記号処理アーキテクチャを大規模言語モデルで再現するための方法論: 自然言語とプログラミング言語のプロンプトとしての比較

Integrating Symbolic Processing Architectures and LLM: Comparison of Natural and Programming Language Prompts

長原 令旺 [1*]　飯田 愛結 [1]　奥岡 耕平 [1]　大森 隆司 [1]　大澤 正彦 [1]
Reo Nagahara[1], Ayu Iida[1], Kohei Okuoka[1], Takashi Omori[1], Masahiko Osawa[1]

[1] 日本大学
[1] Nihon University

Abstract: 既存研究では，認知アーキテクチャ(CA) と大規模言語モデル (LLM) を統合した 2 つの手法，CA Embedded in LLM(CEL) と LLM Embedded in CA(LEC) を提案している．これらの手法は ChatGPT に自然言語を用いたプロンプトを与えて認知アーキテクチャとして機能させている．しかし当該研究は初歩的な検討であり，多面的で詳細な追加検証が必要である．本研究は，記号処理アーキテクチャを説明するプロンプトをプログラミング言語を用いて与える方法を提案し，自然言語を用いた場合と結果を比較する．

1　はじめに

心的状態の表現に関しては，人間の思考や感情を模倣するために，様々なモデルが提案されている [1,2]. これらの研究では言外の意味を扱うコミュニケーションにおいて一定の精度を示している．一方で，大規模言語モデルの発展により，機械が人間の言葉を理解できるようになってきている．しかし，大規模言語モデルは言外の意味を踏まえたコミュニケーションが苦手であることが示されている [3,4].

著者らの一部は，記号処理アーキテクチャと大規模言語モデルを統合する手法を提案している [5]．この手法は，記号処理アーキテクチャやそのモジュールを自然言語で説明することで，大規模言語モデルとの統合を可能にしている．実験では，記号処理アーキテクチャとして BDI（Belief-Desire-Intention）モデル [6] をベースとした人間の心的状態に関するものを利用することで，言外の意味を読み取るタスクの性能が向上することを示唆した．しかし，[5] は初歩的な検討であり，多面的で詳細な追加検証が必要である．

そこで本研究は，記号処理アーキテクチャを説明するプロンプトの適した形式を検討することを目的とする．プログラミング言語として記号処理アーキテクチャを表現することでプロンプトのフォーマットが統制され, 解釈の一貫性を保証できる可能性がある．そこで，従来の自然言語ベースのアプローチとプログラミング言語ベースのアプローチのどちらが効果的であるかを検証する．

本研究の貢献は以下の 2 点にまとめられる．

- 自然言語によるプロンプトとプログラミング言語によるプロンプトを比較して，プログラミング言語の方が優れている場合があることを発見したこと．
- 著者らによる既存研究 [5] で提案した統合手法に関して，既存研究の結果を支持する共通の結果を確認したこと．

2　背景

2.1　大規模言語モデルの課題

数十億から数兆のパラメータを持つ大規模な深層学習言語モデルは，膨大なテキストデータを使って訓練され，自然言語処理の多くのタスクで人間レベルまたはそれ以上の精度を達成している．これらのモデルを用いた ChatGPT は，多様なトピックに対して自然言語で応答できる能力を持っている．しかし，これらのモデルには訓練データのバイアス，常識的な判断の誤り，最新情報や未知のトピックに対する不完全な回答，高い計算コストなどの課題も存在する．

また，言外の意味を扱うコミュニケーションタスクに関して，大規模言語モデルはまだ十分な性能を達成していない [3,4]．このようなコミュニケーションは，言

*連絡先：日本大学文理学部
〒 156-8550 東京都世田谷区桜上水 3-25-40
E-mail: osawa.masahiko@nihon-u.ac.jp

葉だけでなく話者の情報や文脈を含む複雑なものであり，大規模言語モデルが完璧に理解するには限界がある．言語論の分野では，このような皮肉や比喩を含むコミュニケーションが研究されている [7]．いくつかの大規模言語モデルを用いた言語論タスクの評価では，人間と同等の正答率を示すタスクも存在したが [8]，ユーモアや皮肉などのタスクは困難とされている．また，他者の心的状態を推定する心の理論に関連するタスクでも，一部では人間と同等の性能を示すものの，困難なタスクも存在する [9]．

2.2 他者モデルの有効性と言外の意味理解

一方で，他者モデルは特定のタスクにおいて相手の心的状態や行動を予測するモデルである．例えば，横山らの研究 [10] では，ハンタータスクにおいて，エージェントが互いに意図を推測し，行動を予測する他者モデルを用いることで，タスクの解決が容易になることを示している．また，阿部らの研究 [11] では，子供との遊び過程でロボットに他者モデルを搭載し，子供の心的状態を推定してロボットの行動を変化させる実験を行った．このように，他者モデルは，言外の意味を理解するという面で有効である．

2.3 大規模言語モデルと記号処理アーキテクチャの統合

著者らは大規模言語モデルにおける言外の意味理解に関する課題を克服するため，記号処理アーキテクチャと大規模言語モデルの組み合わせを通じて，改善の可能性を検証している [5]．この研究では，発話の意図を考慮した対話生成アーキテクチャ(図 1) を設計した．このアーキテクチャは，自己および他者それぞれの「信念 (Belief)」「願望 (Desire)」「意図 (Intention)」の六つの内部状態と，「他者の意図の推定」「自己の意図の形成」「自己の発話の生成」の 3 つのモジュールを含む構造を有している．

[5] では，記号処理アーキテクチャと大規模言語モデルを統合させるための 2 つの基本的なアプローチが提案されている．1 つ目は，記号処理アーキテクチャをプログラムとして具現化し，その内部モジュールを大規模言語モデルで実現する手法である．2 つ目は，記号処理アーキテクチャ及びその構成要素と挙動を詳細に記述したプロンプトを作成し，これを大規模言語モデルへ入力する手法である．これらのアプローチを比較した結果，前者は記号処理アーキテクチャの中に大規模言語モデルを組み込んでおり，後者は記号処理アーキテクチャを大規模言語モデルに組み込んであると理解できる．よってそれぞれを「LLM Embedded in Cognitive Architecture (LEC)」及び「Cognitive Architecture Embedded in LLM (CEL)」と名付けた．実験は，3 つの異なるシチュエーションに対して，4 種類のアプローチを用いて各 10 回ずつ，合計 120 回実施した．これらのシチュエーションは，言外の意味を持つ典型的な例として「皮肉」「ツンデレ」「社会的制約」と名付け，各シチュエーションでは自己及び他者の信念や願望，さらには他者の発話を定義することで構築した．実験条件として，提案された「LEC（記号処理アーキテクチャ内統合大規模言語モデル)」と「CEL（大規模言語モデル内統合記号処理アーキテクチャ)」の 2 つの手法に加え，「LLM（基本的な大規模言語モデルの振る舞いを模倣するための単純なプロンプトのみを提供)」と「LLM with BD（提案手法に等しい内部状態を備えた大規模言語モデル)」という，計 4 つの条件である．それぞれのアプローチが，言外の意味を考慮した出力を生成すれば成功とみなし，字義的な意味のみを反映した出力であれば失敗と定義した．

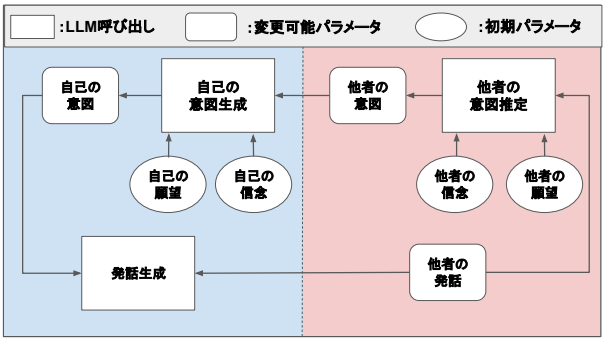

図 1: 対話アーキテクチャ

結果として，一般的な LLM 条件では，どのシチュエーションにおいても言外の意味を理解する発話は生成されず，成功率は 0 %だった．これは，設定したシチュエーションが言外の意味を捉える場面に対する一般的な大規模言語モデルの限界を示している．一方，内部状態として信念や願望を与えた「LLM with BD」条件では，「ツンデレ」と「社会的制約」のシチュエーションで高い成功率を記録したが，これらのシチュエーションではアーキテクチャの必要性が低かったため，アーキテクチャの有効性を検証するには適していなかった．特に「皮肉」のシチュエーションでは，LEC 条件で成功率が大幅に向上し，統合アーキテクチャの効果が示唆されたが，CEL 条件では成功率が 40%に止まった．

3　プログラミング言語ベースのプロンプト設計

本研究では，プログラミング言語で表現された記号処理アーキテクチャを LEC と CEL の手法で大規模言語モデルと統合する．実験で用いるアーキテクチャは [5] と同様である（図 1）．

3.1　プログラミング言語と自然言語のプロンプトの特性

プログラミング言語と自然言語のプロンプトが持つ特性について検討する．プログラミング言語プロンプトの特性には，指示が明確に構造化され，具体的な操作指令を提供する能力が含まれる．具体例として，クラスやメソッドの定義，引数の受け渡し方法など，処理の流れをコードを通じて明確に指示することが挙げられる．これにより，システムに対して実行すべき明確なステップを指示することが可能となる．対照的に，自然言語プロンプトはプログラミング言語のように高度に構造化されていない場合が多く，そのため内部処理過程に曖昧さを内包する．このような自然言語の曖昧性は，大規模言語モデルにプロンプトとして提供された際，解釈の一貫性を保証することが困難となり，結果の不確実性が高まる可能性がある．

3.2　提案システムの設計

言外の意図を推測し発話を生成するシステムの設計について提案する．具体的なプロンプトの例を Listing1 に示す．プログラムの docstring による説明を通じて，システムの各モジュールの機能と役割を明確に示す．本システムは，対話システム内の 3 つのモジュール—意図推定システム，意図生成システム，および発話生成システム—を構成し，それぞれが特定の入力を基に処理を行い，次のモジュールへの入力を生成する．以下は各システムの概要である．

意図推定システム　他者の信念，願望，および発話から，矛盾や違和感のない他者の意図を推定する．

意図生成システム　自己の信念，願望，および推定された他者の意図を踏まえ，自己の意図を生成する．このシステムは，自身の内部状態と他者の意図をふまえて，自己の行動指針を形成する．

発話生成システム　自己の意図と他者の発話を考慮して，矛盾や違和感のない自己の発話を生成する．このシステムは，自己の意図に基づき，適切な発話を構築し，コミュニケーションを促進する．

システムの内部表現は，プログラムの最後にパラメータとして与え，各システムにプログラムとして記載されていない情報は docstring を参照して GPT-4 が補完する．このシステムは，言外の意図を正確に捉え，それに応じた発話を生成することを目的としており，対話の質を向上させることが期待できる．CEL の場合は下記のソースコードをそのまま与え，LEC の場合はモジュールに対応するメソッドのみを記載する．

4　自然言語とプログラミング言語のプロンプトとしての性能比較

本章では，自然言語とプログラミング言語のプロンプトとしての比較を目的とし，発話と発話意図に乖離がある対話を例に性能の評価を行う．

4.1　シチュエーション

シチュエーションは [5] と同様なものを利用した．[5] では，発話とその背後にある意図との間に乖離が生じる具体的な対話シチュエーションを設定し，それらを「皮肉」「ツンデレ」「社会的制約」とした.「皮肉」のシチュエーションでは，話題が時計についてであるにも関わらず，実際には時間がかなり経過していることを暗に示す意図を持っている.「ツンデレ」のシチュエーションは，話者が自分の行為の対象に対して，ほかの予定がある日にその予定を優先しても構わないと口では言うが，実際にはその人ともっと時間を過ごしたいと願っている状況である.「社会的制約」のシチュエーションでは，上司がパワーハラスメントと誤解されないように注意深くコミュニケーションを取りながら，部下の健康を気遣いつつ，仕事を続けてほしいという複雑な意図を持つ状況を表している．分析を容易にするために，この実験では他者の発話に対する自己の発話を生成する場面に限定し，実験の途中での初期値や入力の更新は行わない．

4.2　比較条件

本実験では，合計 6 つの異なる条件を対象に比較分析を行う．最初に，GPT-4 を利用し，純粋に自然言語のみを用いてプロンプトを与えた場合の 2 つの条件，すなわち自然言語を基にした LEC と CEL の条件を，それぞれ N-LEC（NaturalLanguageLEC）および N-CEL（NaturalLanguagCEL）として設定した．次に，プログラミング言語によるプロンプトを与えた場合の 2 つの条件，P-LEC（ProgramLanguageLEC）および P-CEL（ProgramLanguageCEL）を設定した．

Listing 1: プログラミング言語ベースのプロンプト

```
以下のプログラムはシステムの流れです．プログラムを基に自己の発話を生成してください．システムの内部表現はプロンプトの最後にパラメータとして与えます．各システムに書かれてないプログラムはあなたが補完してください．指示のない文章は一切出力しないでください．

class DialogueSystem:

  def __init__(self):
    # システムの内部表現
    self.belief_of_self = [] # 自己の信念
    self.belief_of_other = [] # 他者の信念
    self.desires_of_self = [] # 自己の願望
    self.desire_of_other = [] # 他者の願望
    self.intention_of_self = "" # 自己の意図
    self.intention_of_other = "" # 他者の意図

  def intention_estimation_system(self, belief_of_other, desire_of_other, utterance_of_other):
    '''意図推定システムについて説明します．意図推定システムは,「他者の意図
    (intention_of_other)」を推定するシステムです．入力として与えられた,「他者の信念
        (belief_of_other)」「他者の願望 (desire_of_other)」「他者の発話 (utterance_of_other)」から，矛盾や違和感のない「他者の意図 (intention_of_other)」を「推定」してください
    '''

  def intention_generation_system(self, belief_of_self, desires_of_self, intention_of_other):
    '''意図生成システムについて説明します．意図生成システムは,「自己の意図
    (intention_of_self)」を生成するシステムです．入力として与えられた,「自己の信念
        (belief_of_self)」「自己の願望 (desire_of_self)」「他者の意図 (intention_of_other)」から，矛盾や違和感のない「自己の意図 (intention_of_self)」を「生成」してください．
    '''

  def utterance_generation_system(self, intention_of_self, utterance_of_other):
    '''発話生成システムについて説明します．発話生成システムは,「自己の発話
    (utterance_of_self)」を生成するシステムです．入力として与えられた,「自己の意図
        (intention_of_self)」「他者の発話 (utterance_of_other)」から，矛盾や違和感のない「自己の発話 (utterance_of_self)」を「生成」してください
    '''

  def process_input(self, belief_of_other, desire_of_other, utterance_of_other):
    # 入力を受け取り、つのシステムを順に起動させる 3
    intention_of_other = self.intention_estimation_system(belief_of_other, desire_of_other,
        utterance_of_other)
    intention_of_self = self.intention_generation_system(self.belief_of_self, self.
        desires_of_self, intention_of_other)
    utterance_of_self = self.utterance_generation_system(intention_of_self, utterance_of_other)

    # 各システムの入出力を出力
    print(f他者の意図": {intention_of_other}")
    print(f自己の意図": {intention_of_self}")
    print(f自己の発話": {utterance_of_self}")

# インスタンス化と入力処理の実行
dialogue_system = DialogueSystem()
dialogue_system.process_input(belief_of_other,desire_of_other,utterance_of_other)
```

4.3 実験手順

実験は設定した各条件下で 3 つの異なるシチュエーションに基づく対話をそれぞれ 10 回行い，生成された各応答を成功または失敗として評価した．ここで成功とは，対話相手の発話内容について言外の意味をとらえる語句やフレーズが応答に含まれる場合を指す．また，失敗とは，応答が字義通りの意味にのみ基づく語句やフレーズで構成されているか，または提供されたプロンプトの指示に従わない場合を指す．評価は主に第一著者が規定された基準に従って行い，判断が難しい場合は著者間の協議により決定した．

4.4 実験結果

各シチュエーションにおける各設定条件下でのシステムの成功率を表 1 に示す．

N-LEC 条件では，皮肉と社会的制約シチュエーションで 90%以上の高い成功率を達成したが，ツンデレシチュエーションでは 60%/40%/40%と低い結果となった．N-CEL 条件も N-LEC 条件と同様に，皮肉と社会的制約シチュエーションで高い成功率を示したが，ツンデレシチュエーションでは低い成功率を示した．しかし，社会的制約シチュエーションの意図推定は N-LEC 条件と異なり低水準を示した．P-LEC 条件では，すべてのシチュエーションで意図生成と発話生成において 100%の成功率を達成したが，ツンデレと社会的制約シチュエーションにおける意図推定は 60%となった．P-CEL 条件では，ツンデレシチュエーションにおける意図推定が今回新たに検証した条件の中で高水準を示した．

表 1: 実験結果

皮肉	意図推定	意図生成	発話生成
N-LEC	100%	100%	100%
N-CEL	80%	80%	80%
P-LEC	100%	100%	100%
P-CEL	100%	100%	70%
ツンデレ	意図推定	意図生成	発話生成
N-LEC	60%	40%	40%
N-CEL	40%	40%	50%
P-LEC	60%	100%	100%
P-CEL	100%	60%	50%
社会的制約	意図推定	意図生成	発話生成
N-LEC	90%	90%	90%
N-CEL	50%	70%	100%
P-LEC	60%	100%	100%
P-CEL	90%	90%	90%

5 考察

5.1 プログラミング言語と自然言語の意図推定

自然言語ベースのプロンプトと比較して，プログラミング言語を組み合わせたプロンプトは，3 つのシチュエーションで高い成功率を達成している．この結果は，LLM が対話生成において人間の心的状態や複雑な社会的文脈をより正確に理解し模倣するためには，プログラミング言語によるプロンプトが有効なアプローチの一つであることを示唆している．

5.2 既存研究 [5] との比較

ツンデレシチュエーションにおける P-LEC と P-CEL の処理能力に関する比較を行うと，意図生成の段階において顕著な性能差がみられることがわかる．P-LEC では，3 つのモジュールが独立して機能し，プログラムによる明確な変数の受け渡しを介して連携することで，高度に統合された処理が可能になる．意図生成においては，他者の発話の字義的な解釈に左右されることなく，内部状態や意図に基づいた応答が生成されるため，各シチュエーションにおいて意図生成の成功率が 100%になっている．一方で P-CEL では，認知アーキテクチャの各モジュールが一つのプロンプト内で統合され，このプロンプトには他者の発話も含まれる．他者の発話の字義的な意味が全体の処理に影響を与える可能性があり，言外の意味を読み取る必要があるシチュエーションにおいて，意図生成の成功率が低下する原因となっていると考えられる．つまり，P-CEL では他者の発話に引っ張られることで，実際には異なる意図を持つ発話の背後にある意図を適切に推定し，反映することが難しくなっている．この結果は [5] の結果と同様であり，プログラミング言語によるプロンプトを利用する場合，他者の発話から言外の意味を読み取るタスクにおいては，モジュール間の独立した処理と明確な情報の受け渡しが重要であることが示唆される．

既存研究 [5] では，本実験で成功率が高かった皮肉シチュエーションでの成功率が低かった．一方，本研究ではツンデレシチュエーションにおいて多くの手法で成功率が低水準にとどまった．この差異が生まれる要因は，既存研究 [5] と本研究の差異である GPT のバージョンに起因するものと考えられる．GPT のバージョンによってシチュエーションの得手不得手があるとし

ても，提案した統合手法による成功率向上が認められることは，既存研究 [5] の再現性を裏付ける結果と解釈できる．

6 おわりに

本研究では，大規模言語モデルを活用した自然言語処理において，特に GPT-4 を用いて言外の意味を理解するためのプロンプト形式の検討を行った．プログラミング言語を用いた手法の有効性を検証し，人間の心的状態をより正確に理解し応答するための手法の改善を目指した．性能比較の方法論では，自然言語とプログラミング言語を組み合わせたを用いた実験を行い，異なるシチュエーションにおける言外の意図をどの程度理解し反映できるかを評価した．結果は，プログラミング言語を組み合わせたプロンプトが，特定の条件下で自然言語のみを用いたプロンプトよりも高い成功率を達成していることを示した．

参考文献

[1] Koosha et al. Khalvati. Modeling other minds: Bayesian inference explains human choices in group decision-making. ScienceAdvances,Vol. 5, p. eaax8783, 2019.

[2] Anand S Rao, Michael P Georgeff, et al. Bdi agents: from theory to practice. In Icmas, Vol. 95, pp. 312–319, 1995.

[3] Hu, J., et al.: A fine-grained comparison of pragmatic lan- guage understanding in humans and language models, in ACL, pp. 4194–4213 (2023)

[4] Mahowald, K., et al.: Dissociating language and thought in large language models: a cognitive perspective (2023)

[5] 飯田 愛結，阿部 将樹，奥岡 耕平, 福田 聡子, 大森 隆司, 中島 亮一，大澤 正彦：意図を読む AI の実現に向けて：対話型生成 AI と他者モデルの統合を例に, HAIS (2024)

[6] Michael Bratman. Intention, Plans, and Practical Reason. Cambridge, MA: Harvard University Press, Cambridge, 1987.

[7] Grice, H. P.: Logic and conversation, Brill (1975)

[8] Kosinski, M.: Evaluating large language models' ability to understand metaphor and sarcasm using a screening test for Asperger syndrome, arXiv preprint arXiv:2302.02083 (2023)

[9] Ullman, T.: Large Language Models Fail on Trivial Al- terations to Theory-of-Mind Tasks (2023)

[10] 横山絢美, 大森隆司. 協調課題における意図推定に基づく行動決定過程のモデル的解析. 電子情報通信学会論文誌 A, Vol. 92, No. 11, pp. 734–742, 2009.

[11] 阿部香澄, 岩崎安希子, 中村友昭, 長井隆行, 横山絢美, 下斗米貴之, 岡田浩之, 大森隆司. 子供と遊ぶロボット: 心的状態の推定に基づいた行動決定モデルの適用. 日本ロボット学会誌, Vol. 31, No. 3, pp.263–274, 2013.