Character-LLM 構築のためのキャラクター設定指示

人見 雄太 [1]　木本 晴久 [1,2]　佐藤 大地 [1,3]　跡部 優吾 [1]　福島 千尋 [1]
池田 愛 [1]　並松 竜太郎 [1]　鈴木 沙英子 [1]　岡村 萌加 [1]　簑田 葉月 [1]
小山 正彦 [1]　片田 智大 [1]　橋本 圭 [1]　ジューストー 沙羅 [1]　守屋 貴行 [1]
[1]Aww, Inc.　[2] 茨城大学　[3] 東京大学
hitomi.yuta@aww.tokyo, 20t4032a@vc.ibaraki.ac.jp, satodai370054@gmail.com, yugo.atobe@aww.tokyo, chihiro.fukushima@aww.tokyo, ai.ikeda@aww.tokyo, ryutaro.namimatsu@aww.tokyo, saeko.suzuki@aww.tokyo, moeka.okamura@aww.tokyo, hazuki.minoda@aww.tokyo, masahiko.koyama@aww.tokyo, tomohiro.katada@aww.tokyo, kei.hashimoto@aww.tokyo, sara.giusto@aww.tokyo, m@aww.tokyo

1　はじめに

Li ら [1] の研究以降，Large Language Model（LLM）を用いてキャラクターを演じさせる Character-LLM の研究が進展している.Character-LLM には主に 2 つの構築方法が存在する. 一つ目は，作品からキャラクター自体の対話データを抽出し，Retrieval Augmented Generation（RAG）[2] を用いて，ユーザーの質問に関連するキャラクターの対話データを LLM に対する追加入力として与える RAG 方式 [1]. 二つ目は，対話データを用いて LLM をファインチューニングするファインチューニング方式である [1, 3]. 本研究では，第 3 の方法として，RAG 方式やファインチューニング方式を補完するための，キャラクター設定指示方式を提案する. この方式は，従来方式で再現できなかった細かなキャラクター設定を反映することを目的とする.

従来の方式では，出力結果が基盤となる LLM の個性や知識に影響を受けるという課題がある [3]. 例えば，表 1 に示されるように，GPT-3.5 や GPT-4[4] を基盤モデルとして用いた場合，RAG で追加入力されたキャラクターの対話データにおいて二人称が「あなた」であっても，GPT が「君」を使用する傾向がある. このように，LLM 毎に出力の傾向が存在し，細かなキャラクターの再現には課題が残る.

本研究では，キャラクター設定指示を導入することで，この問題にアプローチする. たとえば，表 1 の従来モデルの課題に対応するために，「二人称として”あなた”を用いてください」というキャラクター設定指示を追加入力として与える. 具体的には，RAG 方式にキャラクター設定指示を追加し，その効果を検証する. 一方，LLM における指示の表現方法によって，その有効性が大きく変わることが報告されている [5]. このため，本研究では，指示エンジニアリングの自動化に関する最近の手法 [6] を取り入れ，キャラクター設定指示がより反映されるように調整するフェーズを設けた. 実験では 3 つのキャラクター設定指示を用いた提案手法と，従来手法の RAG 方式を比較した. 全てのケースにおいて，提案手法で大幅な改善が確認された.

表 1: RAG 方式の従来モデルと提案手法における生成例

• **ユーザー:** 最後に見た映画は何？
• **RAG 方式のみ従来モデル:** **キャラクター:** んー、最後に見た映画は『インターステラー』だよ！宇宙や時間の概念がすごく面白くて、考えさせられるんだ。君は最近何か面白い映画見た？
• **RAG 方式+提案手法:** **キャラクター:** 最後に見た映画は「インターステラー」だよ。宇宙と時間の概念がめちゃくちゃ面白かったんだ。あなたは最近どんな映画を見たの？

2　提案手法

本研究では，キャラクターの細かな設定を LLM に反映させるための設定指示を導入する. 具体的には，ユーザーからの発話 UU が与えられた際，従来の RAG 方式を用いて，T 個の関連するキャラクター対話を取得する. これらの対話は，集合 $\mathbb{RD} = \{RD_1, RD_2, \ldots, RD_T\}$ として表され，各 RD_t は $f_{RAG}(UU, T)$[1]によって収集された T 個の個別のキャラクター対話データを表す. 本研究では，この $\mathbb{RD}$ に，細かなキャラクター設定を反映した K 個の

1　文埋め込みベクトルへの変換には https://huggingface.co/sentence-transformers/stsb-xlm-r-multilingual を用いた.

キャラクター設定指示 $\mathbb{CP} = \{CP_1, CP_2, \ldots, CP_K\}$ を組み込むことを提案する. このアプローチにより, LLM はより詳細な特徴を反映した生成を可能とする. 各キャラクター設定指示 CP_k は, そのキャラクターの特性, 背景, 動機などを詳細に記述し, これらの設定に基づいた対話生成を促すことを期待する.

LLM は関連対話 $\mathbb{RD}$, キャラクター設定指示 $\mathbb{CP}$, およびユーザーからの発話 UU を組み合わせて, 文脈に合ったキャラクター固有の応答 CR を生成する. このプロセスは, 以下の式で表現される.

$$CR = \mathrm{LLM}\left(\bigoplus_{t=1}^{T} RD_t \oplus \bigoplus_{k=1}^{K} CP_k \oplus UU\right) \quad (1)$$

ここで, $\oplus$ はデータの結合を表し, $\bigoplus$ は複数のデータ集合を結合するための演算子を表す. この式により, LLM は与えられた関連対話, キャラクター設定, およびユーザーからの発話を融合し, 詳細なキャラクター設定を反映した応答を生成することを目指す.

2.1 キャラクター設定指示の自己改善

関連対話 $\mathbb{RD}$ は, T 個の異なる人物の発話 OU とそれに対応するキャラクターの応答 CU のペア (OU, CU) で構成されている. このため, LLM への入力の情報量が増加することで, 一般的な指示を用いる場合に比べて, 各 CP_k の反映が弱まる傾向にある. 事前実験により, $\mathbb{RD}$ を LLM の入力に含める場合, 「回答時に二人称を使用する場合は, 二人称として『あなた』を必ず使う」という指示の反映率が, $\mathbb{RD}$ を LLM の入力に含めない場合より大きく悪化した.

この問題に対処するため, 本研究では設定指示を効果的に反映させるための自動指示改善アプローチを採用する. 具体的には, PromptBreeder [6] という手法を設定指示に特化した形に拡張し, 拡張した PromptBreeder f_{PB} を用いて $RD'_t = f_{PB}(RD_t)$ を生成する. このため, 式 1 を以下のように修正する.

$$CR = \mathrm{LLM}\left(\bigoplus_{t=1}^{T} RD'_t \oplus \bigoplus_{k=1}^{K} CP_k \oplus UU\right) \quad (2)$$

この式により, キャラクター設定指示の反映率の向上が期待される.

2.2 PromptBreeder

PromptBreeder は, 複数の変異オペレーターの活用と遺伝的アルゴリズムを組み合わせることで, 改善したプロンプトを自動生成する手法である. データセット X からサンプリングされた M 個の入力と正解のペアを含む訓練用データセット $D_{\text{train}} = \{(Q_1, A_1), (Q_2, A_2), \ldots, (Q_M, A_M)\}$ を用い, LLM を用いてデータセット X を適切に解くための単一の指示 $P^\star$ を自動生成することを目指す. 指示 $P^\star$ は, PromptBreeder によって得られる多様な指示の集合 $\mathbb{P} = \{P_1, P_2, \ldots, P_N\}$ の中から選ばれる. 各指示 P は与えられた入力 Q と連結し, 指示付きの入力として LLM に与えることで, 応答 $R = \mathrm{LLM}(P \oplus Q)$ を生成する. この際, D_{train} に含まれるすべての (Q_m, A_m) に対して, スコア関数 $f_{fitness}(R, A)$ の総和を最大化する指示 $P^\star$ を求める.

$$P^\star = \arg\max_{P \in \mathbb{P}} \sum_{(Q,A) \in D_{\text{train}}} f_{fitness}(R = LLM([P \oplus Q]), A) \quad (3)$$

2.3 設定指示のための修正

キャラクター設定指示の最適化のために, PromptBreeder に 3 つの改良を加える.1 つ目の改良は, LLM の入力に RAG などの追加入力を与えても, キャラクター設定指示が適切に最適化されるようにするための修正である.2 つ目の改良は, 変異された指示の評価を通じて, 不適切なキャラクター設定指示を個体として生成しないようにするための修正である.3 つ目の改良は, 変異指示, 思考スタイル, および変異オペレータに用いる指示についての日本語化を行う修正である. これらの修正を施した拡張版 PromptBreeder を実験に用いる.

2.3.1 応答生成の修正

PromptBreeder では通常, 単一の指示に対して自動改善を行うが, 提案手法の設定では, RAG による追加入力 $\mathbb{RD}$ とキャラクター設定指示 $\mathbb{CP}$ を含めて入力する必要がある. このため, $\mathbb{RD}$ と $\mathbb{CP}$ を R 生成時の入力として与える. 具体的には, 出力の生成を以下の式によって行う.

$$R = \mathrm{LLM}(\bigoplus_{t=1}^{T} RD'_t \oplus \bigoplus_{k=1}^{K} CP_k \oplus P \oplus Q)$$

この修正により, キャラクター設定指示と関連対話の両方を考慮した上で, 適切な指示 P を生成するこ

とが可能となる.

2.3.2 変異指示の評価

PromptBreeder のプロセスには，新しく生成された指示集合 $\mathbb{P}$ がそれぞれ適切な指示であるか評価するステップが含まれていない. このため，不適切な指示が個体として生成されることがある. 例えば，「二人称を用いる際は，あなたを使ってください」という指示が「必ずあなたを使って回答してください」と変異することで，実用時に全ての質問に対して「あなた」を含めるように促してしまう可能性がある. この問題に対処するため，新しく変異個体が作られた際に，変異個体を評価するプロセスを追加する. この評価は分類問題として行い，分類ラベルはキャラクター設定指示毎に定められる. 以下に，$P_{original}$ が「回答時に二人称を使用する場合は，二人称として『あなた』を必ず使う」の場合の評価ラベルを例示する.

1. 二人称を使うときに限り，二人称として「あなた」を使うように指示している.
2. 二人称を使うときに限り，二人称として「あなた」を使うように指示していない.
3. 元の指示の文意が，改良後の指示に含意されていない.

上記の場合，ラベル 1 と評価された個体を進化個体として人口に含めた. ラベル 1 以外に分類された場合は，ラベル 1 に分類される個体が得られるまで，個体の取得を繰り返すように修正を行った.

2.3.3 変異指示などの日本語化

PromptBreeder の元々の変異指示と思考スタイルは英語で書かれている. そのまま使用してもほとんどの場合問題はないが，生成された新しい変異指示が時折英語になることが確認された. この現象は，変異個体作成時の指示に「必ず日本語の指示を作成してください」という文言を含めても観測された. このため，変異指示，思考スタイル，変異オペレータに用いる指示をすべて著者らによって日本語化した.

2.4 評価データセットの構築

この小節では，各キャラクター設定指示を評価するためのデータセットの構築方法について説明する. まず，LLM を使用して生成された大量の質問 $\mathbb{GQ} = \{GQ_1, GQ_2, \ldots, GQ_J\}$ を集める.$\mathbb{GQ}$ は $P_{original}$ に関連するような質問を生成することを指示として LLM に与えることで生成する.LLM に与えられる質問生成の指示はタスクに応じて異なり，例えば「返答時に二人称を使うような質問を考案してください」といった指示である.$\mathbb{GQ}$ の多様性を担保するため，新しく質問が生成される度に，新しい質問と既存の質問の文ベクトル[2]同士のコサイン類似度を計算し，類似度が 0.8 を越えるペアがあった場合は，新しい質問を $\mathbb{GQ}$ に含めなかった.

評価時の効率性を考慮し，$LLM(\bigoplus_{t=1}^{T} RD'_t \oplus GQ_j)$ で R_j を生成し，R_j が $P_{original}$ に関連するかを LLM に判断させ，関連があると判断された GQ_j のみを評価データセットに追加した.$P_{original}$ に関連するかの判定について，$P_{original}$ が「回答時に二人称を使用する場合は，二人称として『あなた』を必ず使う」の場合を例に説明する. このケースでは，「回答に二人称が含まれるか判定してください」といった指示を LLM に与えて二人称が含まれるかどうかを判定する. その後，少数ながらタスクに関連しない回答が含まれていることがあるため，著者らによってそれら R_j と対応する GQ_j を取り除いた. 評価に用いる $P_{original}$ 毎に 24 件の評価データセットを獲得し，そのうち 12 件を拡張版 PromptBreeder の適応度評価に用い，残りの 12 件をテストセットとして用いた.

3 実験

3.1 実験設定

今回の実験には，バーチャルヒューマンの imma を用いた.RAG に用いる情報源としては，PR TIMES の記事，imma のインタビュー記事，SNS の X における imma のツイート[3]，Aww, Inc. の社員から集めた質問に対する imma の開発者の回答などを用いた. また，実験に使用する CP の作成については，Li ら [1] の RAG 方式のチャットボットを社内で利用してもらい，imma について再現できていない課題を収集した. 収集された課題は著者らによって CP へ変換を行った. 今回は CP の中から，表 2 の 3 つを選択して実験を行った.

拡張版 PromptBreeder の実験は，初期人口を 20，進化のステップ数を 600，temperature については先行研究の値を GPT 向けに調整したものを用いた[4]. す

2 文ベクトルの作成には https://huggingface.co/cl-nagoya/sup-simcse-ja-large[7] を用いた.
3 https://twitter.com/imma__jp
4 初期プロンプト生成時と変異個体生成時では，temperature

表 2: 実験に用いる設定指示一覧

タスク名	CP
二人称	回答時に二人称を使用する場合は、二人称として「あなた」を必ず使う。
一人称	回答時の一人称に「あたし」という言葉を必ず使う。
終助詞	回答時の文末に「なんだ」、「んだ」が来る時は、代わりに「だよ」という言葉を必ず使う。

べての実験において LLM は OpenAI の GPT を使用し，回答生成には gpt-3.5-turbo-1106 を，それ以外の場合は gpt-4-0613 を使用した. 評価は，評価用データセットの質問に対して，式 2 を用いて回答を生成させ，著者らが人手で回答を評価した. 回答の生成において，CP が増えた場合でも効果的であることを示すために，テストに用いない CP を 8 つと，テストに使用する各 CP とを連結し，$\mathbb{CP}$ として式 2 に入力した.

評価指標として，正解率（NA）とターゲット指向正解率（TA）を計算した.NA における正解は，タスク毎に定義される. 例えば，二人称のタスクの場合，文中に二人称として「あなた」以外が使われている場合を不正解とし，それ以外を正解とした.TA に関しては，タスクに関連する単語が回答に含まれているケースのみを抽出し，抽出された回答に対してのみ正解率を計算する. 例えば，二人称のタスクの場合，二人称が含まれる回答のみを抽出し，その中での正解率を計算する. 回答生成時には temperature を 0.8 に設定したため，テストセットの 12 件の各質問に対して 5 回ずつ回答を生成し，計 60 個の回答を評価した.

3.2 実験結果

従来手法および提案手法の正解率とターゲット指向正解率を示す. 表 3 から，従来手法ではタスクに応じて正解率が大きく異なり，最小で 13.3 ポイント，最大で 76.7 ポイントであった. この結果は，GPT が持つ固有の生成傾向が存在することを示唆している. たとえば，一人称のタスクにおいて，従来手法では「私」という単語を用いる割合が約 9 割だった. このように，GPT の内在的な生成傾向は，与えられた指示に優先して反映されることが確認できる.

また，提案手法は全てのタスクにおいて正解率とターゲット指向正解率の両方で従来手法より高い結果となった. 加えて，拡張版 PromptBreeder を用いることで，全てのタスクにおいて改善が見られ，最大で 11.6 ポイントの改善が確認できた. 一方，CP の使用により，全タスクにわたってタスク固有の表現の出現率が上昇したことが観察された. 例として二人称のタスクを挙げると，60 件の評価時の回答において，従来手法での二人称の使用回数は 34 件であったのに対し，RAG 方式+$\mathbb{CP}$ では 41 件，さらに拡張版 PromptBreeder を加えた場合は 45 件と増加した. これは，CP がキャラクターの細かな言語的特徴を効果的に反映させるだけでなく，指示に沿った表現の使用を促進していることを示している. この増加が必ずしも望ましい結果であるとは限らないため，潜在的な悪影響の有無について，今後検証する必要がある. また，ターゲット指向正解率を見てみると，従来手法の方が大きくスコアが下がっていることから，提案手法の方がタスク固有の要求を的確に捉え，反映できていることが分かる.

表 3: 従来手法および提案手法の実験結果. 表の一番上はタスク名に対応し，PB は小節 2.3 で説明した拡張版 PromptBreeder を表す.

	二人称 NA/TA	一人称 NA/TA	終助詞 NA/TA
RAG 方式 [1]	76.7/58.8	13.3/3.7	43.3/40.8
RAG 方式 + $\mathbb{CP}$	86.7/80.4	76.7/72.0	83.3/ 83.3
RAG 方式 + $\mathbb{CP}$ + PB	**93.3/91.1**	**88.3/85.5**	**91.6/91.2**

4 まとめと今後の展望

本研究では，LLM に対するキャラクター設定指示を組み込むことで，Character-LLM の性能向上を図るキャラクター設定指示を提案した. 実験結果は，提案手法が従来手法に比べて，正解率とターゲット指向正解率を大幅に向上させることを示している. 提案手法により，GPT の固有の生成傾向を克服し，よりキャラクター固有の要求に適切に応答する能力を向上させることができた.

今後の研究では，本手法のファインチューニング方式への適用とその効果の検証が必要である. また，異なる状況に応じた柔軟な指示対応の拡張も重要である. 例えば，imma が公式な場で「私」という一人称を使用するようなシナリオにおいて，状況に応じた適切なキャラクター設定指示の適用が求められる. このようなシナリオの多様性を考慮した手法の発展は，Character-LLM の応用範囲をさらに広げる上で重要である.

を 0.5 から 1 の範囲でランダムな値を取得し毎回異なる値を設定した. 報酬計算時の評価と変異個体の評価については temperature を 0 とした.

参考文献

[1] Cheng Li, Ziang Leng, Chenxi Yan, Junyi Shen, Hao Wang, Weishi MI, Yaying Fei, Xiaoyang Feng, Song Yan, HaoSheng Wang, Linkang Zhan, Yaokai Jia, Pingyu Wu, and Haozhen Sun. Chatharuhi: Reviving anime character in reality via large language model, 2023.

[2] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems (NeurIPS 2020)*, 2020.

[3] Yunfan Shao, Linyang Li, Junqi Dai, and Xipeng Qiu. Character-LLM: A trainable agent for role-playing. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13153–13187, Singapore, December 2023. Association for Computational Linguistics.

[4] OpenAI. Gpt-4 technical report. *ArXiv*, Vol. abs/2303.08774, , 2023.

[5] Aman Madaan and Amir Yazdanbakhsh. Text and patterns: For effective chain of thought, it takes two to tango. 2022.

[6] Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution, 2023.

[7] Hayato Tsukagoshi. Japanese simple-simcse. https://github.com/hppRC/simple-simcse-ja, 2023.