

《臭いが気になったら「手の殺菌ジェル」をデオドラントに使うハック》

汗だくになる夏は、それなりに心がけてケアするのですが、意外と冬は自分の臭いに鈍感になります....そこで、携帯用のデオドラントを持ち合わせていないとき、お助けグッズとして使えるのが、手の殺菌ジェルです。

女性向けブログメディア「Shine」によると、**手の殺菌ジェルを直接腋の下につける**と、デオドラント効果が得られるそうです。肌につけた後、しばらくは強いアルコール臭を感じるかもしれません、数分でカラダの臭いとともに消えるとか。一日帰宅するまで、快適にすごせますよ。

この記事では、そのメカニズムについては詳しく述べられていませんが、おそらく臭いの元となる細菌をアルコールが滅し、これによって臭いの作用が弱まるのだろうと考えられます。

風邪やインフルエンザの予防に殺菌ジェルを持ち歩いている方は、手の殺菌だけでなく、デオドラントにも使えることを覚えておくといいかもしれません。

《余ったブックエンドで「ティッシュボックス立て」をDIY》

自分の仕事場にティッシュを置いている人は多いと思いますが、ティッシュボックスは意外とスペースを取ります。できればボックスをタテに置き、スペースを取らないようにしたいものです。しかし、わざわざ専用の立て掛けグッズを買うのは...という人のために、余ったブックエンドでティッシュ

ボックスを縦置きにする方法をご紹介しましょう。

今回紹介する「ティッシュボックス立て」に必要なグッズは、ブックエンド 2つだけ。2つのブックエンドを以下のように組み合わせればOKです。

あとは、その2つのブックエンドの間にティッシュボックスを挟み、最後にブックエンドの片方を養生テープなどで固定すれば完成です。ブックエンドの一方だけを固定してブックエンドの幅を自由に変えられるようすれば、どんな大きさのティッシュボックスにも対応できます。

なお、ティッシュ1枚ごとを強く引っ張るとティッシュボックス自体が倒れてしまいます。なので、手前側のブックエンドの「底」の部分を、もう一方のブックエンドの下に「差し込む」ように挟みましょう。こうすれば、ボックスが倒れなくなりますよ。

オフィスや書斎にブックエンドが余っている方は、試してみてはいかがでしょうか？

(安齋慎平)

====

強調はここからでお願いします。

(1) 《シンプルでフリーながらも多機能な Mac 用テキスト拡張ツール『xType』》

...

LLM の実験では、入力長の制限から、次のプロンプトを用いたゼロショット設定を採用した：

Please refer to the title and put boldface emphasis on the text you think is important.

Title: 《シンプルでフリーながらも多機能な Mac 用テキスト拡張ツール『xType』》

Text: MacOSX: 無料のテキスト拡張ツールをお探しの方には『xType』がオススメです。フリーソフトなのにも関わらず、有償ツールに負けず劣ら

ずの機能性を持っています。テキスト拡張ツールのトップとは決して言えませんが、かなり優秀です。

xTypeは、他のテキスト拡張アプリと同様に機能します。使用したいショートカットを入力し、拡張したいテキストを入力、そしてショートカットをテスト、という手順で使用可能です。基本的な機能性に加えてフォーマット情報の保存、画像の追加、日付や時間などのシンプルな変数への対応などもできます。クリップボード機能や入力ダイアログなどの機能がこれに追加されれば、有償アプリを含めてもかなりの強敵となるでしょう。現時点では無料ツールの中では最も優秀、といったところです。

今のところ英語とフランス語のみ対応で、残念ながら日本語は対応言語に含まれていませんが、気になった方は試してみてください！xType(無料)—MacAppStoreviaOneThingWell AdamDachis(原文／訳: まいるす・ゑびす)

タイトルと内容は日本語であるが、プロンプトは英語であることに注意してほしい。この戦略を採用したのは、Kuzman らの研究 [46] によると、マイナー言語で ChatGPT を使用する場合、英語のプロンプトを使用することが望ましいためである。この他にはプロンプトを一切調整していない。他の手法では、訓練済みモデルを 1 つランダムに選択し性能評価を行う。

人手実験と LLM 実験の結果を表 4.2 に示す。参考までに、文章の最初の文を強調するルールベースの手法のスコアも示しており、これを「最初の文を強調」と略する。提案手法は他のすべての手法を上回る性能を示した。

強調例

本節においては、人手強調、LLM による強調、そして我々が提案する手法による出力例を示す。まず、正しい強調は以下のとおりである。

MacOSX: 無料のテキスト拡張ツールをお探しの方には『xType』がオススメです。フリーツールなのにも関わらず、有償ツールに負けず劣らずの機能性を持っています。テキスト拡張ツールのトップとは決して言えませんが、かなり優秀です。

	適合率	再現率	F1
ルールベースの強調			
最初の文を強調	0.175	0.179	0.211
人手実験			
被験者 1	0.306	0.272	0.288
被験者 2	0.249	0.186	0.213
被験者 3	0.337	0.323	0.331
人手平均	0.297	0.260	0.276
大規模言語モデル			
ChatGPT	0.300	0.218	0.253
GPT4	0.289	0.308	0.298
ファインチューニングモデル			
CRF	0.242	0.318	0.275
BiLSTM	0.208	0.360	0.264
BERT	0.268	0.285	0.276
RoBERTa	0.286	0.383	0.327
RoBERTa + Title + CRF	0.292	0.549	0.382

表 4.2: 人手実験と LLM 実験結果. 最初の文を強調: 各段落の最初の文のみを強調する手法.

xType は、他のテキスト拡張アプリと同様に機能します。使用したいショートカットを入力し、拡張したいテキストを入力、そしてショートカットをテスト、という手順で使用可能です。基本的な機能性に加えてフォーマット情報の保存、画像の追加、日付や時間などのシンプルな変数への対応などもできます。クリップボード機能や入力ダイアログなどの機能がこれに追加されれば、有償アプリを含めてもかなりの強敵となるでしょう。現時点では無料ツールの中では最も優秀、といったところです。

今のところ英語とフランス語のみ対応で、残念ながら日本語は対応言語に含まれていませんが、気になった方は試してみてください！xType(無料)—MacAppStoreviaOneThingWell AdamDachis(原文／訳: まいるす・ゑびす)

次に、人手強調(3名の被験者の中からランダムに抽出したもの)の例を以下に示す。

MacOSX: 無料のテキスト拡張ツールをお探しの方には『xType』がオススメです。フリーソフトなのにも関わらず、有償ツールに負けず劣らずの機

能性を持っています。テキスト拡張ツールのトップとは決して言えませんが、かなり優秀です。

xTypeは、他のテキスト拡張アプリと同様に機能します。使用したいショートカットを入力し、拡張したいテキストを入力、そしてショートカットをテスト、という手順で使用可能です。基本的な機能性に加えてフォーマット情報の保存、画像の追加、日付や時間などのシンプルな変数への対応などもできます。クリップボード機能や入力ダイアログなどの機能がこれに追加されれば、有償アプリを含めてもかなりの強敵となるでしょう。現時点では無料ツールの中では最も優秀、といったところです。

今のところ英語とフランス語のみ対応で、残念ながら日本語は対応言語に含まれていませんが、気になった方は試してみてください！xType(無料)—MacAppStoreviaOneThingWell AdamDachis(原文／訳: まいるす・ゑびす)

その後、LLM(GPT-4)による出力例を以下に示す。

MacOSX: 無料のテキスト拡張ツールをお探しの方には『xType』がオススメです。フリーソフトなのにも関わらず、有償ツールに負けず劣らずの機能性を持っています。テキスト拡張ツールのトップとは決して言えませんが、かなり優秀です。

xTypeは、他のテキスト拡張アプリと同様に機能します。使用したいショートカットを入力し、拡張したいテキストを入力、そしてショートカットをテスト、という手順で使用可能です。基本的な機能性に加えてフォーマット情報の保存、画像の追加、日付や時間などのシンプルな変数への対応などもできます。クリップボード機能や入力ダイアログなどの機能がこれに追加されれば、有償アプリを含めてもかなりの強敵となるでしょう。現時点では無料ツールの中では最も優秀、といったところです。

今のところ英語とフランス語のみ対応で、残念ながら日本語は対応言語に含まれていませんが、気になった方は試してみてください！xType(無料)—MacAppStoreviaOneThingWell AdamDachis(原文／訳: まいるす・ゑびす)

最後に、提案手法による出力例を以下に示す。

MacOSX: 無料のテキスト拡張ツールをお探しの方には『xType』がオススメです。フリーツールなのにも関わらず、**有償ツールに負けず劣らずの機能性を持っています**。テキスト拡張ツールのトップとは決して言えませんが、かなり優秀です。

xTypeは、他のテキスト拡張アプリと同様に機能します。使用したいショートカットを入力し、拡張したいテキストを入力、そしてショートカットをテスト、という手順で使用可能です。基本的な機能性に加えて**フォーマット情報の保存、画像の追加、日付や時間などのシンプルな変数への対応などもできます**。クリップボード機能や入力ダイアログなどの機能がこれに追加されれば、有償アプリを含めてもかなりの強敵となるでしょう。現時点では無料ツールの中では最も優秀、といったところです。

今のところ英語とフランス語のみ対応で、残念ながら日本語は対応言語に含まれていませんが、気になった方は試してみてください！xType(無料)—MacAppStoreviaOneThingWell AdamDachis(原文／訳: まいるするびす)

4.6.2 空タイトル実験

アブレーション実験では、我々の手法においてタイトル情報が最も重要な部分であることが分かる。本節では、タイトル情報の有効性を更に検証し、タイトルがモデルの振る舞いにどのように影響するかを探るために実験を設計する。提案モデルのRoBERTa + Title + CRFに対して、推論時にタイトルを空文字列に置換し、モデルの出力を比較する。モデルを再び訓練するのではなく、同じモデルを使用することに注意してほしい。違いは入力にタイトル情報の有無だけであり、ニューラルネットワークのランダム性の影響を最小限に抑えることができる。

2つの手法は図4.2に示されており、左下は通常の入力であり、左上はタイトル情報を得ていない入力(空タイトル)である。右にはモデルの出力があり、下線は強調する箇所である。

図4.2が示すように、多くの例で、空タイトルの場合、強調箇所が増えることが分かる。より厳密なデータを取得するために、我々は空タイトルを入力とするモデルの性能をテストデータセットで評価した。実験結果は表4.3に示される。「タイトルあり」というのは提案モデルRoBERTa + Title + CRFのことで、「空タイトル」とはタイト

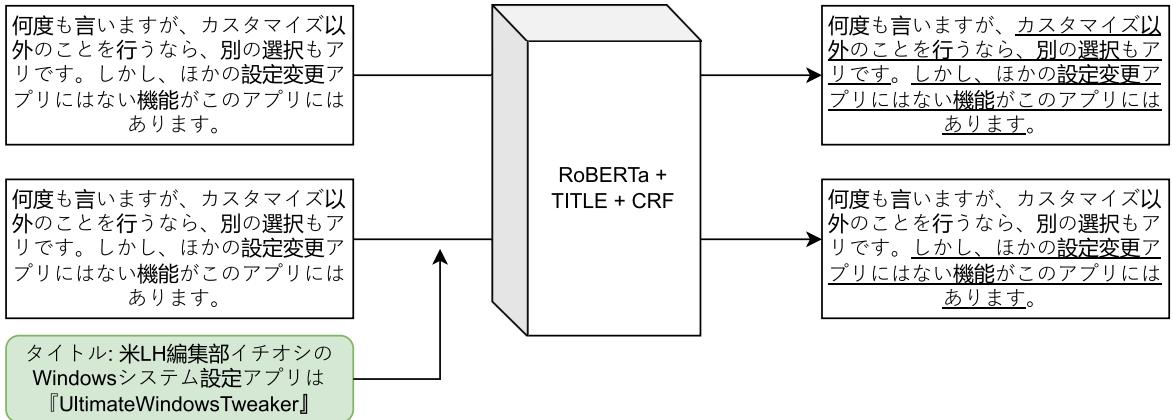


図 4.2: 空タイトル実験の図示. 左下は通常の入力であり, 左上はタイトル情報を得ていない入力(空タイトル)である. 右はモデルの出力であり, 下線は強調する箇所である.

入力	適合率	再現率	F1	強調率
タイトルあり	0.395▲	0.505	0.434▲	22.0%
空タイトル	0.356	0.508	0.409	26.3%▲

表 4.3: 空タイトル実験の結果. ▲: 有意水準 0.05 で統計的に有意である.

ルを空文字列に置き換えたモデルである. ニューラルネットワークのランダム性の影響により, 表 4.3 の提案モデルのスコアと表 4.1 の間にわずかな違いがある.

表 4.3 より, 「タイトルあり」と「空タイトル」の 2 つのモデルの再現率がほぼ同じであることが分かる(有意差なし). しかし, 空タイトルのモデルの精度が 0.039 低下したことから, 空タイトルのモデルがより多くの強調を行っていることが示唆される. 文章ごとの強調率(トークン単位)の平均を調べ, その結果を表 4.3 の 5 列目に示す. 強調率が 22.0% から 26.3% に上昇することより, 空タイトルのモデルがより多くの強調を行うことが分かる.

この現象に対する我々の考えは以下である. 提案モデルは文単位で強調を行い, 文章全体の内容を把握することができなく, すべての文を同等に重要視している. しかし, 文章には重要でない文もたくさんあるためこれは我々の直感に反する. このとき, 文章全体の概要としてのタイトル情報の補助で, モデルはある程度まで文の重要性を判断できる. 重要な文に多くの強調を加え, それほど重要でない文を無視することで, 不必要な強調を減らすことができる.

4.7 結言

我々は日本語文章の自動強調付与を研究し、簡単で効果的な手法を提案した。事前学習済みマスク言語モデル RoBERTa の上に、タイトル情報と条件付きランダムフィールド (CRF) を組み合わせることにより、提案手法は従来の研究や最良のベースラインと比較して大幅に性能が向上した。実験では、提案手法は平均 F1 スコア 0.437 を達成し、最も高性能だったベースライン (スコア 0.399) より 0.038、CRF を用いた従来研究 (スコア 0.313) より 0.124 性能が向上した。有意水準 0.05 で両側ウィルコクソンの符号付き順位検定の結果は、提案手法がすべての比較手法との差が統計的に有意であることを示した。

実験結果から、fastText, BERT, RoBERTa, さらに従来手法である CRF の間で比較すると、RoBERTa がより強力な言語理解能力を示した。この結果は、今後のテキストゲームエージェント研究において RoBERTa を基盤モデルとして用いる理由である。

第5章 誤り行動分類器でFTWP エージェントの性能向上

First TextWorld Problem(FTWP)は、テキストゲーム分野における最大規模のデータセットである。¹ FTWPは料理タスクを中心とした5,176個のゲームで構成されており、エージェントは各ゲームのクックブックに従って料理を行う必要がある。2019年には、MicrosoftによってFTWPを基盤としたコンテストが開催され、優勝したCogniTextworldAgent[7]は当時の最先端性能を達成した。CogniTextworldAgentは、BERT[4]ベースのスコアリングモデルとUpper Confidence Bound 1(UCB1)アルゴリズム[15]を統合し、トレーニングセットに含まれる専門家のウォータースルーを利用した模倣学習によって行動方策(policy)を習得している。ウォータースルーとは、順番とおりに実行することで必ずタスクを完了できる行動列である。

我々は検証セットでの観察を通じて、CogniTextworldAgentにおける失敗パターンの一つを明らかにした。すなわち、エージェントがクックブックの指示に反する「誤り行動」を取ることで、ゲームが早期終了する事例が頻発する点である。この発見は、リスクの高い行動を適切に認識・回避する能力に関して、CogniTextworldAgentが制約を抱えていることを示している。

この制約に対処するため、我々は誤り行動分類器の導入を提案する。具体的には、ヒューリスティックなラベリング戦略を用いて食材処理行動のデータセットを構築し、RoBERTaベースの分類器を訓練して誤り行動を識別させる。その後、UCB1アルゴリズムを適用し、誤りと判定された行動の選択確率を低減させる。

本研究の中心概念——補助分類器(誤り行動分類器)を用いて誤り行動を識別し、それに罰則を与える手法——は、無効行動を除去して行動空間(action space)を縮小することを目的とする「行動消去(Action Elimination)」[47]の研究分野と密接に関連している。行動消去の顕著な制約は、いったん誤りと分類された行動が、以降エージェントにとって永久に利用不可能となる点にある。そのため、誤分類が生じた場合、エー

¹データセット入手先: <https://competitions.codalab.org/competitions/21557>.

ジェントの性能に深刻な悪影響を及ぼす可能性がある。

これに対し我々の手法では、誤分類の影響が最小限に抑えられる。なぜなら、本手法では誤りと分類された行動のスコアを低下させるものの、その行動自体は依然として選択可能だからである。この仕組みは、行動分類器と UCB1 アルゴリズムを組み合わせることで実現している。

実験結果として、我々の手法は FTWP テストセットにおいてエージェントの平均正規化スコアを 0.880 から 0.912 へと向上させることを確認した。この改善は、両側 Wilcoxon 符号順位検定によって統計的に有意であることも示された ($p < 0.05$)。

5.1 FTWP ゲームの紹介

FTWP はテキストゲーム分野において最大規模のデータセットである。FTWP ゲームにおいてエージェントは、以下の手順でタスクを遂行する必要がある。

1. クックブックを発見し、その内容を確認する；
2. クックブックに記載された特定の食材を収集する；
3. 指示に従い、適切な調理器具を用いて食材を処理する；
4. 調理済みの料理を食べる。

FTWP ゲームにおけるクックブックは、エージェントが達成すべき目標を明示するタスクマニュアルとして機能する。以下にクックブックの一例を示す。

Ingredients:

red onion

Directions:

roast the red onion

prepare meal

Ingredients セクションではエージェントが収集すべき食材が列挙され、各食材を初めて入手した時点で 1 ポイントが与えられる。一方、*Directions* セクションでは収集した食材の処理方法が指定される。処理には *kitchenware* が必要であり、そのリストにはナイフ、オーブン、コンロ、BBQ が含まれる。エージェントが食材を正しく処理する

と、さらに1ポイントを獲得できる。すべての食材の収集と処理を終えた後に *prepare meal* と *eat meal* を実行すると、それぞれ追加で1ポイントが付与され、ゲームは終了となる。

以上から明らかなように、各ゲームで得られる最大スコアはクックブックの内容に依存する。参考までに、学習用データセットにおける1ゲームあたりの平均最大スコアは4.48であり、最小値は2、最大値は11である。

FTWP訓練集の各ゲームにはウォームスルーが付属している。ウォームスルーとは、順番とおりに実行することで必ずタスクを完了できる行動列である。以下に例を示す。

```
["inventory", "go north", "go east",
 "examine cookbook", "go west",
 "go south", "take red onion",
 "go north", "go east",
 "cook red onion with oven",
 "prepare meal", "eat meal"]
```

ゲーム開始時、エージェントはベッドルームにおり、初期観測(ゲームの始まりの環境記述)によって赤玉ねぎが同じ部屋に存在することが確認できる。まずエージェントは *inventory* コマンドを実行し(“*inventory*”は所持品を確認する操作)，その結果、何も持っていないことがわかる。次に移動コマンドを用いて環境を探索する。北に移動し、続いて東に進むとキッチンに到達し、そこにはクックブックとオーブンがある。エージェントが *examine cookbook* を実行すると、赤玉ねぎを収集し(*Ingredients*の部分に参照)，それを焼く必要がある(*Directions*の部分に参照)ことが明らかになる。そのため、エージェントはベッドルームに戻り赤玉ねぎを取得して1ポイントを得る。再びキッチンに移動し、オーブンで焼くことでさらに1ポイントを獲得する。クックブックの要件を満たした後、エージェントは *prepare meal* および *eat meal* を実行し、それぞれ追加で1ポイントを得る。この時点では全てのタスクが完了し、ゲームは終了する。

5.2 手法

本研究の中心的な発想は、クックブックの指示に違反する行動(以下「誤り行動」と呼ぶ)を識別するために、追加のニューラルネットワーク分類器を学習させることである。ある行動が誤りと分類された場合、その選択確率を低減する。本セクションでは、

まず FTWP ゲームの概要を紹介し、「誤り行動」を定義する。続いて、誤り行動分類器を学習するためのデータセット構築方法について述べる。最後に、誤り行動分類器を CogniTextworldAgent に統合する手順を詳しく解説する。

5.2.1 ベースライン手法

本研究における主要なベースラインは、FTWP データセットにおいて最先端の性能を示した CogniTextworldAgent である。このエージェントは BERT と UCB1 アルゴリズムで構成されているが、我々の再実装では、BERT をより高性能な RoBERTa に置き換えている。CogniTextworldAgent が行動を選択する具体的な流れは以下のとおりである。

- RoBERTa は現在の観察に基づき、各候補行動に対してスコアを算出する。
- UCB1 は行動の実行頻度に応じてスコアを調整し、頻繁に選択された行動にはペナルティを与える。最終的に、調整後のスコアが最も高い行動が選択される。

RoBERTa への入力は次の形式で構成される：

```
[CLS] <inventory count> <inventory description>  
<cookbook> <environment description> [SEP] <action>
```

ここで、[CLS] と [SEP] は BERT における特別なトークンであり、前者はシーケンスの開始を示し、後者は入力の異なる部分を区切る役割を持つ。今回の再実装では、BERT を RoBERTa[2] に置き換え、それに対応するトークンを利用している。*inventory count* は現在持っているアイテムの数を表し、エージェントがアイテムを持ちすぎているかどうかを判断する際に役立つ[7]。*inventory description* はエージェントが実際に持っているアイテムの内容を示す。*cookbook* は料理本の内容である。*environment description* には部屋の名称とその詳細な環境説明が含まれるが、環境説明が長すぎる場合には、BERT(または RoBERTa) の入力制限に合わせて切り詰められる。最後に、*action* はスコア付け対象となる候補コマンドに対応し、テスト時には最もスコアが高いコマンドがエージェントの出力として選択される。

5.2.2 誤り行動の定義

FTWP では、各ゲームにウォーカーが付属している。ウォーカーとは、すべてのタスクを完了するための一連の理想的な行動である。ウォーカー内の行動を「正しい行動」と見なした場合、それ以外には以下の 2 種類の行動が存在する：

1. **無駄行動**: ウォーカーに現れるが、クックブックの指示には違反せず、実行しても不可逆な影響を及ぼさない行動（例：重要なアイテムを一時的に捨てる行動。捨てた後に再び拾えれば問題ない）。
2. **誤り行動**: クックブックの指示に違反する行動で、実行するとゲームオーバーとなる可能性がある行動（例：「chop the red onion」が求められているのに、「slice red onion with knife」を実行する。Slice した赤玉ねぎを再び chop することができないからだ）。

検証用データセットにおける行動分析の結果、CogniTextworldAgent の失敗の大部分が誤り行動の実行に起因していることが示唆された。これらの誤り行動は、正しい行動と高い類似性を持つため（例：「slice red onion with knife」と「chop red onion with knife」），区別が困難である。この問題に対処するため、クックブック違反行動を検出する分類器のトレーニングを提案する。分類器によって検出された誤り行動には、UCB1 アルゴリズムを用いてペナルティが付与される。図 5.1 に示すように、誤り行動は行動をクックブック内容と比較することで識別可能であるため、分類器の入力としてはクックブック内容と候補行動のみが必要となる。

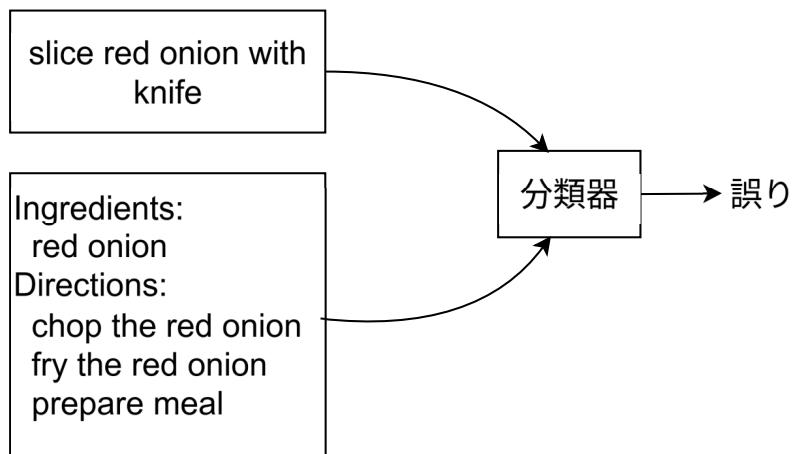


図 5.1: クックブック内容と候補行動で誤り判断

5.2.3 トレーニングデータセット構築

本研究で構築しようとするデータセットの各エントリは、以下のように構造化される：

- 入力 (x)： クックブック内容 + 食材処理行動
- ラベル (y)： 1(誤り行動) または 0(誤りではない行動)

入力 (x) のクックブック内容は、FTWP ゲームで `examine cookbook` コマンドを実行することで得られる内容を指す。クックブックは Ingredients と Directions の 2 つの部分で構成され、エージェントが達成すべき目標を記述している。FTWP では、食材処理行動は `cook` と `cut` の 2 種類に分けられる。`cook` 行動は常に `cook <ingredient> with <kitchenware>` の形式に従い、`cut` 行動は常に `<verb> <ingredient> with knife` の形式に従う。そのため、正規表現を用いて候補行動リストから 2 種類の行動を抽出可能である。クックブックおよび候補行動リストはゲームエンジンから容易に取得できる。

ラベル (y) の割り当てには、以下のヒューリスティックルールを適用した：

- ウォークスルーに登場する食材処理行動は 0(誤りではない行動) とラベル付けする。
- その他の食材処理行動は 1(誤り行動) とラベル付けする。

このラベリング戦略は、以下の観察に基づく：

- Walkthrough には誤り行動が含まれない。なぜなら、誤り行動が実行されるとゲームオーバーとなるためである。
- クックブックに記載された正しい食材処理行動は必ずウォークスルーに含まれる。ウォークスルーは全タスクを達成するための行動列であるためである。
- クックブックに記載のない食材処理行動は、ゲームオーバーを引き起こさない場合もある（誤り行動ではなく無駄行動）が、我々はそれを誤り行動 (1) としてラベル付けした。無駄行動を減らすことはエージェントの性能に悪影響を与えないと仮定している。

この方法により、訓練および検証用データセットを構築した。訓練データは 35,767 件で、誤り行動と正しい行動の比率は 4.77:1 である。検証データは 1,710 件で、誤り行動と正しい行動の比率は 4.52:1 である。

行動分類器には RoBERTa を使用した。RoBERTa は BERT の最適化版で、トレーニングプロセスを改善し、多様なタスクで性能向上が確認されている。分類器の入力は以下の形式でエンコードされる：

```
[CLS] <cookbook> [SEP] <action>
```

シーケンス分類タスクに従い、[CLS] トークンの埋め込みが最終分類に使用される。トレーニングは AdamW オプティマイザを用いて最大 3 エポック実施し、学習率は 1×10^{-5} 、バッチサイズは 32 とした。検証セットでの最高性能モデルは以下のとおりである：

- 誤り行動検出精度:90.4%(1267/1400)
- 誤りではない行動検出精度:100.0%(310/310)

誤り行動分類器と CogniTextworldAgent の入力を比較すると、クックブック内容および候補行動が含まれるため、後者は行動の正確性を判断するのに十分な情報を持っていることがわかる。それでも、CogniTextworldAgent はクックブックに違反する行動を避けることに頻繁に失敗する。これは、エージェント訓練の主要目的が誤り行動の識別ではないためである。対照的に、本手法では、訓練の目的を特化した誤り行動分類器を導入することで、より信頼性の高い誤り行動の識別を可能にする。

5.2.4 分類器統合

誤り行動の選択傾向を抑制するためには、学習済み分類器をエージェントの行動選択プロセスに統合する必要がある。直接的なアプローチとしては、誤りと予測された行動を完全に除外する方法があり、これは行動消去 (AE) フレームワークに対応する。しかし、行動消去は分類ミスの影響を受けやすく、実行可能な行動を永久に除去してしまう可能性がある。

このリスクに対処するため、我々は UCB1 のスコアリングメカニズムを用いたより柔軟な統合手法を提案する。具体的には、行動が誤りと分類された場合、その実行回数を人為的に 1 増加させる。UCB1 アルゴリズムでは、実行回数が多い行動ほどスコアが低くなるため、この調整により該当行動のスコアが下がり、選択確率が減少する。

実行回数を 2 や 3 ではなく 1 だけ増加させる理由は以下のとおりである。UCB1 アルゴリズムでは、実行回数が少ない行動が優先される。したがって、実行回数に n を追

加することは、他のすべての行動が n 回実行されるまで当該行動の考慮を遅延させることと同義である。テキストゲームにおける膨大な行動空間と、エージェントの最大行動数が 100 に制限されている状況では、 $n > 1$ は過度な制約となり、行動が実行される前に上限に達する可能性がある。一方で、 $0 < n < 1$ の場合、RoBERTa の出力スコアを考慮せずに他のすべての後に行動を即座に実行するようエージェントに指示することと等価である。これは検討可能な選択肢の一つではあるが、本研究においてはより直観的な $n = 1$ を採用した。

このように UCB1 アルゴリズムと統合することで、分類器がエージェントの行動戦略を完全に覆すことなく、行動選択に影響を与えることができ、安定性と柔軟性の両立が可能となる。図 5.2 の下半分に提案手法のアーキテクチャを示す。図の上半分と比較すると、提案手法とベースラインの唯一の違いは誤り行動分類器の導入であることが確認できる。誤り行動分類器は、ベースエージェントを再訓練することなく統合可能であり、完全に非侵入的な設計となっている。

UCB1 アルゴリズムは多腕バンディット問題に対する古典的な解法であり、既知の報酬を活用しつつ、潜在的により良い選択肢を探索することのバランスを取るよう設計されている。各状態における行動の UCB1 スコアは次式で計算される：

$$\text{UCB}_i = \begin{cases} \bar{x}_i + 5 & \text{if } n_i = 0, \\ \bar{x}_i + \sqrt{\frac{2 \log n}{n_i}} & \text{if } n_i > 0, \end{cases} \quad (5.1)$$

ここで、

- \bar{x}_i は行動 i の平均報酬であり、RoBERTa の出力スコアに対応する
- n_i は行動 i が選択された回数
- n は現在の状態での総行動選択回数
- $\log n$ は自然対数

一度も選択されていない行動 ($n_i = 0$) には固定の高いボーナス (例:+5) が与えられ、これにより既に選択された行動よりも常にスコアが高くなることが保証される。

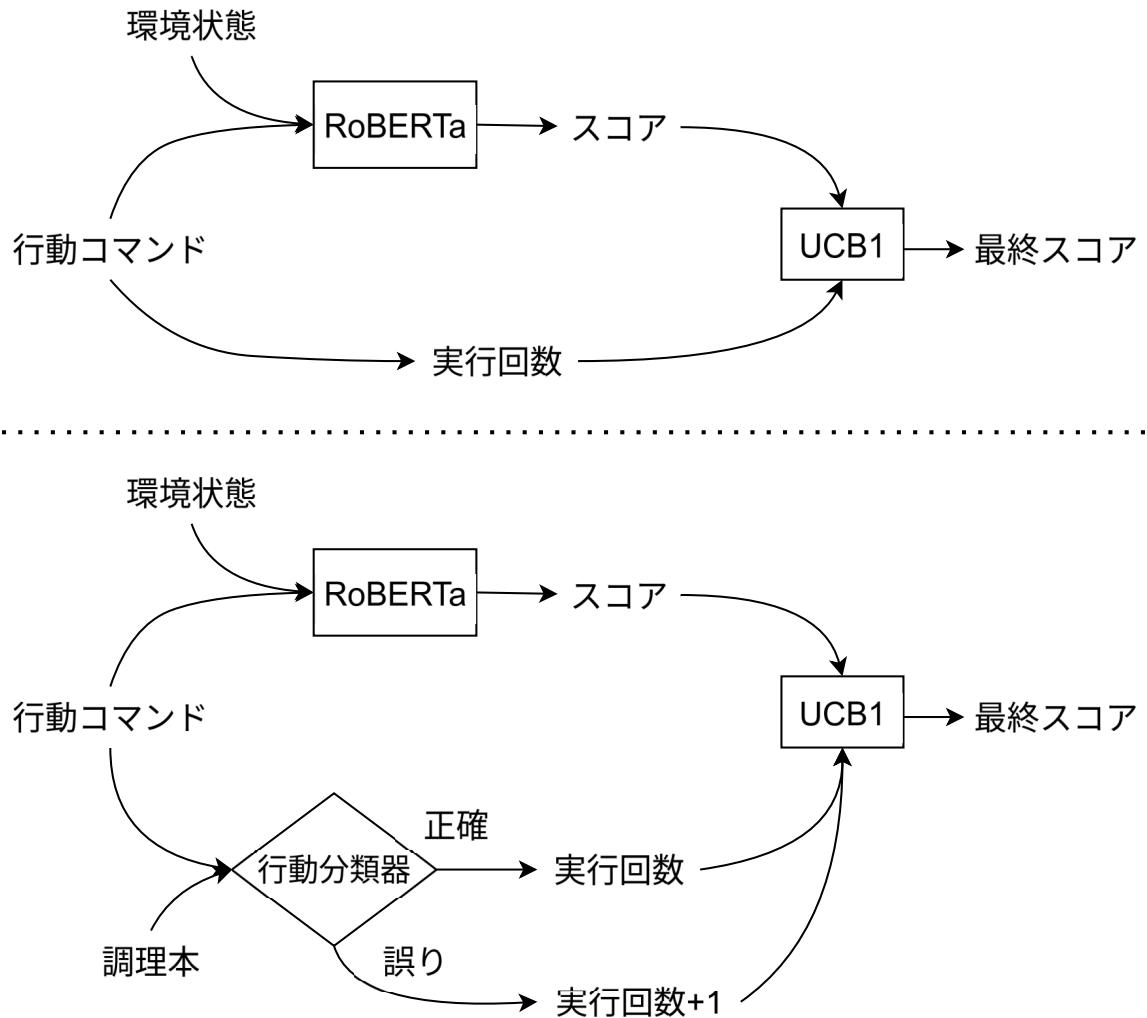


図 5.2: 上部はベースラインの CogniTextworldAgent アーキテクチャ、下部は提案手法を示す。

5.3 実験

5.3.1 データセット

本実験では、Microsoft Researchにより提供されたFirst TextWorld Problems(FTWP)データセットを使用する。FTWPは5,176件のゲームで構成されており、テキストベースのゲームエージェントの学習に利用可能な最大規模のデータセットである。各ゲームでは、エージェントがクックブックに従い調理タスクを完了する必要がある。ゲームは空間構造、使用可能な食材、クックブックの内容などが異なるため、エージェントは未経験のシナリオに対しても効果的に一般化することが求められる。データセッ

トの分割は以下のとおりである:

- トレーニングセット: 4,440 ゲーム
- 検証セット: 222 ゲーム
- テストセット: 514 ゲーム

5.3.2 評価指標

エージェントの性能評価には以下の 2 つの指標を用いる:

- **正規化スコア:** エージェントが獲得した総報酬を最大可能報酬で割った値. 各タスクの達成に 1 ポイントが付与されるため, この指標はタスク達成率を反映する.
- **平均ステップ数:** ゲームごとの行動平均回数. 正規化スコアが同等の場合, ステップ数が少ないほど効率的と評価される. FTWP コンテストのルールに従い, ゲームごとの最大ステップ数は 100 に設定される.

なお, 平均ステップ数のみでエージェントの性能を評価することは不十分である. エージェントが誤った行動を多く取る場合, ゲームが早期終了することがあり, その結果, 実際の性能が低いにもかかわらず平均ステップ数が低く算出されることがある.

5.3.3 実装詳細

本研究では, CogniTextworldAgent を再実装し, まず BERT をより性能の高い RoBERTa(RoBERTa-base) に置き換えた. トレーニング中はウォーカスルーの行動を正解として扱い, 各ステップで候補行動リストからランダムに 5 つの非正解行動をサンプリングする. モデルは AdamW オプティマイザーを用いて最大 5 エポックで学習し, 学習率は 1×10^{-5} , バッチサイズは 8 とした. 検証セットでの性能に基づき最適なチェックポイントを選択する. この手順を 3 回繰り返し, 評価用に 3 つの独立したエージェントを用意した. 誤り行動分類器についても RoBERTa-base を採用し, 学習率 1×10^{-5} , バッチサイズ 32, 最大 3 エポックで学習させた.

	Validation Set (222 games)	Test Set (514 games)
Agent 1	0.941 → 0.946	0.894 → 0.924 ▲
Agent 2	0.933 → 0.954 ▲	0.867 → 0.917 ▲
Agent 3	0.936 → 0.958 ▲	0.878 → 0.895 ▲
Average	0.936 → 0.952 ▲	0.880 → 0.912 ▲

表 5.1: 誤り行動分類器を適用する前後の正規化スコア (\rightarrow : ベースライン → 提案手法). ▲ は有意差を示す.

5.4 実験結果

表 5.1 に示すように, 誤り行動分類器の適用前後で正規化スコアが改善されることが確認できる. エージェント 1 から 3 は, それぞれ独立した実験を 3 回繰り返して得られた異なるエージェントである. 表の各セルには, ベースライン性能 (左側の値) と分類器統合後の性能 (右側の値) が示されている. 結果として, 3 つのエージェント全体にわたり一貫した改善が観察された.

平均値としては以下のとおりである:

- 検証セットにおける正規化スコアは 0.936 から 0.952 に向上し, 増加幅は 0.016 である.
- テストセットにおける正規化スコアは 0.880 から 0.912 に向上し, 増加幅は 0.032 である.

これらの改善は, 本手法の有効性を示している. さらに, スコア改善の有意性を検定した. 誤り行動分類器の追加は, ベースラインエージェントである CogniTextworldAgent の再訓練を必要としないため, ペア関係を確立できる. スコア分布が正規分布に従わないことを考慮し, 有意差評価には両側 Wilcoxon 符号順位検定 ($p < 0.05$) を採用した.

具体的には, 検証セットにおいて各エージェントは, 分類器使用前に 222 件, 使用後に別の 222 件の正規化スコアを取得し, これらの 2 つのスコアリスト間で検定を実施した. 同様に, テストセットでは各エージェントが分類器使用前後に各 514 件のスコアを取得し, 検定を行った.

「平均」とは, 3 つのエージェントのスコアを統合した上で計算した値を指す. 検定結果は各スコアの後ろに追加され, 有意差は ▲ 記号で示した.

より広範な比較結果は表 5.2 に示され, 正規化スコアと平均ステップ数の両方を含む. FTWP コンテストで 2 位となったエージェント LeDeepChef の性能も参考として示しており, 実装されていないため結果は論文から直接引用している.

CogniTextworldAgent と比較すると、本手法は平均ステップ数が高い傾向にある。これは、誤り行動の抑制によってゲームの早期終了が減少したことによると考えられる。

Method	Validation Set (222 games)		Test Set (514 games)	
	Norm. Score	Avg. Steps	Norm. Score	Avg. Steps
LeDeepChef	0.744	24.1	0.693	43.9
CogniTextworldAgent	0.936	14.5	0.880	21.6
Ours	0.952 ▲	15.3	0.912 ▲	24.7

表 5.2: 検証セットおよびテストセットにおける性能比較。▲は CogniTextworldAgent に対する統計的有意性を示す。

5.5 考察

5.5.1 失敗パターンの分析

FTWP ゲームにおける失敗パターンは大きく二種類に分類される:

- 最大ステップ数の超過 (FTWP コンテストのルールに基づく)
- 誤り行動 (クックブックに反する行動) の実行

誤り行動分類器がエージェントの挙動に与える影響を検証するため、分類器を適用する前後における失敗数を、テストセット (514 ゲーム) 全体で集計した。結果を表 5.3 に示す。

	Exceeded Step Limit	Performed Wrong Action	Total Failures
Agent 1	16 → 31	80 → 51	96 → 82
Agent 2	3 → 15	100 → 67	103 → 82
Agent 3	13 → 38	108 → 60	121 → 98
Average	10.7 → 28.0	96.0 → 59.3	106.7 → 87.3

表 5.3: テストセットにおける誤り行動分類器適用前後の失敗回数 (→: 適用前 → 適用後)。Agent1 から 3 は初期値の異なるエージェントを表す。

分類器の導入後、以下の傾向が確認された:

- 誤り行動による失敗は大幅に減少した。
- ステップ数超過による失敗は増加した。

- 総失敗数は減少し、エージェントの安定性向上が示唆された。

特に 2 点目について、ステップ数超過の増加は想定とおりの結果である。誤り行動を抑制することでゲームの早期終了が減少し、その結果としてより多くの行動を選択する機会が生じ、ステップ数上限を超過する確率が高まったと考えられる。それでもなお、全体の失敗回数が減少している点は、誤り行動分類器の有効性を強く示している。

5.5.2 UCB1 との併用の意義

本研究では、誤りと分類された行動を単純に除去するのではなく、UCB1 アルゴリズムを用いて行動スコアにペナルティを与える方式を採用した。では、直接の行動除去と比較して、UCB1 との併用にはどの程度の意味があるのだろうか。

今回の実験では、分類器の精度が高く、誤りでない行動を誤分類することが確認されなかったため、直接行動を除去しても性能低下は生じないと予測される。実際、検証セットにおいても非誤り行動の識別精度は 100% であった。したがって、正しい行動を誤って除去するリスクが低い。

しかしながら、UCB1 との統合の重要性を示すために、あえて「すべての潜在的に報酬を得られる行動」を誤分類する極端な分類器を構築し、以下の 2 条件で CogntextworldAgent に適用した：

- UCB1 と併用 (提案手法),
- 直接に行動を除去.

FTWP ゲームにおいて「潜在的に報酬のある行動」には、食材処理行動に加え、重要な食材を獲得する際に得点が与えられる `take` 行動も含まれる。結果を表 5.4 に示す。

	Validation Set (222 games)	Test Set (514 games)
Agent 1	0 → 0.476	0 → 0.458
Agent 2	0 → 0.464	0 → 0.435
Agent 3	0 → 0.469	0 → 0.442
Average	0 → 0.470	0 → 0.445

表 5.4: 極端な分類器を用い、潜在的に報酬のあるすべての行動を誤分類した場合の正規化スコア (→: 行動除去 → UCB1 との併用)。エージェント 1 から 3 は初期値の異なる設定である。

予想とおり、潜在的に報酬を得られる行動をすべて誤分類した場合、行動を直接除去する方式ではスコアは 0 に固定された。一方、UCB1 と併用する我々の手法では、検証セットで平均 0.470、テストセットで 0.445 のタスク達成率を示した。

この結果は、分類器性能が極端に低い条件下においても、UCB1 と統合することで安定性と効果が確保されることを明確に示している。

5.5.3 LLM の性能

近年、大規模言語モデル(LLM)の発展は目覚ましく、幅広いタスクにおいて顕著な成果を収めている。本節では、テストセットからランダムに抽出した 10 個のゲームを対象に、GPT-4.1-mini の性能を評価する。実験設定としては、OpenAI Playground 上で gpt-4.1-mini を使用し、すべてのパラメータをデフォルト値に固定した。以下に、使用したプロンプトの一例を示す。

```
Task: Find and examine the cookbook, follow the instructions to
prepare the meal, and eat it.

Room: Kitchen

Inventory: some water. Total 1 items.

Recipe: missing

Available actions:

0 drop water
1 examine cookbook

Next action (answer with the action directly):
```

ここで Task は、エージェントに対してクックブックを見つけ、その内容に従ってタスクを進めるように指示している。対話記録に残られるため、タスク説明を最初の対話時に与えるで十分である。その他の情報は、CogniTextworldAgent および提案手法と共にものを採用している。具体的には、Room は現在の部屋名を示し、Inventory は現在所持しているアイテムとその数を列挙する。Recipe はクックブックの内容を表示するが、examine cookbook を実行する前は「missing」と表示される。Available actions には、各ステップでエージェントが選択可能な行動が列挙されている（ゲームエンジンに直接に請求できる）。上記プロンプト例に対し、GPT-4.1-mini は「1 examine cookbook」と応答した。10 本のランダムに選ばれたテストゲームにおける各手法の性能を、表 5.5 に示す。

Method	Normalized Score	Average Steps
LLM (GPT-4.1-mini)	0.395	7.4
CogniTextworldAgent	0.868	15.3
Ours	0.882	21.4

表 5.5: ランダムに選択した 10 本の FTWP テストゲームにおける性能比較.

表からわかるように、GPT-4.1-mini は正規化スコア 0.395(全 48 タスク中)，平均ステップ数 7.4 を達成した。行動の詳細を分析すると，低い平均ステップ数は誤り行動を頻繁に選択した結果であることが確認された。この結果は，微調整を行わない大規模言語モデルが FTWP ゲームの複雑さに対応することの困難さを示している。特に，料理手順や調理器具の対応関係といったタスク固有の知識は，学習を通じて獲得されなければならない。ゼロショット性能の優れた LLM であれば，より高いスコアを達成できると予想されるが，誤り行動は依然として主要な障害として残ると考えられる。

例えば，cookbook に「赤玉ねぎを炒める」と記されており，エージェントが赤玉ねぎを持ちながら「オープンはあるがコンロはない」部屋にいる場合を考える。このとき，エージェントは別の部屋を探索すべきか，それとも「オープンで赤玉ねぎを調理する」と試みるべきかを判断する必要がある。「炒める」という行動と「コンロ」との関連性を学習したモデルであれば探索を選択する可能性が高いが，現状の LLM にはそのような柔軟な判断は難しい。

LLM のプレイ例

参考として，GPT-4.1-mini のプレイ例を以下に示す。

Task: Find and examine the cookbook, follow the instructions to prepare the meal, and eat it.

Room: Bedroom

Inventory: a chicken wing, a raw chicken leg. Total 2 items.

Recipe: missing

Available actions:

0 go south

1 drop chicken leg

2 go east

3 drop chicken wing

Next action (answer with the action directly):

> 0

Executed action 1: go south

Room: Livingroom

Inventory: a chicken wing, a raw chicken leg. Total 2 items.

Recipe: missing

Available actions:

0 drop chicken wing

1 go north

2 drop chicken leg

Next action (answer with the action directly):

> 2

Executed action 2: drop chicken leg

Room: Livingroom

Inventory: a chicken wing. Total 1 items.

Recipe: missing

Available actions:

0 drop chicken wing

1 go north

Next action (answer with the action directly):

> 1

Executed action 3: go north

Room: Bedroom

Inventory: a chicken wing. Total 1 items.

Recipe: missing

Available actions:

- 0 go south
- 1 drop chicken wing
- 2 go east

Next action (answer with the action directly):

> 2

Executed action 4: go east

Room: Corridor

Inventory: a chicken wing. Total 1 items.

Recipe: missing

Available actions:

- 0 go south
- 1 drop chicken wing
- 2 go east
- 3 go west

Next action (answer with the action directly):

> 2

Executed action 5: go east

Room: Kitchen

Inventory: a chicken wing. Total 1 items.

Recipe: missing

Available actions:

- 0 go west
- 1 go south
- 2 examine cookbook
- 3 drop chicken wing
- 4 go east

Next action (answer with the action directly):

> 2

Executed action 6: examine cookbook

Room: Kitchen

Inventory: a chicken wing. Total 1 items.

Recipe: Ingredients: carrot chicken wing olive oil red onion yellow bell pepper Directions: dice the carrot grill the carrot fry the chicken wing slice the red onion roast the red onion chop the yellow bell pepper grill the yellow bell pepper prepare meal

Available actions:

0 take yellow bell pepper

1 cook chicken wing with stove

2 take knife

3 drop chicken wing

4 go west

5 go south

6 take carrot

7 take red onion

8 go east

9 cook chicken wing with oven

Next action (answer with the action directly):

> 6

Executed action 7: take carrot

Reward received: 1

Room: Kitchen

Inventory: a diced grilled carrot, a chicken wing. Total 2 items.

Recipe: Ingredients: carrot chicken wing olive oil red onion yellow bell

pepper Directions: dice the carrot grill the carrot fry the chicken wing slice the red onion roast the red onion chop the yellow bell pepper grill the yellow bell pepper prepare meal

Available actions:

- 0 cook carrot with oven
- 1 go west
- 2 drop carrot
- 3 take knife
- 4 cook chicken wing with oven
- 5 cook chicken wing with stove
- 6 take yellow bell pepper
- 7 go south
- 8 drop chicken wing
- 9 take red onion
- 10 cook carrot with stove
- 11 go east

Next action (answer with the action directly):

> 9

Executed action 8: take red onion

Reward received: 2

Room: Kitchen

Inventory: a diced grilled carrot, a sliced roasted red onion, a chicken wing.

Total 3 items.

Recipe: Ingredients: carrot chicken wing olive oil red onion yellow bell pepper Directions: dice the carrot grill the carrot fry the chicken wing slice the red onion roast the red onion chop the yellow bell pepper grill the yellow bell pepper prepare meal

Available actions:

- 0 drop red onion

1 cook chicken wing with stove

2 cook carrot with stove

3 go west

4 cook carrot with oven

5 take knife

6 go south

7 cook chicken wing with oven

8 go east

9 drop chicken wing

10 take yellow bell pepper

11 cook red onion with stove

12 cook red onion with oven

13 drop carrot

Next action (answer with the action directly):

> 10

Executed action 9: take yellow bell pepper

Reward received: 3

Room: Kitchen

Inventory: a diced grilled carrot, a chopped grilled yellow bell pepper, a chicken wing, a sliced roasted red onion. Total 4 items.

Recipe: Ingredients: carrot **chicken wing** olive oil red onion yellow bell pepper Directions: dice the carrot grill the carrot fry the chicken wing slice the red onion roast the red onion chop the yellow bell pepper grill the yellow bell pepper prepare meal

Available actions:

0 cook yellow bell pepper with oven

1 cook carrot with oven

2 cook red onion with oven

3 cook red onion with stove

- 4 cook chicken wing with oven
- 5 cook carrot with stove
- 6 drop chicken wing**
- 7 take knife
- 8 go east
- 9 cook yellow bell pepper with stove
- 10 cook chicken wing with stove
- 11 drop red onion
- 12 go west
- 13 drop carrot
- 14 drop yellow bell pepper
- 15 go south

Next action (answer with the action directly):

> **6**

Executed action 10: drop chicken wing

Reward received: 3

Room: Kitchen

Inventory: a diced grilled carrot, a chopped grilled yellow bell pepper, a sliced roasted red onion. Total 3 items.

Recipe: Ingredients: carrot chicken wing olive oil red onion yellow bell pepper Directions: dice the carrot grill the carrot fry the chicken wing slice the red onion roast the red onion chop the yellow bell pepper grill the yellow bell pepper prepare meal

Available actions:

- 0 cook red onion with stove
- 1 go south
- 2 cook carrot with oven
- 3 go west
- 4 cook red onion with oven

5 take knife
6 take chicken wing
7 go east
8 cook yellow bell pepper with stove
9 drop red onion
10 drop carrot
11 cook yellow bell pepper with oven
12 cook carrot with stove
13 drop yellow bell pepper

Next action (answer with the action directly):

> 6

Executed action 11: take chicken wing

Reward received: 3

Room: Kitchen

Inventory: a diced grilled carrot, a chopped grilled yellow bell pepper, a chicken wing, a sliced roasted red onion. Total 4 items.

Recipe: Ingredients: carrot chicken wing olive oil red onion yellow bell pepper Directions: dice the carrot grill the carrot **fry the chicken wing** slice the red onion roast the red onion chop the yellow bell pepper grill the yellow bell pepper prepare meal

Available actions:

0 cook red onion with oven
1 go east
2 drop red onion
3 cook chicken wing with stove
4 drop chicken wing
5 take knife
6 cook chicken wing with oven
7 go south

8 cook yellow bell pepper with oven
9 drop yellow bell pepper
10 cook carrot with stove
11 drop carrot
12 cook yellow bell pepper with stove
13 cook red onion with stove
14 go west
15 cook carrot with oven

Next action (answer with the action directly):

> **6**

この例において注目すべき点は二つある（これらの箇所はすでにハイライトで示されている）。第一に、最終ステップにおいて、cookbook が「fry the chicken wing」を要求した際に、GPT-4.1-mini は「cook chicken wing with oven」を選択した。これは誤り行動の例であり（正しい答えは「cook chicken wing with stove」である）、その結果としてゲームが早期終了してしまった。第二に、第十ステップにおいて（「Executed action 9: take yellow bell pepper」で始まる部分）、cookbook が chicken wing を要求した際に、GPT-4.1-mini は「drop chicken wing」を選択した。この行為はタスクの達成に役立たないが、ゲームを早期終了させるわけではないため、無駄行動の例とみなされる。その直後のステップにおいて、GPT-4.1-mini はすぐに「chicken wing」を拾い上げている。

我々のエージェントが実行した行動の一覧は以下のとおりである（訓練済みエージェントの一つをランダムに選択）。

go south
go north
go east
go east
examine cookbook
drop chicken leg
cook chicken wing with stove

take carrot

go east

take olive oil

go west

take red onion

take yellow bell pepper

prepare meal

eat meal

ハイライト部分は点数をもらったステップである。この例から、我々のエージェントは誤り行動を回避しつつ、タスクを完了するために必要な行動を適切に選択できていることがわかる。

5.6 結言

本研究では、ゲームの早期終了を招くクックブック違反行動を特定する誤り行動分類器を導入し、FTWP データセットにおける最先端エージェントである CogniTextworldAgent を強化する手法を提案した。さらに、UCB1 との組み合わせにより、ベースエージェントを再訓練することなく安定性を向上できることを示した。FTWP データセットでの実験の結果、提案手法はテストセットにおける正規化スコアを 0.880 から 0.912 へと改善し、この性能向上は統計的に有意であることが確認された (Wilcoxon 符号順位検定, $p < 0.05$)。これらの結果から、ベースエージェントの再訓練を必要とせずに安定性を高められる誤り行動分類器の有効性が実証されたといえる。

第6章 ナビゲータによるFTWP エージェントの性能向上

テキストゲームエージェントの訓練において、「go east」のような具体的な移動コマンドを正解として扱うと、学習目標が曖昧になり、学習効率が低下するという問題がある。この課題に対処するため、我々は冗長な移動シーケンスの代わりに「navigate to red onion」のようなアイテムナビゲーションコマンドを導入し、エージェント性能の向上を試みた。実験の結果、ナビゲーションコマンドを使用可能なエージェントはFTWP テストセットにおいて 0.937 のタスク完了率を達成した。さらに、前章で紹介した誤り行動分類器を併用することで、正規化スコアは 0.968 に到達した。この性能は、従来最高水準であった CogniTextworldAgent(0.880) を大きく上回り、両側 Wilcoxon 順位和検定により有意差 ($p < 0.05$) が確認された。

6.1 先行研究とその問題点

FTWP ゲームの詳細は 5.1 節に記載したとおりであり、本節では FTWP における先行研究とその問題点について述べる。

FTWP コンテスト (2019) において、優勝エージェント CogniTextworldAgent[7] は、BERT[4] と *Upper Confidence Bound 1*(UCB1)[15] アルゴリズムを用い、トレーニングセットのウォークスルーに基づく模倣学習で訓練された。BERT は事前学習済み transformer モデルの代表例であり、強力な文脈理解能力を持ち、自然言語理解タスクで広く採用されている。一方、UCB1 は多腕バンディット問題に対する古典的手法であり、頻繁に選択される行動にペナルティを課すことで、探索と利用のバランスを取る。模倣学習は強化学習において広く用いられる手法であり、専門家の行動（ウォークスルー）を模倣してタスク解決を学習する。

しかし模倣学習には問題がある。すなわち、ウォークスルー内のすべての行動が等しく重要であると仮定してしまう点である。次の抜粋を考察する。

```
[..., "examine cookbook", "go west",
"go south", "take red onion", ...]
```

ここで *go west* や *go south* は正解行動とされるが、移動コマンド自体には報酬がなく、その正解性は曖昧である。これらは単に最終目標である *take red onion* に至るための中間ステップにすぎない。さらに、同一の移動コマンドが文脈により異なる目的に対応することもありうる。そのため、模倣学習で訓練されたエージェントは、行動の根底にある目的を理解しないまま移動を模倣する傾向がある。先行研究 [27, 48] も、このような曖昧な教師信号が方策学習 (policy learning)、強化学習においてタスク達成のための戦略を学習する過程) を著しく阻害しうることを示している。

この制約を緩和するため、我々はアイテムナビゲーションコマンドの導入を提案する。このコマンドは、移動シーケンスを单一の目標指向行動 (例: *navigate to red onion*) に集約する。これによりエージェントの目的が明示化され、教師信号の曖昧さが大幅に軽減される。

FTWP 環境は本来ナビゲーションコマンドを実行できないため、我々はゲームエンジンを拡張し、ナビゲータモジュールを実装した。このモジュールは以下のように機能する。

- 適切なタイミングで候補行動リストにナビゲーションコマンドを追加する。
- コマンドが選択された場合にエージェントを目標アイテムのある部屋へ移動させる。

ナビゲータモジュールの実装自体は単純だが、課題はエージェントがこれらのコマンドを有効に活用できるようにする点にある。そこで我々は、訓練セットに含まれる可能な限り多くの移動シーケンスをナビゲーションコマンドで置き換えるよう修正し、修正版データでエージェントを再訓練した。その結果、我々の手法は FTWP において state-of-the-art の性能を達成した。

6.2 手法

6.2.1 ナビゲータ

本研究では、エージェントに適切なタイミングでナビゲーションコマンドを提示し、その実行によって目的のアイテムが存在する部屋へ自動的に到達できる仕組みの構築

を目指す。この考えに基づき、我々は FTWP 環境を拡張し、ナビゲータを設計した。ナビゲータは以下の 3 つの要素から構成される：

- 探索によって逐次更新される地図
- 「アイテム - 部屋」の対応関係を保持するアイテム記録
- 幅優先探索 (BFS) アルゴリズムに基づく経路探索関数

地図はグラフ構造として保存され、各ノードは部屋の名前を、エッジは移動可能な方向（最大で東・南・西・北の 4 方向）を表す。実装においては、より扱いやすい形として Python の辞書を用いた。例えば、図 6.1 に示す家の構造は以下のように表現できる。

```
map = {
    "bed room": {
        "north": "corridor"
    },
    "corridor": {
        "east": "kitchen",
        "south": "bed room"
    },
    "kitchen": {
        "west": "corridor"
    }
}
```

アイテム記録も Python 辞書で実装し、アイテム名をキー、対応する部屋名を値とする。以下の例では、赤玉ねぎが寝室に、オーブンと料理本が厨房に位置することを示している。マップと同様に、アイテム記録も初期状態では空であり、エージェントが探索を進める過程で新たなアイテム（食材や調理器具を含む）が発見されるたびに更新される。

```
itemRecord = {
    "red onion": "bed room",
    "oven": "kitchen",
```

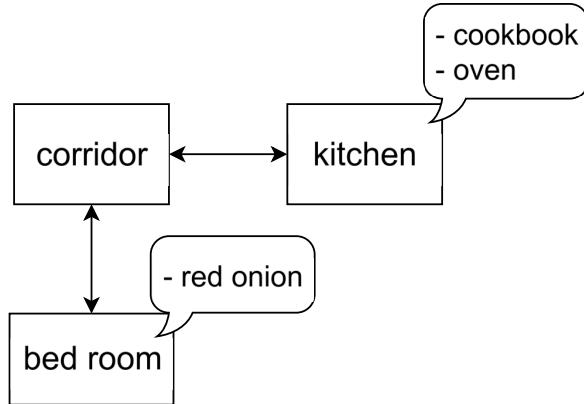


図 6.1: 3 つの部屋とそれらに配置されたアイテムからなる FTWP ゲーム環境の空間構造

```
"cookbook": "kitchen"
```

```
}
```

経路探索関数は BFS(幅優先探索) アルゴリズムに基づき実装されている。BFS は古典的な経路探索手法であり、連結している限りグラフ内の任意の 2 ノード間の最短経路を求めることができる。本研究では、この BFS を用いて地図上の既知の 2 つの部屋間の経路を算出する。

以上の設計に基づき、ナビゲータは以下のように機能する。

1. エージェントが新しい部屋に入るたびに地図を更新する。これは、FTWP 環境における部屋の配置が事前には不明であり、探索によってのみ明らかになるためである。
2. 各行動の後、その部屋に存在するアイテムの情報を更新する（アイテムは行動によって移動する可能性があるため）。
3. エージェントが料理本を確認した後、必要なアイテムに対応するナビゲーションコマンドを行動候補に追加する。

アイテムナビゲーションコマンドはアイテム記録から生成される。具体的には、テンプレート *navigate to <item>* にアイテム名を埋め込むことで作成される。ただしノイズを減らすため、以下の 2 種類のアイテムに関するコマンドは除外する。すなわち、

1. エージェントが現在いる部屋に存在するアイテム、

2. 料理本に記載されていない食材である.

例えば, 料理本が *red onion* のみを必要としている場合, *navigate to apple* は不要なため生成されない.

アイテムナビゲーションコマンドが実行されると, ナビゲータは経路探索関数を用いて目的アイテムの所在する部屋までの最短経路を計算し, その経路を一連の移動コマンドに変換してゲームエンジンに順次入力する. この仕組みにより, エージェントは目的のアイテムが存在する部屋に確実に到達できる.

図 6.2 は, ナビゲータがエージェントにアイテムナビゲーションコマンドを提示する過程を示している. 例として, 2 部屋のみからなる簡易化された FTWP 環境を考える. エージェントは寝室から開始し, そこにはリンゴが置かれている. この時点では他の部屋はまだ探索されていないため, *unknown* とマークされている. *Go east* コマンドを実行すると, エージェントは新たにキッチンを発見し, そこに料理本があることが分かる. 料理本を調べた結果, リンゴが必要な食材であると判明する. このとき, ナビゲータのアイテム記録にはリンゴの情報が含まれているため, *navigate to apple* コマンドが候補リストに追加される.

以上のように, ナビゲータはエージェントにアイテムナビゲーションコマンドを提供し, 選択された場合にはその実行を支援する. しかし, 既存のエージェントはこれらのコマンドをまだ利用できない. 次のステップとして, エージェントがナビゲーションコマンドを理解し活用できるように訓練する必要がある. これについては次のサブセクションで詳しく述べる.

6.2.2 トレーニングセットの修正

ナビゲータの導入により, エージェントはナビゲーションコマンドを利用できるようになった. しかし, その使用方法をエージェントに学習させる必要がある. そこで我々は, 訓練集のウォークスルーを更新し, 可能な限り多くの移動シーケンスをナビゲーションコマンドに置き換えることを試みた. 手順は以下のとおりである:

1. ナビゲータ付きのエージェントにウォークスルーを逐次実行させ, 最初に現れる移動コマンドを検出する (この時点では, 移動コマンド自体はまだ実行されない).

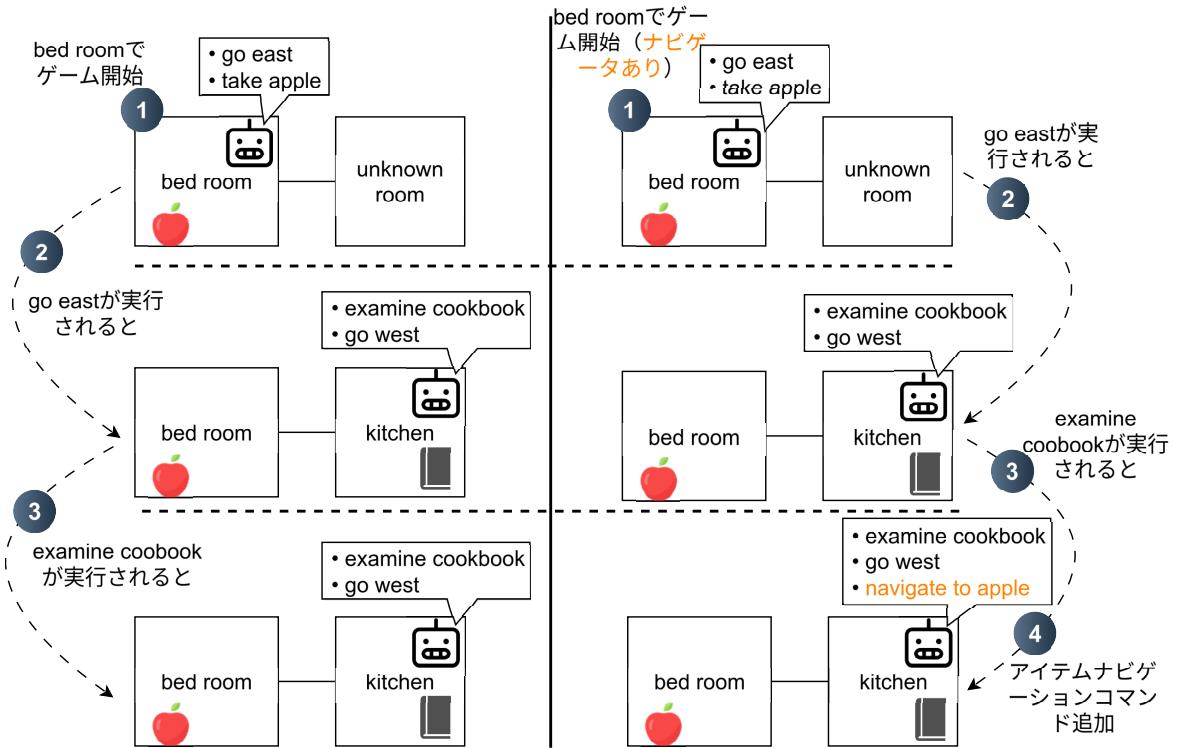


図 6.2: 左側はナビゲータを持たないベースラインエージェント、右側はナビゲータを有効にしたエージェントを示す。右側のようにナビゲータが有効な場合、*examine cookbook* コマンド(ステップ 3)の実行後にナビゲーションコマンドが候補行動リストへ追加され、以前に寝室で観察された *apple* に向けてエージェントを導くことができる。

2. 当該移動コマンドから後方に遡り、最初に現れる非移動コマンド(例:*take red onion*)を特定し、対象アイテムを抽出する。
3. そのアイテムがナビゲータの記録に存在する場合は、移動シーケンスを *navigate to <item>* に置換して実行し、存在しなければスキップする。

例として、以下のウォークスルーを考える：

```
["inventory", "go north", "go east",
 "examine cookbook", "go west",
 "go south", "take red onion", ...]
```

まず、エージェントは「*go north*」までを順に実行する。次に、「*go north*」以降で最初に現れる非移動コマンドである「*examine cookbook*」を特定し、対象アイテム *cookbook* を抽出する。しかし、この時点ではナビゲータの記録に *cookbook* の情報が

存在しないため、このシーケンスは置換されず、そのままウォーカスルーを実行する。続いて「go west」に注目し、直後の非移動コマンド「take red onion」を特定する。ここで red onion はナビゲータに記録されているため、この移動シーケンスは navigate to red onion に置換される。結果として、更新後のウォーカスルーは次のようになる：

```
["inventory", "go north",
 "go east", "examine cookbook",
 "navigate to red onion",
 "take red onion", ...]
```

訓練集には 4,440 のゲームが含まれ、全ウォーカスルーのコマンド数は合計 50,941 である。そのうち移動コマンドは 11,993 個 (24.0%) を占める。置換プロセスを適用した結果、686 箇所で置換が行われ、最終的に 1,630 個の移動コマンドがナビゲーションコマンドに置き換えられた。これは全移動コマンドの 13.6% に相当する。

6.2.3 ナビゲータとエージェントの統合

前節では、ナビゲータの仕組みを述べた。要約すると、ナビゲータはエージェントにアイテムナビゲーションコマンドを提供し、エージェントがそのコマンドを選択すると、対応するアイテムの所在部屋まで導く。この機能をエージェントに学習させるため、我々は訓練集のウォーカスルーを修正し、可能な限り多くの移動シーケンスをナビゲーションコマンドに置換したデータセットを用いて再訓練を行う。

本節では、ナビゲータが CogniTextworldAgent(および CogniTextworldAgent + Action Classifier) に統合される仕組みを説明する。焦点は RoBERTa への入力形式に置く。すなわち、入力構造を明確化することで、エージェントがどのようにナビゲータを利用するのかを示す。概要として、アイテムナビゲーションコマンドはエージェントの候補行動リストに追加される。

CogniTextworldAgent では、環境から得られる観測は以下の形式に整形され、RoBERTa に入力される：

```
[CLS] <inventory count>
<inventory description> <cookbook>
<environment description> [SEP] <action>
```

ここで、[CLS] および [SEP] は BERT[4] に由来する特殊トークンである。前者はシーケンスの開始を表し、後者は異なる入力コンポーネントを区切る。我々の再実装では、BERT をより高性能な RoBERTa[2] に置換しており、対応する特殊トークンも利用している。*inventory count* は所持アイテム数を示し、エージェントがアイテム過多であるかを判断する助けとなる[7]。*inventory description* は所持アイテムの具体的な内容を記述し、*cookbook* はクックブックの内容を表す。*environment description* は部屋の名前とその説明文を含むが、長すぎる場合には RoBERTa が処理可能な長さに切り詰められる。最後に *action* は候補コマンドの一つを表し、テスト時には最高スコアを得たコマンドが出力として選択される。

すなわち、ナビゲータは候補コマンドリストにナビゲーションコマンドを追加するだけでもよく、ナビゲーションコマンドは *action* のところに埋められ、他の候補コマンドと同様にスコアリングされる。

候補コマンドリストの生成方法は複数存在する。元の CogniTextworldAgent では、固有表現抽出とコマンドテンプレートを組み合わせた複雑な戦略を採用し、コンテストの評価ルール下で高いスコアを実現した。¹ これらのルールは、ゲームエンジンから直接コマンドリストを取得する方法を低く評価し、生成または合成された方法を高く評価する。しかし、精緻な実装で高いスコアを獲得するのが本研究の目的ではなく、それより再現性が重要視される。そのため我々の再実装では、ゲームエンジンから直接候補コマンドリストを取得し、ルールベースのフィルタを適用することで不要なコマンド（例：*put* や *close* などのコマンド）を除去し、訓練効率を高めている。詳細なルールについては 6.3.3 節で説明する。本研究で実装されたすべてのエージェントは、同一状態に対して同一の候補コマンドリストを共有する。

図 6.3 に、我々の最高性能エージェントである CogniTextworldAgent + Action Classifier + Navigator のアーキテクチャを示す。CogniTextworldAgent の部分は RoBERTa と UCB1 アルゴリズムで構成される。RoBERTa は観測（すなわち環境情報、例えば *inventory* や *environment description*）に基づいて行動（コマンド）をスコアリングする。UCB1 はさらに各行動の実行回数を考慮してスコアを調整し、頻繁に実行された行動にはより低いスコアを与える。

Action Classifier は第 5 章で導入されたもので、行動が誤り、すなわちクックブックの指示と矛盾しているかを判定する。例えば、クックブックに *slice red onion* とある

¹ ルール詳細: https://competitions.codalab.org/competitions/21557#learn_the_details-evaluation

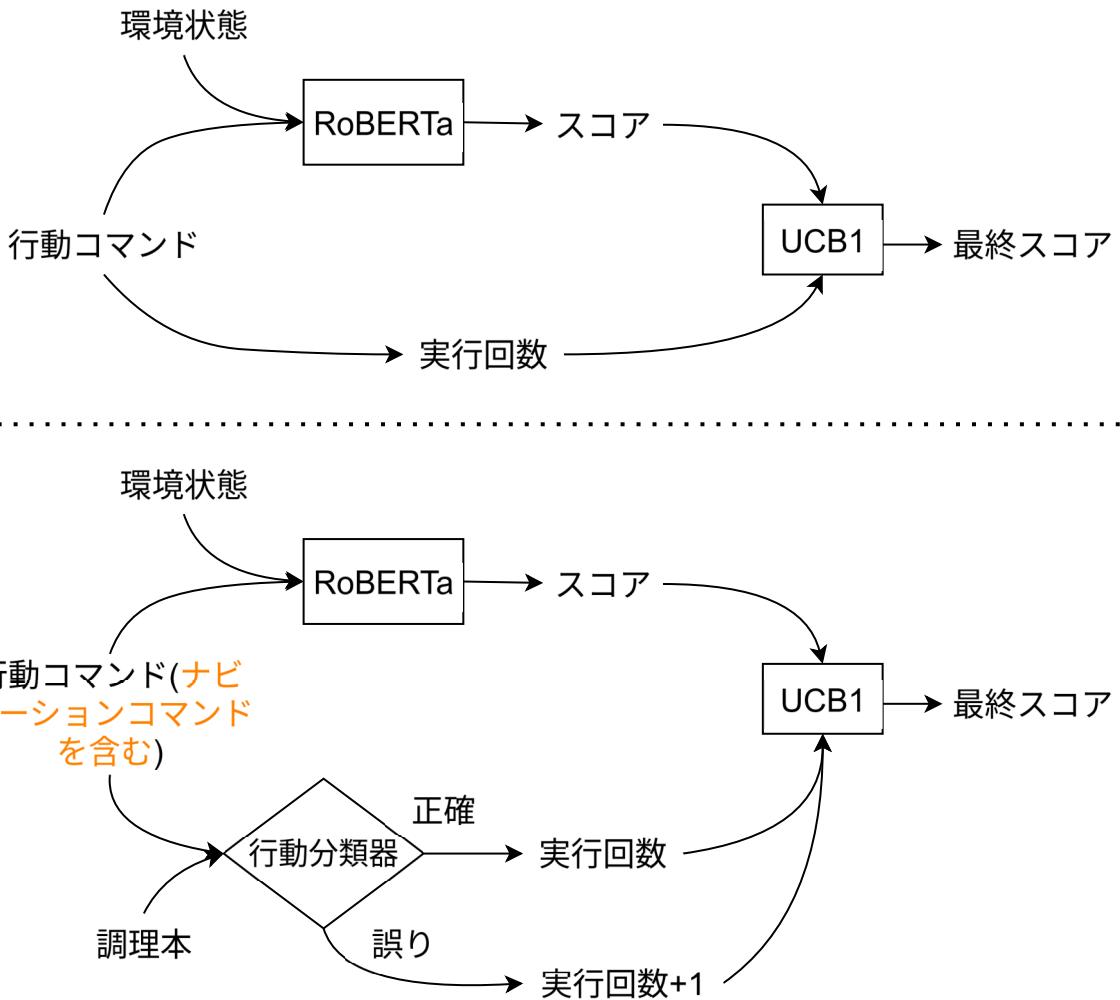


図 6.3: 我々の最高性能エージェント CogniTextworldAgent + Action Classifier + Navigator のアーキテクチャ概要

にもかかわらず, *dice red onion with knife* を実行すると矛盾が生じる. 玉ねぎを一度みじん切りにした後では, スライスはできないからである. もし行動が誤りと判定されれば, その行動の実行回数は人為的に 1 回増加し, UCB1 により最終スコアが減少する.

以上のように, ナビゲータはアイテムナビゲーションコマンドを候補コマンドリストに追加し, それらは他の行動と同じ方式でスコアリングされる. エージェントは通常の行動と同様にナビゲーションコマンドを評価し, 最終的に最も高いスコアを持つものを出力として選択する.

6.3 実験

6.3.1 データセット

Microsoft Research が公開した First TextWorld Problem(FTWP) データセットは、公開されているテキストゲームデータセットの中で最大規模であり、5,176 個のゲームインスタンスから構成される。FTWP におけるエージェントの目的は、クックブックの指示に従って食事を調理し、それを食べて空腹を満たすことである。各ゲームは空間構造や食材、クックブックの内容が異なるため、テストセットで良い性能を得るには強い汎化能力が求められる。データセットは train(4,440 ゲーム), validation(222 ゲーム), test(514 ゲーム) の 3 分割で提供されている。

6.3.2 評価指標

エージェントの性能は、正規化スコアと平均ステップ数の 2 指標で評価する。正規化スコアは、全ゲームで得られた合計スコアを獲得可能な最大スコアで割った値である。5.1 節で述べたように、FTWP の各タスクの報酬は 1 点であるため、正規化スコアは完了したタスクの割合に対応する。

平均ステップ数は、1 ゲームあたりの平均行動数を表す。同一スコアに対してステップ数が少ないほど効率的であることを意味する。FTWP ではエピソードは最大 100 ステップで強制終了する。また、タスクを完了した場合や、クックブックに違反する行動(例: *slice red onion* とあるのに *dice red onion with knife* を実行し、もはやスライス不可能になる場合)を行った場合にも終了する。このような誤り行動については第 5 章で詳しく述べた。なお平均ステップ数は単独では指標として不十分である。誤り行動を頻繁に行えば、ゲームが早期終了し見かけ上ステップ数が少なくなるためである。

6.3.3 実装詳細

ベースラインである CogniTextworldAgent の再実装に際し、以下の調整を行った。

- BERT[4] の代わりに高性能な RoBERTa[2] を使用。
- 学習ではウォークスルー中のコマンドを正例とし、同一状態で可能な他のコマンドを負例とした。負例は学習の安定性のため最大 5 個に制限し、候補行動リストから重複なしでランダム抽出した。

Agent	Valid set			Test set		
	Norm.	Score	Avg. Steps	Norm.	Score	Avg. Steps
LeDeepChef	0.744	24.1	0.693	43.9		
CogniTextworldAgent	0.934	14.5	0.880	21.6		
+ Navigator	0.994	▲	12.9	0.937	▲	21.2
- Navigator(テスト時)	0.955		15.4	0.890		25.4
CogniTextworldAgent + Action Classifier	0.952	15.3	0.912	24.7		
+ Navigator	0.999		13.1	0.968	▲	22.8
- Navigator(テスト時)	0.965		16.1	0.918		28.2

表 6.1: FTWP データセットにおける性能比較.“+ Navigator”は提案手法, ▲は導入による改善が統計的に有意であることを示す.“- Navigator”はテスト時にナビゲータを外したアプリケーション.

- 最適化には AdamW[49] を用い, 学習率は 1×10^{-5} , バッチサイズは 8.
- 各エージェントは最大 5 エポック学習し, 検証性能が最良のモデルを採用.
- 初期化のランダム性を考慮し, 学習を 3 回繰り返し異なる 3 エージェントを得た.

誤り行動分類器(第 5 章参照)も RoBERTa を用いて実装し, 学習率 1×10^{-5} , バッチサイズ 32, 最大 3 エポックで学習した.

ナビゲータを CogniTextworldAgent に適用した場合, 手法を「CogniTextworldAgent + Navigator」と呼ぶ. エージェントがアイテムナビゲーションコマンドを実行可能にするため, 第 6.2.2 節で説明した訓練集を用いて再訓練を行った. ハイパーパラメータは CogniTextworldAgent と同一であり, 学習は異なる初期値で 3 回実施した.

さらに誤り行動分類器と組み合わせた手法を「CogniTextworldAgent + Navigator + Action Classifier」と呼ぶ. ハイパーパラメータ設定は第 5 章と同じであり, 誤り行動分類器の統合に追加の再訓練は不要である.

また第 6.2.3 節で述べたように, すべてのエージェントはゲームエンジンから得た同一の候補行動リストを用い, 不要な行動をフィルタリングして学習を加速させる. 具体的には[“examine”, “close”, “eat”, “look”, “drink”, “put”, “insert”]で始まるコマンドを削除し, 例外として「examine cookbook」と「eat meal」のみを残した. これは CogniTextworldAgent の元実装に準拠している.

6.4 実験結果

実験結果を表 6.1 に示す.

表 6.1 に示すように、提案手法 (“+ Navigator”) により性能が一貫して向上した。CogniTextworldAgent[7] にナビゲータを導入すると、テストセットの正規化スコアは 0.880 から 0.937 へ上昇し、増加幅は 0.057 であった。この改善は両側 Wilcoxon rank-sum test($p < 0.05$) により有意であることを確認した。検証セットにおいても 0.934 → 0.994 と同様の改善が得られ、手法の安定性が示された。

さらに誤り行動分類器と組み合わせた場合、テストスコアは 0.912 から 0.968 に上昇し（増分は 0.056）、同様に有意差が確認された ($p < 0.05$)。検証セットでも 0.952 → 0.999 の改善が得られた。

有意差検定は、初期値の異なる 3 エージェントを対象に全ゲームで評価し、検証セットで 666 スコア (222×3)、テストセットで 1, 542 スコア (514×3) を得て実施した。表 6.1 中の ▲ は、有意差があると意味する。LeDeepChef については再実装を行わず、元論文 [21] の報告値を引用した。

平均ステップ数に関しては、CogniTextworldAgent 系のエージェント間で大きな差は見られなかった。ただしナビゲータを導入した場合には若干の減少が確認され、これは BFS ベースの経路探索により常に部屋間の最短経路を発見できるためと考えられる。

6.5 考察

6.5.1 テスト時にナビゲータを除去

結果から、ナビゲータの導入が大幅な性能改善をもたらすことが確認された。この改善は、ナビゲータが経路探索やアイテム追跡といった低レベルタスクを処理し、エージェントが高レベルの意思決定に専念できたためだと考えられる。一方で、ナビゲータへの依存が強すぎると、それを外した場合に性能が大きく低下するのではないかという懸念が残る。この点を検証するため、アイテムナビゲーションコマンドで学習したエージェントに対し、テスト時にナビゲータを無効化するアプリケーション実験を行った。結果は表 6.1 の「-Navigator(テスト時)」に示す。

CogniTextworldAgent + Navigator では、テスト集の正規化スコアが 0.937 から 0.890 に、検証集では 0.994 から 0.955 に低下した。また CogniTextworldAgent + Navigator + Action Classifier では、テスト集で 0.968 から 0.918、検証集で 0.999 から 0.965 に低下した。予想どおりナビゲータを外すと性能は下がったが、それでもベースラインと同等かそれ以上の性能は維持されていた。したがって、ナビゲータを組み込んで訓練

したエージェントは、ナビゲータが利用可能な場合に性能を大幅に高め、利用できない場合でも少なくとも従来水準の性能を確保できることが分かる。

6.5.2 大規模言語モデルの性能

我々は第5章で、テスト集からランダムに選んだ10ゲームにおいてgpt-4.1-miniの性能を検証した。本節ではさらにgpt-4.1を用いた実験を行う。使用したプロンプトを以下に示す。

```
Task: Find and examine the cookbook,  
follow the instructions to prepare  
the meal, and eat it.  
Executed action 23: take yellow potato  
Reward received: 3  
Room: Garden  
Inventory: a raw yellow potato,  
a fried red apple. Total 2 items.  
Recipe: Ingredients: red apple  
yellow potato Directions:  
fry the red apple  
fry the yellow potato prepare meal  
Available actions:  
1 drop red apple  
2 drop yellow potato  
3 go south  
Next action (answer with the action  
directly):
```

ここで、*Task*はゲーム目標の概要、*Executed action*は直前のコマンドであり、後の数字は累積ステップ数を示す。*Reward received*は累積スコア(タスク1つにつき1点)、*Room*はエージェントの現在地、*Inventory*は所持アイテムの一覧である。*Recipe*はクックブックの内容であり、「examine cookbook」を実行するまでは「missing」と表示される。最後に*Available actions*が候補行動リストである。このリストはゲームエンジンから取得され、学習高速化のためにフィルタリングされている。

ナビゲータの有効性を調べるために、我々は *Available actions* にナビゲーションコマンドを追加した。その他の部分は変更していない。例を以下に示す。

```
...
Available actions:
0 navigate to stove
1 navigate to knife
2 drop red apple
3 drop yellow potato
4 go south
...
```

アイテムナビゲーションコマンドは「`navigate to <item>`」というテンプレートにアイテム名を当てはめて生成し、ノイズ低減のためにフィルタリングを行った。LLMがこのコマンドを選択すると、ナビゲータがそれを移動コマンド列に変換して実行する。

実験は OpenAI Playground 上で `gpt-4.1-mini` および `gpt-4.1` を用いて行い、設定は以下のとおりである。

```
text.format: text, temp: 1.0,
tokens: 2048, top_p: 1.00, store: false
```

プロンプトは LLM に行動のみを出力するよう明示的に指示したが、応答は必ずしも期待形式に従わなかった。そのため、意図された行動を人手で抽出し、ゲームエンジンに入力した。また実験の負荷を抑えるため、LLM のステップ数上限を 50 に設定した。公平性を確保するため、訓練済みエージェントにも同じ上限を適用した。結果を表 6.2 に示す。

結果から、`gpt-4.1` は `gpt-4.1-mini` より明らかに高い性能を示し、より大きなモデルが FTWP タスクにおいてより優れた性能を発揮することが示唆された。これは推論能力の向上によるものと考えられる。さらに、ナビゲータを導入することで LLM の性能は一貫して向上した。実際のゲームプレイを分析すると、`gpt-4.1` と `gpt-4.1-mini` の両者が自発的にナビゲーションコマンドを利用していた。ナビゲーション支援がない場合、LLM は複雑なマップで移動ループに陥ることが多く、ステップを浪費してタスク未完了のまま上限に達することもあった。これに対し、ナビゲーションコマンド

Model	Norm. Score	Avg. Steps
GPT-4.1-mini	0.395	7.4
+ Navigator	0.563	11.8
GPT-4.1	0.625	15.6
+ Navigator	0.750	11.1
CogniTextworldAgent	0.868	15.0
+ Navigator	0.979	14.4
CogniTextworldAgent + Action Classifier	0.875	17.3
+ Navigator	0.979	14.8

表 6.2: ランダムに選んだ 10 個のテストゲームにおける LLM と学習済みエージェントの性能。

を利用できる場合は既知アイテムの所在部屋に安定して到達でき、大きな性能向上につながった。

訓練済みエージェントは一貫して LLM を上回り、FTWP 環境における特定スキルはゼロショット推論よりも学習によって獲得する方が適していることが示された。特にナビゲータの利用は訓練済みエージェントにとっても大きな改善をもたらし、本研究の主要な発見と一致している。

LLM が自発的にナビゲータを利用する例

以下に、GPT-4.1 + Navigator がナビゲータを自発的に利用して成功したゲームの一例を示す、紙幅の都合上、重要な部分のみを抜粋する。

...

Executed action 24: take yellow potato

Reward received: 3

Room: Garden

Inventory: a knife, a raw yellow potato, a fried red apple. Total 3 items.

Recipe: Ingredients: red apple yellow potato Directions: fry the red apple
fry the yellow potato prepare meal

Available actions:

0 navigate to BBQ

1 drop yellow potato

2 slice yellow potato with knife

3 slice red apple with knife
4 dice red apple with knife
5 chop red apple with knife
6 navigate to stove
7 go south
8 drop red apple
9 navigate to oven
10 dice yellow potato with knife
11 drop knife
12 chop yellow potato with knife

Next action (answer with the action directly):

> 6 navigate to stove

Executed action 25: navigate to stove

Reward received: 3

Room: Kitchen

Inventory: a knife, a fried red apple, a raw yellow potato. Total 3 items.

Recipe: Ingredients: red apple yellow potato Directions: fry the red apple
fry the yellow potato prepare meal

Available actions:

0 cook yellow potato with oven
1 cook red apple with stove
2 chop yellow potato with knife
3 go north
4 drop knife
5 dice red apple with knife
6 cook yellow potato with stove
7 dice yellow potato with knife
8 drop red apple
9 prepare meal

- 10 go east
- 11 chop red apple with knife
- 12 slice red apple with knife
- 13 drop yellow potato
- 14 navigate to BBQ
- 15 slice yellow potato with knife
- 16 cook red apple with oven

Next action (answer with the action directly):

> 6 cook yellow potato with stove

Executed action 6: cook yellow potato with stove

Executed action 30: cook yellow potato with stove

Reward received: 4

Room: Kitchen

Inventory: a fried yellow potato, a knife, a fried red apple. Total 3 items.

Recipe: Ingredients: red apple yellow potato Directions: fry the red apple
fry the yellow potato prepare meal

Available actions:

- 0 drop yellow potato
- 1 slice red apple with knife
- 2 cook red apple with oven
- 3 dice yellow potato with knife
- 4 dice red apple with knife
- 5 drop red apple
- 6 slice yellow potato with knife
- 7 drop knife
- 8 chop red apple with knife
- 9 cook yellow potato with oven
- 10 chop yellow potato with knife

11 prepare meal
12 go north
13 cook red apple with stove
14 cook yellow potato with stove
15 navigate to BBQ
16 go east

Next action (answer with the action directly):

> 11 prepare meal

...

ハイライトで示した部分に注目する。 「Executed action 25: navigate to stove」 で「navigate to stove」が正しく実行されたことがわかる。次のプロンプトを観察すると、最初の行には「Executed action 30: cook yellow potato with stove」と表示されている。ここでコマンド番号が 25 から 30 に変化していることが確認できる。これは、「navigate to stove」が実際には 5 つの移動指令に変換されて実行されたことを意味する。

ナビゲーション指令が存在しない場合、LLM が同じ目的を達成するためには、部屋の空間配置および stove の位置を正確に把握する必要がある。さらに、移動の過程において常に自らの目標を維持し、他の物体に注意を逸らさないことが求められる。この例は、navigation 指令がエージェントにとって大きな助けとなることを明確に示している。同一のゲームにおいて、ナビゲーターを利用した GPT-4.1 は 32 ステップでゲームをクリアし、満点である 6 点を獲得した。一方、ナビゲーターを利用できなかった GPT-4.1 は行動上限が 50 ステップを超えてタスクを全部達成できず、最終得点は 2 点にとどまった。

6.6 結言

本研究は、公開されている中で最大のテキストゲームデータセットである FTWPにおいて、エージェント性能を改善する新しいアプローチを提示した。ナビゲータの導入より、従来の行動模倣手法が抱える重要な制約—曖昧な行動による性能低下—を克服するため、提案手法は FTWPにおいて最先端の性能を達成した。我々の最良のエー

ジェント, CogniTextworldAgent + Action Classifier + Navigator はテストセットで 0.968 のタスク完成率を達成し, 従来研究の 0.880 を大きく上回った. 検証セットでも一貫した改善が見られ (0.999 vs. 0.952), 提案手法の安定性が確認された. これらの結果は, 曖昧な行動を明示的なナビゲーションコマンドに置き換えることで, 訓練効率を高めるだけでなく, テキストゲームエージェントの性能向上に直結することを示している.

第7章 結論

7.1 研究の総括

テキストゲームは自然言語処理と強化学習の交差領域である。本論文では、二種類の文字ゲームエージェントの実装方法を提案し、二つの公開データセット上でそれらを評価した。本節では、本研究の主要な成果をまとめる。

第3章では、TextWorld Commonsense(TWC) データセットに対して LLM に基づくエージェント実装方法を提案した。この実装方法では LLM をファインチューニングする必要はなく、精巧に設計されたプロンプトを用いて LLM が合理的な行動を生成するように導く。プロンプトには、タスク記述、現在の部屋の記述、候補行動リストなどの必要情報が含まれ、さらに、行動履歴を逐次更新することでエージェントの記憶を保持する。さらに、我々は近年のプロンプトエンジニアリング分野の進展に注目し、プロンプトに多様な技術を導入した。これらの技術には Chain-of-Thought, few-shot プロンプト、出力テンプレートなどが含まれる。さらに、本研究では「フィードバック強化」と呼ぶ新たなプロンプト技術を提案した。これはゲームエンジンからのフィードバックを明示化することで、LLM が行動結果をより良く理解できるようにするものである。実験結果およびアブレーション実験の結果、フィードバック強化技術が LLM エージェントの性能を顕著に向上させることができた。

TWC より複雑で大規模なテキストゲーム環境に対しては、LLM ベースのエージェントはコスト面や性能面で満足のいくものになりにくい。こうした環境に対応するためには、LLM より小規模な言語モデルをローカルで訓練することを考える必要がある。では、どのような言語モデルを訓練対象として選ぶべきか？第4章では、テキストゲームエージェントの実装に最も適したモデルを選定するために、いくつかの代表的な言語モデルの性能を評価した。テキストゲームエージェントにとって言語理解は不可欠な能力であるため、モデルの言語理解能力を十分に反映できるタスク—「自動強調付与」を評価基準にした。具体的には、日本語文章を対象とした自動強調付与タスクにおいて、fastText[3]、BERT[4]、RoBERTa[2]、CRF[5] といった言語モデルを比

較評価した。その結果、transformer ベースのモデル (BERT および RoBERTa) は他のモデルを大きく上回る性能を示し、さらに transformer モデル間においても RoBERTa が常に優れた結果を示した。この知見に基づき、我々は第 4 章以後の研究において一貫して RoBERTa を採用することとした。

第 5 章は、より複雑な文字ゲームデータセット First TextWorld Problem(FTWP) に注目した。これは現時点での公開されている最大規模の文字ゲームデータセットである。我々は、このデータセット上で最も高性能とされる CogntextworldAgent を RoBERTa を用いて再実装し、その検証セットにおける挙動を分析した。分析の結果、エージェントは誤り行動の実行によってゲームが早期終了することが多いと判明した。この問題を解決するために、我々は RoBERTa に基づく誤り行動分類器を提案し、クリックブックの指示と矛盾する行動を識別できるようにした。この分類器を CogntextworldAgent に統合した結果、テストセットにおける正規化スコアは 0.880 から 0.912 に向上した。

第 6 章では、さらに FTWP エージェントの性能向上を試みた。ゲームマップが複雑であるため、FTWP におけるナビゲーションタスクは困難である。一方で、我々は模倣学習に基づく学習方法が、移動コマンドをそのまま模倣することで性能を低下させる可能性があることを指摘した。同じ移動コマンドでも全く異なる目的地に至るために、因果的混同 (causal confusion) と呼ばれる問題が生じるからである。この問題を解決するために、我々は移動コマンドをより明確なナビゲーション指令に置き換えることを提案した。これを実行するために、FTWP 環境の拡張として BFS に基づくナビゲータを設計した。修正済みの訓練セットで学習したエージェントはナビゲーションコマンドの使用を習得し、テストセットにおけるタスク達成率は 0.968 まで向上した。これは FTWP データセットにおける最先端の成績である。

以上の成果は、テキストゲームエージェントを実装するための方法論を構成する。第一に、小規模環境に対しては、LLM の強力な推論能力を利用し、精巧に設計されたプロンプトによって合理的な行動を導くことができる。この方法は極めて簡便で将来性が高く、テキストゲームエージェント実装の第一選択肢となり得る。第二に、大規模かつ複雑な環境に対しては、LLM のコストが大きい場合や推論能力が複雑なタスクに十分でない場合、事前学習済みの transformer モデルに基づき、模倣学習による訓練を行うことができる。このようなエージェントは拡張性に優れ、多様なモジュールを統合することで性能をさらに向上させることができる。

7.2 今後の課題

今後の研究では、以下の課題に取り組みたい。

第一に、本研究で提案した CogniTextworldAgent + Action Classifier + Navigator は、専門家の行動を模倣して訓練されていて、FTWP データセット上において最先端の性能を示した。この成果は、FTWP が豊富な専門家行動データを含んでいることと大きく関係している。しかしながら、提案手法は、専門家行動データが乏しい、あるいは全く存在しないゲーム環境には対応できないという制約を抱えている。これは提案手法の応用範囲を大きく制限する要因となっている。一つの解決策として、人手による専門家行動データの作成が考えられるが、そのコストは無視できない。したがって、今後の研究では、専門家行動データに依存しない強化学習のアルゴリズム（例えば Q 学習、SARSA など）を提案手法に導入することを検討する。専門家行動データが存在しない状況でも、FTWP 環境において現在の最良エージェントに匹敵する性能を達成できるエージェントの実現を目指す。

第二に、本研究では二種類のテキストゲームエージェントを提案し、それぞれのデータセットで強力な性能を確認した。しかし、これらのエージェントはテキストゲームに特化しており、Atari や MineCraft といったビジュンゲーム、さらには現実世界のタスクには直接適用できないという制限がある。この課題を克服することは、将来的な研究における重要な方向性である。我々は、テキスト理解能力を備えたエージェントが言語をツールとして活用することで、より高度な推論や計画を可能にする点に大きな優位性があると考えている。したがって、本研究の成果はテキストゲームにとどまらず、より複雑かつ実用的な環境においても価値を発揮できると期待される。

謝辞

本研究を進めるにあたり、研究の進め方や本論文の書き方など、細部にわたる御指導を頂きました、鳥取大学大学院工学研究科情報エレクトロニクス専攻自然言語処理研究室の村田真樹教授に心から御礼申し上げます。また、本研究を進めるにあたり、御指導、御助言を頂きました、村上仁一准教授に心から御礼申し上げます。その他様々な場面で御助言を頂いた自然言語処理研究室の皆様に感謝の意を表します。

参考文献

- [1] Yunqiu Xu, Meng Fang, Ling Chen, Yali Du, Joey Tianyi Zhou, and Chengqi Zhang. Deep reinforcement learning with stacked hierarchical attention for text-based games. *Advances in Neural Information Processing Systems*, Vol. 33, pp. 16495–16507, 2020.
- [2] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.
- [3] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, Vol. 5, pp. 135–146, 2017.
- [4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (Long and short papers)*, pp. 4171–4186, 2019.
- [5] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.
- [6] Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Pushkar Shukla, Sadhana Kumaravel, Gerald Tesauro, Kartik Talamadupula, Mrinmaya Sachan, and Murray Campbell. Text-based rl agents with commonsense knowledge: New challenges, environments and baselines. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35, pp. 9018–9027, 2021.

- [7] Pedro Lima. First textworld challenge — winning solution notes, 2019.
- [8] Matan Haroush, Tom Zahavy, Daniel J. Mankowitz, and Shie Mannor. Learning how not to act in text-based games. In *6th International Conference on Learning Representations, ICLR 2018*, 2018. Publisher Copyright: © 6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings. All rights reserved.; 6th International Conference on Learning Representations, ICLR 2018 ; Conference date: 30-04-2018 Through 03-05-2018.
- [9] Shunyu Yao, Karthik Narasimhan, and Matthew Hausknecht. Reading and acting while blindfolded: The need for semantics in text game agents. *arXiv preprint arXiv:2103.13552*, 2021.
- [10] Chen Feng Tsai, Xiaochen Zhou, Sierra S Liu, Jing Li, Mo Yu, and Hongyuan Mei. Can large language models play text games well? current state-of-the-art and open questions. *arXiv preprint arXiv:2304.02868*, 2023.
- [11] Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. Textworld: A learning environment for text-based games. In *Computer Games: 7th Workshop, CGW 2018, Held in Conjunction with the 27th International Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, July 13, 2018, Revised Selected Papers 7*, pp. 41–75. Springer, 2019.
- [12] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, Vol. 35, pp. 24824–24837, 2022.
- [13] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, Vol. 33, pp. 1877–1901, 2020.
- [14] Joon Sung Park, Joseph O’Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of

- human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–22, 2023.
- [15] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, Vol. 47, No. 2, pp. 235–256, 2002.
 - [16] Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? *arXiv preprint arXiv:2203.07540*, 2022.
 - [17] Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. Language understanding for text-based games using deep reinforcement learning. *arXiv preprint arXiv:1506.08941*, 2015.
 - [18] Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. Deep reinforcement learning with a natural language action space. *arXiv preprint arXiv:1511.04636*, 2015.
 - [19] Prithviraj Ammanabrolu and Matthew Hausknecht. Graph constrained reinforcement learning for natural language action spaces. *arXiv preprint arXiv:2001.08837*, 2020.
 - [20] Keerthiram Murugesan, Subhajit Chaudhury, and Kartik Talamadupula. Eye of the beholder: Improved relation generalization for text-based reinforcement learning agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36, pp. 11094–11102, 2022.
 - [21] Leonard Adolphs and Thomas Hofmann. Leddeepchef deep reinforcement learning agent for families of text-based games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34, pp. 7342–7349, 2020.
 - [22] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

- [23] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, No. 5, pp. 834–846, 2012.
- [24] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.
- [25] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.
- [26] Shengyi Huang and Santiago Ontañón. A closer look at invalid action masking in policy gradient algorithms. *arXiv preprint arXiv:2006.14171*, 2020.
- [27] Pim De Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. *Advances in neural information processing systems*, Vol. 32, , 2019.
- [28] Adi Botea, Bruno Bouzy, Michael Buro, Christian Bauckhage, and Dana Nau. Pathfinding in games, 2013.
- [29] Abdul Rafiq, Tuty Asmawaty Abdul Kadir, and Siti Normaziah Ihsan. Pathfinding algorithms in game development. In *IOP Conference Series: Materials Science and Engineering*, Vol. 769, p. 012021. IOP Publishing, 2020.
- [30] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023. URL <https://arxiv.org/abs/2305.16291>, 2023.
- [31] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.
- [32] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al.

- Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.
- [33] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [34] Amirreza Shirani, Franck Dernoncourt, Nedim Lipka, Paul Asente, Jose Echevarria, and Thamar Solorio. Semeval-2020 task 10: Emphasis selection for written text in visual media. *arXiv preprint arXiv:2008.03274*, 2020.
- [35] Masaki Murata and Yuki Abe. Using machine learning for automatic estimation of emphases in japanese documents. *IEICE TRANSACTIONS on Information and Systems*, Vol. 100, No. 10, pp. 2669–2672, 2017.
- [36] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [37] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.
- [38] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.
- [39] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- [40] Shashi Narayan, Nikos Papasarantopoulos, Shay B Cohen, and Mirella Lapata. Neural extractive summarization with side information. *arXiv preprint arXiv:1704.04530*, 2017.
- [41] Ahmed Abdul-Hamid and Kareem Darwish. Simplified feature set for arabic named entity recognition. In *Proceedings of the 2010 named entities workshop*, pp. 110–115, 2010.

- [42] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.
- [43] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.
- [44] Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. Portuguese named entity recognition using bert-crf. *arXiv preprint arXiv:1909.10649*, 2019.
- [45] Dawid Jurkiewicz, Łukasz Borchmann, Izabela Kosmala, and Filip Graliński. Aplicaai at semeval-2020 task 11: On roberta-crf, span cls and whether self-training helps them. *arXiv preprint arXiv:2005.07934*, 2020.
- [46] Taja Kuzman, Igor Mozetic, and Nikola Ljubešic. Chatgpt: Beginning of an end of manual linguistic data annotation? use case of automatic genre identification. *ArXiv, abs/2303.03953*, 2023.
- [47] Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J Mankowitz, and Shie Mannor. Learn what not to learn: Action elimination with deep reinforcement learning. *Advances in neural information processing systems*, Vol. 31, , 2018.
- [48] Felipe Codevilla, Eder Santana, Antonio M López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9329–9338, 2019.
- [49] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

付 錄 A 研究業績

A.1 学術雑誌論文

	著者・論文題目・発表機関	本文 第几章
01	ZHUO, Binggang, Ryota HONDA, and Masaki MURATA. "Information for Transformer-based Japanese Document Emphasis." IEICE Transactions on Information and Systems (2025), vol.E108-D, no.7, pp.808-819	第4章
01	ZHUO, Binggang, and Masaki MURATA. "Enhancing Text Game Agent Performance via Navigator." IEICE Transactions on Information and Systems, 投稿中.	第6章

A.2 国際会議論文

	著者・論文題目・発表機関	本文 第几章
01	Binggang Zhuo, and Masaki Murata. "Utilizing GPT-4 to Solve TextWorld Commonsense Games Efficiently." Proceedings of the 10th Workshop on Games and Natural Language Processing@ LREC-COLING, vol.2024. pp.76-84	第3章
02	Binggang Zhuo, and Masaki Murata. "Enhancing Text Game Agent Performance via Wrong Action Classification." Proceedings of the 2025 9th International Conference on Natural Language Processing and Information Retrieval, 掲載決定.	第5章

A.3 学会研究会論文

	著者・論文題目・発表機関	本文
01	Binggang Zhuo, 本田涼太, 村田真樹. “マスク言語モデルによる日本語文章の自動強調付与.” 研究報告情報基礎とアクセス技術 (IFAT) (2023), vol.7, pp.1-6.	第4章
02	Binggang Zhuo, Masaki Murata. “インターラクティブフィクションにおける大規模言語モデルの性能.” 言語処理学会第30回年次大会発表論文集, vol.2024, pp.314-319.	第3章