修 士 学 位 論 文

題　　目

大規模言語モデルにおけるユーザー嗜好学習方法の重みに基づくモデル変化解析

主査　　吉岡 理文 教授

副査　　森 直樹 教授

副査　　内海 ゆづ子 講師

令和　6　年（　2024　年）度修了

（No.　BGA23130　）　　西村昭賢

大阪公立大学大学院情報学研究科
基幹情報学専攻　知能情報学分野

目 次

1 はじめに **1**

2 要素技術 **2**

2.1 Transformer 2
2.2 Generative Pre-trained Transformers (GPT) 3
2.3 Llama 5
2.4 Qwen 6
2.5 ユーザー嗜好の LLM 出力の制御 6
2.5.1 Supervised Fine-Tuning (SFT) 7
2.5.2 Reinforcement Learning from Human Feedback (RLHF) 7
2.5.3 Direct Preference Optimization (DPO) 8
2.6 Low-Rank Adaptation (LoRA) 8
2.7 量子化 9
2.8 テキスト評価指標 9
2.8.1 BLEU 9
2.8.2 BERTScore 10
2.8.3 LLM による自動評価 (LLM-as-a-judge) 10
2.9 OjousamaTalkScriptDataset 11

3 関連研究 **13**

3.1 LLM によるキャラクターロールプレイ 13
3.2 Transformer 内部の定量的な解析 13
3.3 本研究の位置づけ 15

4 提案手法 **16**

4.1 データセット 16
4.2 Conflict Limited L2 ノルム 17

5 数値実験 **19**

5.1 実験 1 19
5.2 実験 2 23
5.3 実験 3 24

6 結果と考察 **25**

6.1 実験 1 25

6.2 実験 2 28

6.3 実験 3 28

7 まとめと今後の課題 **36**

謝辞 **37**

参考文献 **38**

図 目 次

2.1 Transformer (文献 [1] Figure 1. 参照) のアーキテクチャ 3
2.2 GPT (文献 [2] Figure 1. 参照) のアーキテクチャ 4
2.3 Transfomer を用いた言語モデルの発展 (文献 [3] Figure 1. 参照) 5
5.1 elyza/Llama-3-ELYZA-JP-8B のアーキテクチャ 20
5.2 Qwen/Qwen2.5-7B-Instruct のアーキテクチャ 20

表目次

4.1 データセットの例 17
5.1 QLoRA パラメータ 21
5.2 SFTTrainer パラメータ 22
5.3 DPOTrainer パラメータ 22
5.4 テキスト生成の際のパラメータ 23
6.1 実験 1 における (学習後のモデル, 学習前のモデル) の関係にある 2 モデル間の L1 ノルム, L2 ノルム 25
6.2 実験 1 における SFT を施したモデルとベースモデルとのタスクベクトルにおける絶対値上位 20 % の要素の層ごとの L2 ノルムと平均値 (太字は上位 10 層) 26
6.3 DPO を適用後モデルと DPO を適用前モデルとのタスクベクトルにおける絶対値上位 20 % の要素の層ごとの L2 ノルムと平均値 (太字は上位 10 層) 27
6.4 学習方法が同一でデータセットが異なる 2 モデル間の L1 ノルム, L2 ノルム . . . 28
6.5 $M_{\mathrm{SFT}_{\mathrm{D0}}}$, $M_{\mathrm{SFT}_{\mathrm{D1}}}$ 間の コンフリクト 数, コンフリクト率, Conflict Limited L2 29
6.6 $M_{\mathrm{DPO}_{\mathrm{D0}}}$, $M_{\mathrm{DPO}_{\mathrm{D1}}}$ 間の コンフリクト 数, コンフリクト率, Conflict Limited L2 30
6.7 $M_{\mathrm{DPO}_{\mathrm{D0}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D2}}})}$, $M_{\mathrm{DPO}_{\mathrm{D1}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D2}}})}$ 間の M_{base} を基準とした コンフリクト 数, コンフリクト率, Conflict Limited L2 31
6.8 $M_{\mathrm{DPO}_{\mathrm{D0}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D2}}})}$, $M_{\mathrm{DPO}_{\mathrm{D1}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D2}}})}$ 間の $M_{\mathrm{SFT}_{\mathrm{D2}}}$ を基準とした コンフリクト 数, コンフリクト率, Conflict Limited L2 32
6.9 $M_{\mathrm{DPO}_{\mathrm{D0}}}$, $M_{\mathrm{SFT}_{\mathrm{D0}}}$ 間 コンフリクト 数, コンフリクト率, Conflict Limited L2 33
6.10 $M_{\mathrm{DPO}_{\mathrm{D1}}}$, $M_{\mathrm{SFT}_{\mathrm{D1}}}$ 間 コンフリクト 数, コンフリクト率, Conflict Limited L2 34
6.11 ($M_{\mathrm{DPO}_{\mathrm{D0}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D1}}})}$, M_{base}), ($M_{\mathrm{DPO}_{\mathrm{D0}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D1}}})}$, $M_{\mathrm{SFT}_{\mathrm{D1}}}$), ($M_{\mathrm{DPO}_{\mathrm{D0}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D1}}})}$, $M_{\mathrm{DPO}_{\mathrm{D0}}}$) 間の L1 ノルム, L2 ノルム 35
6.12 実験 2 における (学習後のモデル, 学習前のモデル) の関係にある 2 モデル間の L1 ノルム, L2 ノルム 35

1 はじめに

近年，Generative Pre-trained Transformers (GPT) [2] に代表される大規模言語モデル (Large Language Model, LLM) の急速な発展により，自然言語処理分野において高度なテキスト生成やテキスト理解が実現されている．LLM の応用先は多岐にわたっており，情報検索，文書要約，さらにはプログラミング支援など幅広い領域で活用が進展している．特に，人間と自然な対話を実現するチャットボットの可能性が大きく広がっており，従来の単なる情報伝達手段を超えて，対話を通じてユーザーに親しみや共感を抱かせる能力によりユーザー満足度の向上が期待される．また，ゲームや VR といったエンターテインメント分野においては，キャラクター性を持たせたチャットボットが魅力的かつ没入感のあるコミュニケーションを演出し，ユーザーの感情や体験に深い影響を及ぼす存在として期待されている．

キャラクター性を持たせたチャットボットの実現に向けて，LLM にユーザーの嗜好や属性を組み込みそれらを反映したテキスト生成を実現する研究が注目されている．具体的には，Supervised Fine Tuning (SFT) などの手法を用いてモデルを微調整し，応答スタイルを反映させるアプローチや，Direct Preference Optimization (DPO))[4] のようにユーザーの好みに基づく報酬設計する手法が代表例として挙げられる．これらの手法により，LLM は特定のスタイルの応答を実現するとともに，社会的・倫理的に好ましくない応答の生成を抑制し，安全かつ適切な応答の提供が可能となる．しかし，これらの手法を用いた場合に，モデル内部でどのような変化が生じるか，またその結果として生成出力や性能にどのような影響が及ぶかについては，定量的に十分解明されていない．

本研究では，日本語におけるロールプレイタスクにおいて，SFT および DPO を用いてモデルを調整した場合の学習過程やモデル内部の重み変化を定量的に比較・検討し，それらが生成出力や性能に及ぼす影響を明らかにすることを目的とする．さらに，両手法の併用によって期待される性能向上や，ベースモデルへの回帰可能性についても考察する．本研究の成果は，LLM におけるユーザー嗜好学習の最適化およびその内部動作の理解を深めるための基盤的知見を提供することが期待される．

2 要素技術

2.1 Transformer

Transformer [1] は, Long Short-Term Memory (LSTM) [5],[6] や Gated Recurrent Unit (GRU) [7] に代表される Recurrent Neural Network (RNN) を用いずに, Attention 機構 [8],[9] を基本構造とする Encoder-Decoder モデルである. ここで, Encoder が入力系列 $\boldsymbol{x} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n})$ を連続系列 $\boldsymbol{z} = (\boldsymbol{z_1}, \boldsymbol{z_2}, \ldots, \boldsymbol{z_n})$ へと写像し, Decoder が $\boldsymbol{z}$ から出力系列 $\boldsymbol{y} = (\boldsymbol{y_1}, \boldsymbol{y_2}, \ldots, \boldsymbol{y_n})$ を生成する場合を考える.

図 2.1 に Transformer の概略を示す. Encoder は左側に, Decoder は右側にそれぞれ位置している. N は層数を表しており, $N = 6$ に設定されている.

Transformer は RNN のように再帰構造を持たないため, 入力系列の位置情報を Positional Encoding や Positional Embedding で考慮する. 前者は各位置に対して要素が固定のベクトルを加算する.

Positional Encoding の行列を PE とすると,

$$\begin{aligned} \mathrm{PE}(\mathrm{pos}, 2i) &= \sin\left(\frac{\mathrm{pos}}{p_{\mathrm{freq}}^{2i/d_{\mathrm{model}}}}\right) \\ \mathrm{PE}(\mathrm{pos}, 2i+1) &= \cos\left(\frac{\mathrm{pos}}{p_{\mathrm{freq}}^{2i/d_{\mathrm{model}}}}\right) \end{aligned} \tag{2.1}$$

となる. ただし, d_{model} は入力 Embedding の次元数, pos, i は Positional Encoding の位置および成分である.

一方で, Positional Embedding は各位置に対して要素が学習により可変なベクトルを加算する. 初期値は 0 や乱数などが用いられる.

Encoder の各層は 2 層のサブレイヤおよび Layer Normalization [10] と残差接続 [11] を持つ. 1 層目は Multi-Head Attention であり, 2 層目は単純な Position-wise Feed-Forward Network (FFN) である. FFN は, ReLU 関数を間に有する 2 層の全結合層で構成され, (2.2) 式のように表される.

$$\mathrm{FFN}(\boldsymbol{x}) = \max(\boldsymbol{0}, \boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_2)\boldsymbol{W}_2 + \boldsymbol{b}_2 \tag{2.2}$$

ただし, $\boldsymbol{W}_i, \boldsymbol{b}_i (i \in \{1, 2\})$ はそれぞれ全結合層の重みとバイアスである.

Decoder では, Encoder の 2 層のサブレイヤに加えて, Encoder の出力に対して Multi-Head Attention を計算するための 3 層目のサブレイヤが挿入される.

Attention 機構は, Query, Key および Value への写像として表現される. ここで, Query, Key および Value はそれぞれベクトルである. 出力は Value の加重合計により

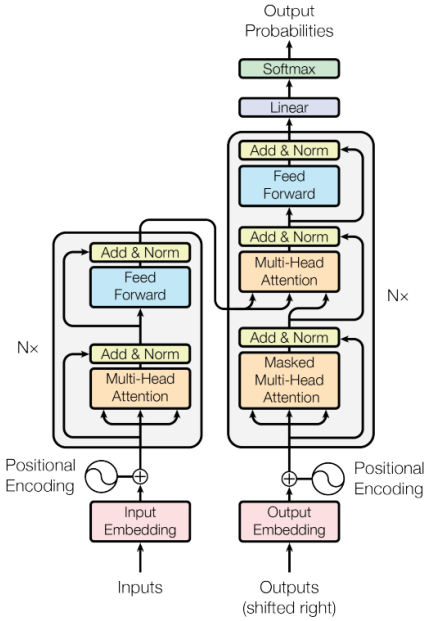

図 2.1: Transformer (文献 [1] Figure 1. 参照) のアーキテクチャ

求められる. なお, 各 Value に対する重みは Query と対応する Key から算出される値である.

2.2 Generative Pre-trained Transformers (GPT)

Transformer を用いたモデルとして, Bidirectional Encoder Representations from Transformers (BERT) [12], コンピュータビジョンの分野では Vision Transformer (ViT) [13] に代表される Encoder のみを用いたモデル, Text-to-Text Transfer Transformer (T5)[14] に代表される双方を使用した Encoder-Decoder モデルが存在する. そして Transformer の Decoder のみを使用した代表的なモデルには Generative Pre-trained Transformers (GPT) [2] が挙げられる.

図 2.2 に GPT にのアーキテクチャを示す. GPT は Decoder only のモデルを用いているため, 図 2.1 の右側の Decoder に存在する Multi Head Attention は無くなっている. このモデルにおいてトークン系列 $U = (u_{-k}, \ldots, u_{-1})$ からトークン u を予測する場合を考える. このとき, 入力テキストの埋め込み表現 W_e, position embedding の位置表現 W_p を用いて Embedding 層の計算は (2.3) 式のように表される.

$$h_0 = UW_e + W_p \tag{2.3}$$

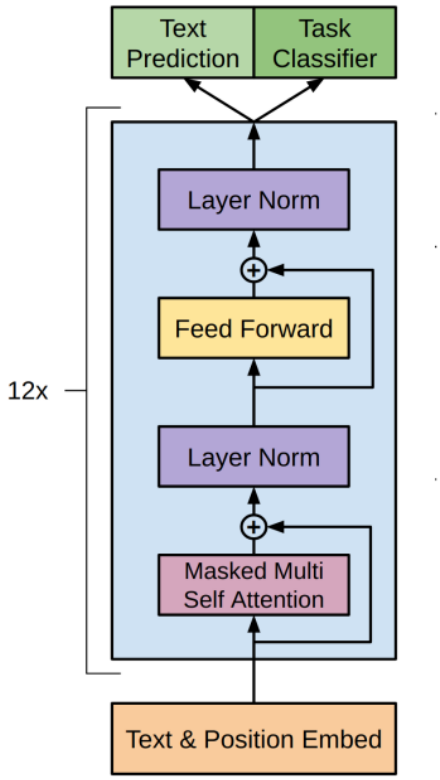

図 2.2: GPT (文献 [2] Figure 1. 参照) のアーキテクチャ

そして, Masked Mulit Self Attention 層, Feed Forward 層, Layer Norm 層で構成される構造を transformer_block と置くと, transfomer_block のレイヤー数を n として, トークン u の予測確率は (2.5) 式のように表される.

$$h_l = \text{transfomer\_block}(h_{l-1}), \forall l \in [1, n] \tag{2.4}$$

$$P(u) = \text{softmax}(h_n W_e^T) \tag{2.5}$$

GPT では $n = 12$ としており, このモデルをラベルなしデータでの事前学習と少量のラベル付きデータによるファインチューニングにより応用可能性が広いモデルを実現している.

GPT は近年急速に進化を続けており, 2020 年には OpenAI から 1750 億パラメータという大規模なパラメータをもつ GPT-3[15], 2023 年には GPT-3 の改良版である GPT-3.5 をさらに複雑な推論を可能とした GPT-4[16], 2024 年には GPT-4 を超える推論性能を残す OpenAI o1 が公開されている. GPT-4 以降のモデルはテキストだけでなく画像などマルチモーダルな入力が可能となっており注目を集めているが, 安全性

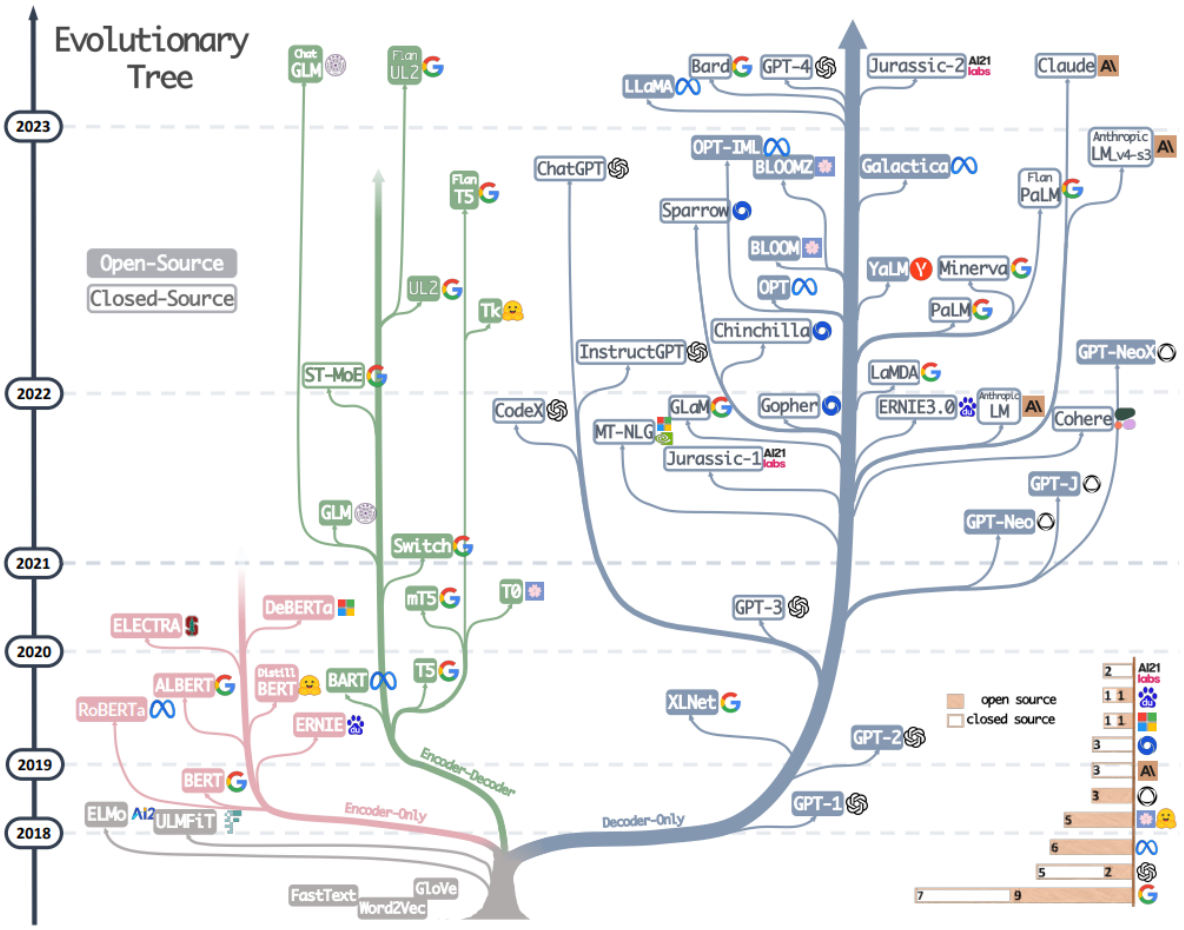

図 2.3: Transfomer を用いた言語モデルの発展 (文献 [3] Figure 1. 参照)

や競争のリスクから具体的なパラメータ数や学習方法, 学習データセットなどは公開されていない.

2.3　Llama

図 2.3 に Transfomer を用いた言語モデルの発展を示す. 近年の LLM はほとんどが Decoder-only モデルとなっている.

Meta によって公開された Llama [17] も Decoder only のモデルの 1 つであり, 一般に利用可能なデータセットのみを用いて学習され, ほとんどのベンチマークで Llama2 を上回る精度を達成した. 2023 年に Llama を改良した Llama2[18] が登場し, 2024 年には更に複雑な推論を可能とした Llama3 [19] が登場した. Llama3 から表現力の向上, 処理速度の向上などの改良がなされた Llama3.1 は, 8B, 70B, 405B の 3 つのモデルサ

イズが用意されており, 57 の多様なタスクで構成される言語理解ベンチマークである MMLU (Massive Multitask Language Understanding)[20], 与えられた指示に基づいて Python コードを生成する能力を評価するベンチマークである HumanEval[21], 小学レベルの数学の文章問題を解く能力を評価するベンチマークである GSM-8K[22] において, Llama3 405B は GPT-4 を超える性能を残している.

2.4 Qwen

Qwen[23] は Alibaba が公開してる大規模言語モデルであり, 2023 年に公開された Qwen-72B は MMLU, HumanEval, GSM-8K などのベンチマークにおいて Llama2-70B と比較して高い数値を記録した.

2024 年に公開された Qwen-2.5 [24] は 0.5B, 1.5B, 3B, 7B, 14B, 32B, 72B などの幅広いパラメータ数のモデルが公開されており, それに加えてコード生成に特化した Qwen2.5-Coder[25], 数学向けに特化した Qwen2.5-Math[26] と行った特定のタスクに特化したバージョンも公開されている.

2.5 ユーザー嗜好の LLM 出力の制御

LLM にはプロンプトと呼ばれる実行指示を記述した文を入力として与えることで与えた指示に従ったユーザーの求める応答を生成する. より高性能かつユーザーの要望に沿った応答を実現するための最も簡単なアプローチとしてプロンプトエンジニアリングが挙げられる. プロンプトエンジニアリングとは LLM に与えるプロンプトを最適化する手法である. 特定のタスクに取り組む際にいくつかの (入力, 期待される出力) の例を与える Few-shot Prompting, 中間的な推論を例として与えることで複雑な推論を可能にする Chain of Thoughts(CoT)[27] など, プロンプトエンジニアリングは多くの手法が考案されている [28]. また, 遺伝的アルゴリズムを用いてプロンプトエンジニアリングを自動化する PromptBreeder[29] といった研究もなされている. このようなプロンプトエンジニアリングは, LLM 内部の重みを変えずにユーザーが求める応答を得ることができるという点で OpenAI API に代表される API 経由でのみアクセスすることができる LLM を活用する際に有効なアプローチとなる.

さらにモデル内部の重みに干渉可能なローカル LLM を活用できる場合には, Supervised Fine-Tuning (SFT) や Direct Preference Optimization (DPO) のような手法で

LLM をファインチューニングし LLM 内部の重みまで変えることでユーザーが望む応答が実現しやすくなることが期待できる.

2.5.1 Supervised Fine-Tuning (SFT)

Supervised Fine-Tuning (SFT) はベースとなる LLM に対して, 入力と人間が「好ましい」と考える模範解答のペアを用意しこれを直接の教師データとしてモデルを教師ありファインチューニングする手法である. SFT の代表的な例として instruction tuning が挙げられる. 膨大な量のテキストデータで事前学習された LLM はそのままでは入力となる文の続きを生成する振る舞いとなり, ユーザーの指示を踏まえた対話的な形でのテキスト生成はできない. そこで Alpaca Dataset[1] に代表される大規模な instruction tuning 用のデータセットで instruction tuning することで, より広範な指示に対応できる LLM を生み出すことができる.

2.5.2 Reinforcement Learning from Human Feedback (RLHF)

プロンプトに対して人間が「好ましい」と考える応答, また「好ましくない」と考える応答のペアを学習データセットとして, 好ましい応答の生成確率を高め, 好ましくない応答の生成確率を抑制する形で LLM を学習する手法を preference training と呼ぶ. preference training の代表的な手法として Reinforcement Learning from Human Feedback (RLHF)[30], Direct Preference Optimization (DPO) がある.

RLHF は以下の 3 プロセスから構成される.

1. データ収集と言語モデルの事前学習
2. 報酬モデルの学習
3. PPO[31] を用いた強化学習による言語モデルのファインチューニング

データ収集と言語モデルの事前学習ではラベル付け担当者がデータセットからランダムに抽出されたデータセットから好ましい応答を提示しそのデータで LLM を SFT する.

報酬モデルの学習では, 同じプロンプトを条件とする 2 つの言語モデルから生成されたテキストをラベル付け担当者が比較し集まった比較データでユーザー嗜好をスカラー値で返す報酬モデルを学習する.

[1] https://huggnngface.co/datasets/tatsu-lab/alpaca

最後に強化学習アルゴリズムである PPO で事前学習済みのモデルをファインチューニングする. 具体的にはプロンプトと回答が与えられると, 報酬モデルによって報酬を生成する. 次に報酬モデルの過剰最適化を緩和するために, 事前学習済みモデルと強化学習モデルのトークンごとの KL divergence[32] を求める. 報酬と KL divergence を基にモデルの重みを更新することで強化学習モデルが事前学習済みから大きく離れることを防ぎつつ高い報酬を獲得するように学習を進行させることができる.

RLHF は事前学習済みモデル, 報酬モデル, 強化学習モデルの 3 つを同時に扱うため, 大量の計算資源を必要とする, 他のモデルの出力を基に学習をする必要があるため並列化が難しく学習に時間がかかるといった問題がある.

2.5.3 Direct Preference Optimization (DPO)

DPO は RLHF における報酬モデルを不要とした手法であり, (2.6) 式と表される損失関数を最適化する手法である. ここで, D_{pair} はデータセット, x はプロンプト, y^w はプロンプト x に対する好ましい出力, y^l はプロンプト x に対する好ましい出力であり, π_θ は学習対象の LLM, π_{ref} は事前学習済みのモデルである.

$$L_{\text{DPO}} = -\mathbb{E}_{(x,y^w,y^l)\sim D_{\text{pair}}}\left[\log \sigma(\beta \log \frac{\pi_\theta(y^w \mid x)}{\pi_{\text{ref}}(y^w \mid x)} - \beta \log \frac{\pi_\theta(y^l \mid x)}{\pi_{\text{ref}}(y^l \mid x)})\right] \tag{2.6}$$

DPO は数学的に RLHF と等価であることが知られており, 単純な勾配法でモデルを直接最適化する手法であるため上記で述べた RLHF の欠点を改善した手法といえる.

2.6 Low-Rank Adaptation (LoRA)

Low-Rank Adaptation (LoRA) [33] とは, 学習するパラメータ数を削減しつつ fine-tuning する手法である. モデルの線形層のパラメータを $D_{\text{in}} \times D_{\text{out}}$ 次元の行列 $\boldsymbol{W}$ とし, 入力ベクトルを $\boldsymbol{x}$ とすると, 出力 $\boldsymbol{h}$ は (1) 式で表される.

$$\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} \tag{2.7}$$

LoRA では線形層のパラメータ $\boldsymbol{W}$ と同次元の差分行列 $\Delta\boldsymbol{W}$ を用意し, 出力 $\boldsymbol{h}$ は (2) 式で表される. 学習の際には $\boldsymbol{W}$ を固定し, $\Delta\boldsymbol{W}$ のみを学習する.

$$\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} + \Delta\boldsymbol{W}\boldsymbol{x} \tag{2.8}$$

この時, ランク r を設定し, $D_{\text{in}} \times r$ 次元の行列 $\boldsymbol{A}$, $r \times D_{\text{out}}$ 次元の行列 $\boldsymbol{B}$ で, $\Delta\boldsymbol{W}$ は (3) 式で表せる.

$$\Delta\boldsymbol{W} = \boldsymbol{A}\boldsymbol{B} \tag{2.9}$$

$\boldsymbol{W}$ のパラメータ数は $D_{\text{in}} \times D_{\text{out}}$ となる一方で $\Delta\boldsymbol{W}$ のパラメータ数は $r\,(D_{\text{in}} + D_{\text{out}})$ となり, 一般的に r は D_{in}, D_{out} に比べて非常に小さい値であるため, 学習パラメータ数を大きく減らすことができる.

2.7 量子化

大規模なニューラルネットワークや LLM などの膨大なパラメータを持つモデルでは, 演算の際に膨大な数の乗加算を必要とするため演算に時間がかかる. また, 多くのパラメータを保持するために多くのメモリが必要となる. これらの問題を軽減するためのアプローチとして量子化がある. 一般的に LLM の学習や推論では 16 ビット浮動小数点 (FP16) が用いられることが多いが, これを量子化し 4 ビットに変換する際には FP16 で表現されているパラメータを 2^4通りの値いずれかにマッチングする. 量子化により精度は減少するものの消費するメモリ量を大幅に軽減することができる.

本研究で用いた Quantized Low-Rank Adaption (QLoRA) [34] では LLM のパラメータの多くが正規分布に従うことを利用した NormalFloat4 という量子化手法により 4 ビット量子化した LLM において効率的に fine-tuning できることを示している.

2.8 テキスト評価指標

テキスト生成タスクにおける評価指標として, モデルが生成した文と参照文の類似度を測定することが一般的となっている. 文同士の類似度を定量化した指標として BiLingual Evalution Understudy (BLEU)[35], BERTScore[36] がある.

2.8.1 BLEU

BLEU スコアは機械翻訳の品質を評価するために広く使用される指標であり, 生成文と参照文とをコーパス単位で比較して計算される. BLEU スコアは次のように計算される.

$$\text{BLEU} = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{2.10}$$

ここで, p_n は n グラムの精度を表し, w_n は各 n グラム精度に割り当てられた重みである. Brevity Penalty (BP) は、生成された文が参照文の内容を完全にカバーしていない場合に調整をする.

$$BP = \min\left(1, \exp\left(1 - \frac{\text{参照文の総単語数}}{\text{生成文の総単語数}}\right)\right) \tag{2.11}$$

n グラム精度 p_n は, 生成された n グラムの総数に対する一致した n グラムの割合として計算される.

$$p_n = \frac{\sum_{i=1}^{M} \text{生成文における n グラム数}}{\sum_{i=1}^{M} \text{参照文における n グラム数}} \tag{2.12}$$

また, m_n は生成されたテキストの n グラムと参照文との一致数を示す.

$$m_n = \sum_{i=1}^{M} \text{生成文と参照文で一致した n グラム数} \tag{2.13}$$

BLEU スコアは 0 から 1 の範囲で評価される. ただし, このスコアは n グラムの精度に大きく依存しており, 生成文と参照文との間で多くの一致する n グラムがあればBLEU スコアは高くなる. BLEU スコアは文同士の類似度を測る指標として有用であるが, より詳細な評価には人間の評価と組み合わせて使用することが重要である.

2.8.2 BERTScore

BERTScore は、事前学習された BERT モデルを使用し文の意味的類似度を測定するために考案された指標である. BLEU に代表される n グラムベースの精度評価方法とは異なり, BERTScore は文脈を考慮した埋め込みベースでの類似度を計算するためテキスト中の語が完全に一致する必要がなく, より人間に近い翻訳評価を提供する. BERTScore の計算には, まず各単語の埋め込みベクトルが BERT モデルを用いて抽出される. ここで, トークン長 N の参照文, トークン長 M の生成文をそれぞれ BERT に入力し, 参照文のトークン埋め込み列 $\mathbf{x}_1, ..., \mathbf{x}_N$ と, 参照文のトークン埋め込み列 $\mathbf{y}_1, ..., \mathbf{y}_M$ を得た後にコサイン類似度を用いてトークン埋め込み列間の類似度を算出し, 以下の式で適合率, 再現率, F 値を計算する.

$$\text{適合率} = \frac{1}{M} \sum_{j=1}^{M} \max_{1 \le i \le N} \frac{x_i^\top x_j}{\|x_i\| \|x_j\|} \tag{2.14}$$

$$\text{再現率} = \frac{1}{N} \sum_{i=1}^{N} \max_{1 \le j \le M} \frac{x_i^\top x_j}{\|x_i\| \|x_j\|} \tag{2.15}$$

$$\text{F 値} = \frac{2 \cdot \text{適合率} \cdot \text{再現率}}{\text{適合率} + \text{再現率}} \tag{2.16}$$

2.8.3 LLM による自動評価 (LLM-as-a-judge)

従来の NLP タスクの学習では, 評価指標に基づく高評価が得られても人間の評価から見た品筆の高さを保証するものではない点が問題点として挙げられる. たとえば

BLEU, ROUGE[37] のようなテキスト要約用の評価指標は忠実性や事実性との相関が低いことが報告されており, ROUGE の値が高い場合でも Hallucination (幻覚) が生成されている [38]. また, 正解が複数ある場合に対応できないため本研究のような LLM のロールプレイの出力などの評価は BLEU や ROUGE 単体だけだと難しい.

そのような背景の中, LLM の発展により GPT-4 などの高性能の LLM を評価者として利用する LLM-as-a-judge という手法が登場した. LLM-as-a-judge は従来の評価指標よりも柔軟で精度の高い評価を可能にしており, 対話型 AI, 情報検索といった分野で有望な応用が期待できる手法である [39]. LLM-as-a-judge によって GPT-4 を評価者 LLM として他の LLM の性能を評価する Japanese Vicuna QA BenchMark といったベンチマークも存在する [40].

しかし, LLM-as-a-judge には以下の課題もあり, テキストの評価には複数の評価指標と人から見た定性的な評価を組み合わせて評価する必要がある.

- LLM の評価は完全に公平ではなく, 事前学習データなどから発生するバイアスを含む可能性がある
- 評価者 LLM による評価の一貫性が保証されていない
- 評価者の性能に大きく結果が依存される

2.9 OjousamaTalkScriptDataset

OjousamaTalkScriptDataset [2] は一般的な問いかけとお嬢様スタイルの応答がペアとなっている会話データを 202 件収録している MIT ライセンスで公開されたデータセットである. また, OjousamaTalkScriptDataset におけるお嬢様は以下に示すキャラクターの設定が与えられている.

- 高校生
- 女性
- 外見の設定はあえてしていません
- ミュージカルが好きでミュージカル女優に憧れていた
- 両親は不動産業
- 兄はアメリカに留学中
- バイオリンを子供の頃から習っている

[2] https://github.com/matsuvr/OjousamaTalkScriptDataset

- 一時期、祖父の住む長野で暮らしていた
- 現在は東京在住

3 関連研究

本章では LLM による特定のキャラクターのロールプレイに関する研究, また Transformer の解析に関する研究を紹介し, 本研究の位置づけを明確にする.

3.1 LLM によるキャラクターロールプレイ

ロールプレイの概念は, 単なる AI アシスタントの役割を超えて人々が求める心理的・娯楽的なニーズを満たすために発展してきた.

Character-LLM: A Trainable Agent for Role-Playing[41] では, ルートヴィヒ・ヴァン・ベートーヴェンなどの歴史上の人物の精神的な活動や物理的な行動を模倣し再構築した経験を用いて LLM に SFT を適用することでキャラクターを演じる LLM, Character-LLM を提案した. 研究では, LLM の性能を ChatGPT によって生成された質問に答えたデータを GPT-3.5 によって, Memorization, Values, Personality, Hallucination, Stability の 5 段階で評価した.

ChatHaruhi: Reviving Anime Character in Reality via Large Language Model[42] では, ユーザーのクエリに対してシステムプロンプト, キャラクターの記憶, 対話履歴などを組み合わせることで LLM のロールプレイの性能を向上させるフレームワークを提案している. また, 32 人のアニメやドラマなどに登場するキャラクターについて実際の脚本に登場した会話例に加えて SFT したモデルによって対話データを拡張し, 合計約 54000 個の会話データからなるデータセットを構築した.

RoleLLM: Benchmarking, Eliciting, and Enhancing Role-Playing Abilities of Large Language Models[43] では, 100 個のロールについて GPT を用いてロール固有の QA ペアを生成し, キャラクターごとの 400 以上の質問データセットを作成し, システムプロンプトを組み込んだローカル LLM のファインチューニングにより従来のオープンソース LLM と比較してロールプレイ能力が大幅に向上し, かつ新しいロールへの適応能力も高い LLM を学習する方法を提案した. また, LLM のロールプレイ能力を測る RoleBench[3] といった新しいベンチマークを提案した.

3.2 Transformer 内部の定量的な解析

What do you learn from context? Probing for sentence structure in contextualized word representations [44] では, CoVe[45], ELMo[46], BERT, GPT の 4 つの異なる文埋め込みモ

[3] https://huggingface.co/datasets/ZenMoore/RoleBench

デルに 9 種類の NLP タスクを適用して 4 種類のモデルの性能を解析し, 言語表現をどのように内部に保持しているかを調査した. NLP タスクは以下の通り. なお, SPR においてはデータセットが 2 種類存在し論文中では分けて 2 タスクとして扱っている.

- Part-of-speech tagging (POS)
  各単語に対して品詞を割り当てるタスク
- Constituent labeling
  構造木の中で句や節と行った構成素にラベルを割り当てるタスク
- Dependency labeling
  文中の係り受け関係などの依存関係とそのラベルを予測するタスク
- Named entity labeling
  人名や地名と行った固有表現にラベルを割り当てるタスク
- Semantic role labeling (SRL)
  動詞や述語を中心に，文中の各項のどんな意味的役割を割り当てるタスク
- Coreference
  文中で同じ実体を指す表現を結びつけるタスク
- Semantic proto-role (SPR)
  SRL において, より微細な概念的・意味的特徴をアノテートしているデータセット. 動詞と引数のペアについてその引数がどんな意味的性質を持つかを複数ラベルで判定するタスク
- Relation Classification (Rel.)
  2 つのエンティティに対して，それらの関係を割り当てるタスク

これらのタスクにより単語情報から意味関係まで幅広い言語表現のタスクにより文埋め込みモデルがどの層でどの言語情報を埋め込んでいるのかを分析することができ, この手法を edge probing と呼ぶ.

edeg probing による解析の結果, BERT が最も優れたスコアを残し, 従来の単語埋め込みと比べ文埋め込みモデルは依存構造約構造など統語的タスクに置いては顕著に改善が見られたが, より高次の意味的タスクでは限定的な改善しか見られないといった結果を得た.

BERT Rediscovers the Classical NLP Pipeline[47] では edge probing を用いて BERT の各層が文法や意味情報をどのように処理しているか解析した. 結果として, BERT において POS や構文解析などの文法情報は 1 から 7 層の低層で処理され, 意味役割付与 (SRL) や Coreference のような高レベルの位置情報は 9 から 20 層の高層で処理

される傾向があった.

3.3 本研究の位置づけ

Character-LLM, ChatHaruhi, RoleLLM においてファインチューニングした LLM の性能評価では生成結果を ROUGE-L, LLM-as-a-judge といった評価指標による評価 人手評価, 定性的な評価がなされている. ロールプレイのタスクにおいて SFT や DPO といったファインチューニングでユーザー嗜好学習をした LLM の重みの変動に基づいた定量的な考察は十分になされていない.

また, Transformer の重みに関しても 先行研究では BERT という分類モデルでの考察にとどまっており, LLM のような decoder only の生成モデルに関しては十分に考察されていない.

そのため, 本研究では日本語でのデータセットを用いたロールプレイタスクにおいて SFT, DPO などの手法を用いて手法とデータセットがそれぞれ LLM 内部にどのような重みの変動をもたらすか調査し両手法, データセットによる違いを定量的に解析する.

4 提案手法

4.1 データセット

本研究では OjousamaTalkScriptDataset を基に ChatGPT o1 pro を用いて新たに一般的な応答を 202 件収録したデータセット, 男子大学生の応答を 202 件収録したデータセットを新たに構築した.

OjousamaTalkScriptDataset を参考に男子大学生のキャラクターには以下の設定を与えている.

- 大学生
- 男性
- 外見の設定はあえてしていません
- 一人称は俺で関西弁
- ぶっきらぼうな物言いだが、内面は熱い情熱と誠実さを秘めている
- 実家は滋賀の大津市にあり、両親は共働きのサラリーマン
- 小さい頃から病弱な二歳年下の妹の面倒を見てきた
- 大阪の公立大学に進学し、公務員となるため勉学に励んでいる

データセット構築のため, ChatGPT o1 pro に与えたプロンプトを以下に示す.

一般的な応答を収録したデータセットを構築するプロンプト

以下はお嬢様の応答のデータセットである。
お嬢様の comletion 以下の応答部分をすべて同じ意味でありながら可能な限り没個性的な応答に書き換えよ。
出力は同じ prompt 順の同じ json 形式とし、応答内の文以外は変えないこと。
–ここから–
{ OjousamaTalkScriptDataset の jsonl 形式のデータセット }

男子大学生の応答を収録したデータセットを構築するプロンプト

以下はお嬢様の応答のデータセットである。
お嬢様の comletion 以下の応答部分をすべて同じ意味でありながら以下に定義された男子大学生の応答に書き換えよ。
出力は同じ prompt 順の同じ json 形式とし、応答内の文以外は変えないこと。
変換する男子大校生の定義：
- 大学生 - 男性 - 外見の設定はあえてしていません - 一人称は俺で関西弁 - ぶっきらぼうな物言いだが、内面は熱い情熱と誠実さを秘めている - 実家は滋賀の大津市にあり、両親は共働きのサラリーマン - 小さい頃から病弱な二歳年下の妹の面倒を見てきた - 大阪の公立大学に進学し、公務員となるため勉学に励んでいる
生成時の注意点： - このペルソナは参考としているお嬢様とは完全に無関係である - このペルソナに合致しない生成はしないこと． 以下の内容に関連する生成は今回のペルソナでは不可: なルール - 兄がいる - 高級志向で高級という言葉を多用する - 海外旅行の経験がある - ミュージカルやバイオリンに興味があり話題にする - 女性的に内容 - 知らないこと答えられないことはわからないと答えて良い - データ数の追加欠損は付加 - 生成したデータが上記の不可なルールに抵触しないか厳密にチェック．男子大学生のペルソナに合致しない場合は合致するまで生成をやり直すこと
–ここからデータ–
{ OjousamaTalkScriptDataset の jsonl 形式のデータセット }

表 4.1 に「夕日に向かって走ろう」といったクエリに対する各データセットの応答を示す.

表 4.1: データセットの例

応答スタイル	「夕日に向かって走ろう」のクエリに対する応答
お嬢様	とてもロマンチックな気分になりそうですわね
一般的	ロマンチックな気分が味わえそうです
男子大学生	めっちゃロマンチックな気分になりそうやな

4.2 Conflict Limited L2 ノルム

本手法は Ties-Merging[48] から着想を得ている. Ties-Merging はモデルマージにおけるマージ手法の 1 つであり, モデルマージにおいて下記のタスクベクトルを導入し

て異なるタスクにおけるモデルの知識を事前学習モデル + タスクベクトルという形で統合することで複数タスクに汎用的に対応できるモデルマージを実現している.

ここで, あるベースモデル M_{base} を特定のタスク T に対してファインチューニングして得られたモデル M_{T} について, M_{base} のパラメータを θ_{base}, M_{T} のパラメータを θ_{T} とするとタスクベクトル Δ_{T} は (4.1) 式と表される.

$$\Delta_{\text{T}} = \theta_{\text{T}} - \theta_{\text{base}} \tag{4.1}$$

Ties-Merging を用いた場合, 以下の 3 ステップでモデルマージをする.

1. パラメータの変化量が小さい値を切り捨て

   全てのパラメータを含んだタスクベクトルの単純な平均を取ると, あるモデルにおける重要な変化が他のモデルの冗長な変化により相殺される問題がある. 論文中では各タスクベクトルにおいて変化量上位 20 % のみ保持し, 残り 80 % は変化量を 0 とし冗長なパラメータによる干渉を防ぐ.

2. パラメータの符号の一致

   同一パラメータに置いてタスクベクトル間で変化の方向が異なることがある. タスクベクトルの単純な平均を取ると正負の変化が相殺され両タスクについて性能が低下する問題がある (符号の衝突). そのため, 各パラメータについてタスクベクトルの対応するパラメータの支配的な符号を決定する.

3. 符号が一致するパラメータのみを平均化して統合

   タスクベクトルの各パラメータに対して, 支配的な符号と一致する成分のみ平均を計算しマージに用いるタスクベクトルの成分とする.

ここで, 符号の衝突に関してはどのタスク同士が強く対立しやすいかあるいはどのパラメータ領域で相反的に学習が進むかを示していると解釈でき, タスク間の相関を定量的に捉える一助となる.

本研究で提案する Confilct Limited L2 ノルム ではデータセット間, 学習手法間の差を定量的に確認するため符号の衝突が起こっている部分に限定した L2 ノルムを計算する. ベースモデル M_{base} を特定のタスク $T1$, $T2$ に対してファインチューニングして得られたモデルのパラメーア $\theta_{\text{T1}}, \theta_{\text{T2}}$ においてそれぞれのベースモデルからのタスクベクトル $\Delta_{\text{T1}}, \Delta_{\text{T2}}$ において, コンフリクト数を C とすると (4.2) 式と表される.

$$\text{Conflict Limited } \|\Delta W\|_2 = \sqrt{\sum_{i \in C} \left(\Delta^i_{\text{T1}} - \Delta^i_{\text{T2}}\right)^2} = \sqrt{\sum_{i \in C} \left(\theta^i_{\text{T1}} - \theta^i_{\text{T2}}\right)^2} \tag{4.2}$$

5 数値実験

本研究で用いたデータセットを以下のように記載する.

- D_0 : 一般的な応答を 202 件収録したデータセット
- D_1 : お嬢様スタイルの応答を 202 件収録したデータセット
- D'_1 : 男子大学生スタイルの応答を 202 件収録したデータセット
- D_2 : D_0, D_1 を半数ずつ用いて計 202 件収録したデータセット
- D'_2 : D_0, D'_1 を半数ずつ用いて計 202 件収録したデータセット

また、本研究で使用したベースモデルは以下の 2 つ. 図 5.1, 5.2 に 2 つのモデルのアーキテクチャを示す.

- elyza/Llama-3-ELYZA-JP-8B[4]

  ELYZA 社が公開しているモデル. Llama3 に instruction tuning を施した meta-llama/Meta-Llama-3-8B-Instruct に対してさらに大規模な日本語コーパスを用いて追加事前学習, instruction tuning をし日本語に対する指示遂行能力を強化した.

- Qwen/Qwen2.5-7B-Instruct[5]

  Qwne2.5 に大規模なデータセットで instruction tuning をして対話性能を強化したモデル. 日本語, 英語をはじめとしてフランス語やスペイン語など 29 以上の言語に対応している.

5.1 実験 1

ベースモデル M_{base} として, elyza/Llama-3-ELYZA-JP-8B を用いる. ベースモデルに対して, 以下のモデルをファインチューニングする. ファインチューニングするにあたって各データセットは事前に train : test = 8 : 2 となるよう分割されている.

- $M_{\text{SFT}_{\text{D0}}}$: M_{base} に D_0 を用いて SFT を適用
- $M_{\text{SFT}_{\text{D1}}}$: M_{base} に D_1 を用いて SFT を適用

[4]https://huggingface.co/elyza/Llama-3-ELYZA-JP-8B
[5]https://huggingface.co/Qwen/Qwen2.5-7B-Instruct

```
LlamaForCausalLM(
  (model): LlamaModel(
    (embed_tokens): Embedding(128256, 4096)
    (layers): ModuleList(
      (0-31): 32 x LlamaDecoderLayer(
        (self_attn): LlamaSdpaAttention(
          (q_proj): Linear(in_features=4096, out_features=4096, bias=False)
          (k_proj): Linear(in_features=4096, out_features=1024, bias=False)
          (v_proj): Linear(in_features=4096, out_features=1024, bias=False)
          (o_proj): Linear(in_features=4096, out_features=4096, bias=False)
          (rotary_emb): LlamaRotaryEmbedding()
        )
        (mlp): LlamaMLP(
          (gate_proj): Linear(in_features=4096, out_features=14336, bias=False)
          (up_proj): Linear(in_features=4096, out_features=14336, bias=False)
          (down_proj): Linear(in_features=14336, out_features=4096, bias=False)
          (act_fn): SiLU()
        )
        (input_layernorm): LlamaRMSNorm((4096,), eps=1e-05)
        (post_attention_layernorm): LlamaRMSNorm((4096,), eps=1e-05)
      )
    )
    (norm): LlamaRMSNorm((4096,), eps=1e-05)
    (rotary_emb): LlamaRotaryEmbedding()
  )
  (lm_head): Linear(in_features=4096, out_features=128256, bias=False)
)
```

図 5.1: elyza/Llama-3-ELYZA-JP-8B のアーキテクチャ

```
Qwen2ForCausalLM(
  (model): Qwen2Model(
    (embed_tokens): Embedding(152064, 3584)
    (layers): ModuleList(
      (0-27): 28 x Qwen2DecoderLayer(
        (self_attn): Qwen2SdpaAttention(
          (q_proj): Linear(in_features=3584, out_features=3584, bias=True)
          (k_proj): Linear(in_features=3584, out_features=512, bias=True)
          (v_proj): Linear(in_features=3584, out_features=512, bias=True)
          (o_proj): Linear(in_features=3584, out_features=3584, bias=False)
          (rotary_emb): Qwen2RotaryEmbedding()
        )
        (mlp): Qwen2MLP(
          (gate_proj): Linear(in_features=3584, out_features=18944, bias=False)
          (up_proj): Linear(in_features=3584, out_features=18944, bias=False)
          (down_proj): Linear(in_features=18944, out_features=3584, bias=False)
          (act_fn): SiLU()
        )
        (input_layernorm): Qwen2RMSNorm((3584,), eps=1e-06)
        (post_attention_layernorm): Qwen2RMSNorm((3584,), eps=1e-06)
      )
    )
    (norm): Qwen2RMSNorm((3584,), eps=1e-06)
    (rotary_emb): Qwen2RotaryEmbedding()
  )
  (lm_head): Linear(in_features=3584, out_features=152064, bias=False)
)
```

図 5.2: Qwen/Qwen2.5-7B-Instruct のアーキテクチャ

- $M_{\mathrm{SFT}_{\mathrm{D2}}}$: M_{base} に D_2 を用いて SFT を適用
- $M_{\mathrm{DPO}_{\mathrm{D0}}}$: M_{base} に D_0 を chosen, D_1 を rejected として DPO を適用
- $M_{\mathrm{DPO}_{\mathrm{D1}}}$: M_{base} に D_1 を chosen, D_0 を rejected として DPO を適用
- $M_{\mathrm{DPO}_{\mathrm{D0}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D2}}})}$: $M_{\mathrm{SFT}_{\mathrm{D2}}}$ に D_0 を chosen, D_1 を rejected として DPO を適用
- $M_{\mathrm{DPO}_{\mathrm{D1}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D2}}})}$: $M_{\mathrm{SFT}_{\mathrm{D2}}}$ に D_1 を chosen, D_0 を rejected として DPO を適用
- $M_{\mathrm{DPO}_{\mathrm{D0}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D1}}})}$: $M_{\mathrm{SFT}_{\mathrm{D2}}}$ に D_0 を chosen, D_1 を rejected として DPO を適用
- $M_{\mathrm{DPO}_{\mathrm{D1}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D1}}})}$: $M_{\mathrm{SFT}_{\mathrm{D2}}}$ に D_1 を chosen, D_0 を rejected として DPO を適用

学習した 9 個のモデルに対して, 以下の評価手法を用いて SFT や DPO の特性や影響範囲, 生成結果への影響を評価する.

- パラメータの重みからの評価

  L1 ノルム, L2 ノルム, Conflict Limited L2 ノルム

- 生成結果からの評価

  BLEU, ROUGE-L, BERT-Score, LLM-as-a-judge, 定性的な評価

ここで, SFT には Transformers Reinforcement Learning (TRL) ライブラリ [49] で提供されている SFTTrainer クラスを使用し, DPO においても同じく TRL で提供されている DPOTrainer クラスを使用した. また学習の際には効率化のため両手法ともに QLoRA の処理をしている. 表 5.1, 5.2, 5.3 に各手法のパラメータを示す. SFTTrainer, DPOTrainer ともに学習率とエポック数を揃えている.

また, 表 5.4 にテキスト生成の際のパラメータを示す.

表 5.1: QLoRA パラメータ

パラメータ	値
量子化サイズ	4 ビット
r	8
lora_alpha	128
target_modules	モデル内の線形層全て
lora_dropout	0.05

表 5.2: SFTTrainer パラメータ

パラメータ	値
epoch 数	3
バッチサイズ	2
最適化手法	Adam
初期学習率	1e-5
学習率スケジューラ	cosine

表 5.3: DPOTrainer パラメータ

パラメータ	値
epoch 数	3
バッチサイズ	2
最適化手法	Adam
初期学習率	1e-5
学習率スケジューラ	cosine
β	0.3

また, 学習の際 ChatHaruhi [42] を参考として, 推論の際にも用いるシステムプロンプト付きのプロンプトを学習データに含めた. さらに, 学習後のモデルの出力を安定させるために LLM の末尾に tokenizer の EOS (End Of Sentense) トークンを付与した. 上記の処理を施した学習データを示す.

実験 1 における処理後の学習データ

I want you to act like a young lady. She always behaves gracefully. I want you to respond and answer like her, using the tone, manner and vocabulary she would use. I would like you to keep your response to about one sentence in length and brief, about 30 words. Please respond to the following questions with the above instructions and information. 必ず日本語で応答してください。
question
{ クエリ }
{ 応答 }{EOS トークン }

また, 評価者 LLM は GPT-4o-mini を用いて以下のプロンプトでエージェントの口調を 10 段階でスコア化して評価する. プロンプトは Character-LLM のプロンプトを参考にした. ここでプロンプト内の {profile} はデータセットで定義されているペルソナを記述した文, {conversation_example} はデータセット内からランダムに抽出した会話例, {question}, {answer} は評価する会話のクエリと LLM の応答が入る.

表 5.4: テキスト生成の際のパラメータ

パラメータ	値
max_token	128
do_sample	True
temperature	0.1
top_p	1.0

評価者 LLM に与えるプロンプト

You will be given responses written by an AI assistant mimicing the character. Your task is to rate the performance of character using the specific criterion by following the evaluation steps. Below is the data of character who mimiced by assistant:
***
[Profile] {profile}
[Examples of Conversation] {conversation_example}
[Interactions]
[user]{question}
[assistant] {answer}
***
[Evaluation Criterion]
Tone (1-10): How well do the responses reflect the character's tone of voice?
[Evaluation Steps]
1. Review the profile and conversation examples to identify the manner of speaking or style (tone) of the original character.
2. Read through the interactions and examine the interactions and note the AI assistant's tone.
3. Compare the AI assistant's tone to the character's, based on the gathered information. Assess whether the response is natural and of appropriate length.
4. Use a 1–10 scale to rate how well the assistant's tone reflects the character's style. 1 means it does not reflect the character's tone at all, and 10 means it perfectly reflects the character's tone.
***
First, write out in a step by step manner your reasoning about the criterion to be sure that your conclusion is correct. Avoid simply stating the correct answers at the outset. Then print the score on its own line corresponding to the correct answer. At the end, repeat just the selected score again by itself on a new line. Please responce in Japanese.

5.2 実験 2

実験 1 で elyza/Llama-3-ELYZA-JP-8B としていたベースモデルを Qwen/Qwen2.5-7B-Instruct に変更し実験 1 と同じ条件で実験し, ベースモデルが異なる場合の影響を

確かめる.

5.3　実験 3

実験 1 で用いていたデータセット D_1, D_2 を D'_1, D'_2 に変更し, データセットが異なる場合の結果を確かめる. ロールプレイするペルソナがお嬢様から男子大学生に変わるため, システムプロンプトは以下のようになる.

実験 3 における処理後の学習データ

I want you to act like a young male college student . he always behaves bluntly. I want you to respond and answer like him, using the tone, manner and vocabulary he would use. I would like you to keep your response to about one sentence in length and brief, about 30 words. Please respond to the following questions with the above instructions and information. 必ず日本語で応答してください。
question
{ クエリ }
{ 応答 }{EOS トークン }

表 6.1: 実験 1 における (学習後のモデル, 学習前のモデル) の関係にある 2 モデル間の L1 ノルム, L2 ノルム

モデル組	L1 ノルム	L2 ノルム
$M_{\mathrm{SFT}_{\mathrm{D0}}}$, M_{base}	2.32498 $\times 10^5$	1.51147
$M_{\mathrm{SFT}_{\mathrm{D1}}}$, M_{base}	2.37030 $\times 10^5$	1.64791
$M_{\mathrm{SFT}_{\mathrm{D2}}}$, M_{base}	2.34274 $\times 10^5$	1.57759
$M_{\mathrm{DPO}_{\mathrm{D0}}}$, M_{base}	1.54711 $\times 10^5$	0.51305
$M_{\mathrm{DPO}_{\mathrm{D1}}}$, M_{base}	1.38068 $\times 10^5$	0.36422
$M_{\mathrm{DPO}_{\mathrm{D0}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D2}}})}$, M_{base}	3.08548 $\times 10^5$	2.67890
$M_{\mathrm{DPO}_{\mathrm{D1}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D2}}})}$, M_{base}	2.97963 $\times 10^5$	2.50216
$M_{\mathrm{DPO}_{\mathrm{D0}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D2}}})}$, $M_{\mathrm{SFT}_{\mathrm{D2}}}$	1.63760 $\times 10^5$	0.58680
$M_{\mathrm{DPO}_{\mathrm{D1}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D2}}})}$, $M_{\mathrm{SFT}_{\mathrm{D2}}}$	1.47138 $\times 10^5$	0.38904
$M_{\mathrm{DPO}_{\mathrm{D0}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D1}}})}$, M_{base}	3.12721 $\times 10^5$	2.75796
$M_{\mathrm{DPO}_{\mathrm{D1}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D1}}})}$, M_{base}	2.94477 $\times 10^5$	2.46993
$M_{\mathrm{DPO}_{\mathrm{D0}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D1}}})}$, $M_{\mathrm{SFT}_{\mathrm{D1}}}$	1.66718 $\times 10^5$	0.58686
$M_{\mathrm{DPO}_{\mathrm{D1}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D1}}})}$, $M_{\mathrm{SFT}_{\mathrm{D1}}}$	1.37648 $\times 10^5$	0.30320

6　結果と考察

以下, 各実験の結果とその考察を示す.

6.1　実験 1

まず, (学習後のモデル, 学習前のモデル) となっているモデル間の L1 ノルム, L2 ノルムを計算した. 表 6.1 に結果を示す. 表 6.1 から, 同じ学習率においても SFT と DPO ではベースモデルからの重みの変化の大きさが SFT の方が大きくなっていることがわかる. また, DPO においては D_1 よりも D_0 を chosen として学習した方がベースモデルからの重みの変化が大きくなっている.

ここからデータセット間のコンフリクト→データセットの違いがどの層に現れているか考察

ここからデータセット同じで手法違う→手法の重み変動によって同じ層に置いてどの差がでるか, どの層が

L1 301066.875 L2norm 2.5385672218268884

表 6.2: 実験 1 における SFT を施したモデルとベースモデルとのタスクベクトルにおける絶対値上位 20 % の要素の層ごとの L2 ノルムと平均値 (太字は上位 10 層)

	$M_{\mathrm{SFT}_{\mathrm{D0}}}$	$M_{\mathrm{SFT}_{\mathrm{D1}}}$	$M_{\mathrm{SFT}_{\mathrm{D2}}}$	$M_{\mathrm{DPO}_{\mathrm{D0}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D2}}})}$	$M_{\mathrm{DPO}_{\mathrm{D1}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D2}}})}$
model.layers.0	0.13545	0.14156	0.13148	0.37981	0.28800
model.layers.1	0.17689	0.16912	0.15699	0.40888	0.30776
model.layers.2	0.14191	0.15556	0.13645	0.40222	0.30098
model.layers.3	0.16408	0.15989	0.13740	0.38577	0.29911
model.layers.4	0.18212	0.17565	0.16597	0.40737	0.32798
model.layers.5	0.18007	0.18302	0.17449	0.42770	0.34036
model.layers.6	0.19416	0.20257	0.19586	0.43685	0.35986
model.layers.7	0.21560	0.23083	0.21880	0.46268	0.38550
model.layers.8	0.21011	0.23462	0.22258	0.46174	0.39446
model.layers.9	0.24736	0.25930	0.26324	0.47535	0.41182
model.layers.10	0.26771	0.27654	0.26115	**0.50501**	0.43276
model.layers.11	**0.29819**	0.30216	**0.33676**	**0.53699**	**0.46587**
model.layers.12	**0.28976**	0.31112	**0.30817**	**0.52805**	0.46332
model.layers.13	**0.30671**	**0.32759**	**0.32524**	**0.53282**	**0.47931**
model.layers.14	**0.34153**	**0.34064**	**0.32616**	**0.54224**	**0.48757**
model.layers.15	**0.32560**	**0.33556**	**0.33932**	**0.50631**	**0.46868**
model.layers.16	**0.30120**	0.29259	0.28485	0.47671	0.43870
model.layers.17	**0.28243**	0.27852	0.27994	0.47708	0.43391
model.layers.18	0.28112	**0.33444**	0.29786	0.49973	**0.47979**
model.layers.19	0.27045	0.29498	0.26845	0.48729	0.45291
model.layers.20	0.25059	0.28824	0.26117	0.47987	0.44701
model.layers.21	0.26333	0.29714	0.30765	0.48084	0.45368
model.layers.22	0.27892	**0.31896**	0.27064	0.48529	0.45956
model.layers.23	0.27091	**0.32325**	0.23775	0.50139	**0.47386**
model.layers.24	0.25485	0.30979	0.29704	0.48735	0.45864
model.layers.25	0.26817	0.30994	0.28377	**0.50307**	0.45556
model.layers.26	0.25588	0.30577	**0.31176**	0.49986	0.45944
model.layers.27	0.27243	**0.31467**	**0.34524**	0.49778	**0.46778**
model.layers.28	0.25018	0.28184	0.27093	0.45876	0.43019
model.layers.29	**0.31264**	**0.36474**	**0.36430**	**0.55003**	**0.50950**
model.layers.30	**0.36188**	**0.41735**	**0.36297**	**0.58687**	**0.55357**
model.layers.31	**0.44451**	**0.47415**	**0.45437**	**0.63152**	**0.59476**
average	0.25927	0.28163	0.26871	0.48448	0.43069

表 6.3: DPO を適用後モデルと DPO を適用前モデルとのタスクベクトルにおける絶対値上位 20 % の要素の層ごとの L2 ノルムと平均値 (太字は上位 10 層)

	$M_{\mathrm{DPO}_{\mathrm{D0}}}$	$M_{\mathrm{DPO}_{\mathrm{D1}}}$	$M_{\mathrm{DPO}_{\mathrm{D0}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D2}}})}$	$M_{\mathrm{DPO}_{\mathrm{D1}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D2}}})}$
model.layers.0	0.04658	0.02891	0.06931	0.03935
model.layers.1	0.09113	0.05179	0.10322	0.05133
model.layers.2	0.06173	0.02839	0.08102	0.03998
model.layers.3	0.05655	0.02698	0.07109	0.03736
model.layers.4	0.08034	0.03342	0.10395	0.05460
model.layers.5	0.08648	0.04637	0.10482	0.05957
model.layers.6	0.06359	0.03201	0.08481	0.04518
model.layers.7	0.07995	0.04024	0.09003	0.05345
model.layers.8	0.08667	0.04034	0.09903	0.05829
model.layers.9	0.09571	0.04434	0.10843	0.06085
model.layers.10	**0.09975**	0.04689	**0.12758**	0.06366
model.layers.11	**0.09983**	0.04667	**0.14091**	0.07193
model.layers.12	**0.10192**	0.04674	**0.12564**	0.05868
model.layers.13	**0.13155**	0.06933	**0.14789**	**0.08262**
model.layers.14	**0.13326**	**0.07868**	**0.14260**	**0.08359**
model.layers.15	0.08632	0.05243	0.09447	0.06469
model.layers.16	0.08678	0.06218	0.08482	0.07031
model.layers.17	0.06954	0.05620	**0.10984**	**0.08038**
model.layers.18	0.06498	0.05899	0.09383	**0.08320**
model.layers.19	0.09512	**0.07844**	0.08475	0.07519
model.layers.20	0.08128	0.06250	0.08439	0.06936
model.layers.21	0.08066	**0.08438**	0.08844	**0.08570**
model.layers.22	0.08561	0.06572	0.08020	0.06903
model.layers.23	0.07898	**0.09167**	0.08740	**0.09234**
model.layers.24	**0.11635**	**0.07806**	0.08260	**0.07736**
model.layers.25	**0.11432**	**0.08989**	**0.11243**	**0.08224**
model.layers.26	0.08975	**0.08621**	0.09168	**0.08319**
model.layers.27	**0.10897**	**0.08806**	0.09716	0.07230
model.layers.28	0.07286	0.07418	0.07420	0.06173
model.layers.29	**0.09733**	0.07399	**0.11804**	0.07090
model.layers.30	0.08976	**0.10200**	**0.11859**	**0.07834**
model.layers.31	**0.10088**	**0.08145**	**0.14313**	0.07536
average	0.08858	0.06086	0.10145	0.06725

表 6.4: 学習方法が同一でデータセットが異なる 2 モデル間の L1 ノルム, L2 ノルム

モデル	L1 ノルム	L2 ノルム
$M_{\mathrm{SFT}_{\mathrm{D0}}}$, $M_{\mathrm{SFT}_{\mathrm{D1}}}$	2.77361×10^5	2.20032
$M_{\mathrm{DPO}_{\mathrm{D0}}}$, $M_{\mathrm{DPO}_{\mathrm{D1}}}$	2.71417×10^5	2.59660
$M_{\mathrm{DPO}_{\mathrm{D0}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D2}}})}$, $M_{\mathrm{DPO}_{\mathrm{D1}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D2}}})}$	2.90157×10^5	2.96470
$M_{\mathrm{DPO}_{\mathrm{D0}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D1}}})}$, $M_{\mathrm{DPO}_{\mathrm{D1}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D1}}})}$	2.81640×10^5	2.75609

L1: 294705.625

L2: 2.484670393554694

ここからモデル回帰に関する考察

6.2 実験 2

(学習後のモデル, 学習前のモデル) となっているモデル間の L1 ノルム, L2 ノルムを計算した. 表 6.12 に結果を示す.

6.3 実験 3

表 6.5: $M_{\mathrm{SFT}_{\mathrm{D0}}}$, $M_{\mathrm{SFT}_{\mathrm{D1}}}$ 間の コンフリクト 数, コンフリクト率, Conflict Limited L2

Key	コンフリクト数	コンフリクト率	Conflict Limited L2
model.layers.0	120	5.5018e-07	0.00438
model.layers.1	583	2.6729e-06	0.01011
model.layers.2	229	1.0499e-06	0.00614
model.layers.3	303	1.3892e-06	0.00677
model.layers.4	336	1.5405e-06	0.00733
model.layers.5	283	1.2975e-06	0.00687
model.layers.6	307	1.4075e-06	0.00719
model.layers.7	613	2.8105e-06	0.00987
model.layers.8	483	2.2145e-06	0.00887
model.layers.9	806	3.6953e-06	0.01118
model.layers.10	1144	5.245e-06	0.01350
model.layers.11	1848	8.4727e-06	0.01696
model.layers.12	1434	6.5746e-06	0.01537
model.layers.13	3568	1.6359e-05	0.02396
model.layers.14	3842	1.7615e-05	0.02445
model.layers.15	**6688**	**3.0663e-05**	**0.03332**
model.layers.16	4836	2.2172e-05	0.02809
model.layers.17	**5096**	**2.3364e-05**	**0.02892**
model.layers.18	**10577**	**4.8493e-05**	**0.04275**
model.layers.19	**6446**	**2.9554e-05**	**0.03305**
model.layers.20	3707	1.6996e-05	0.02450
model.layers.21	4067	1.8646e-05	0.02529
model.layers.22	**8805**	**4.0369e-05**	**0.03798**
model.layers.23	**9957**	**4.5651e-05**	**0.04076**
model.layers.24	**5262**	**2.4125e-05**	**0.02951**
model.layers.25	**5635**	**2.5835e-05**	**0.03026**
model.layers.26	4611	2.1141e-05	0.02858
model.layers.27	4707	2.1581e-05	0.02832
model.layers.28	3172	1.4543e-05	0.02295
model.layers.29	3981	1.8252e-05	0.02548
model.layers.30	**14061**	**6.4467e-05**	**0.04973**
model.layers.31	**8320**	**3.8146e-05**	**0.03846**
all layer	125827	1.8028e-05	0.14515

表 6.6: $M_{\mathrm{DPO}_{\mathrm{D0}}}$, $M_{\mathrm{DPO}_{\mathrm{D1}}}$ 間の コンフリクト 数, コンフリクト率, Conflict Limited L2

モデル	コンフリクト数	コンフリクト率	Conflict Limited L2
model.layers.0	2656	1.2177e-05	0.02065
model.layers.1	13560	6.217e-05	**0.05236**
model.layers.2	3158	1.4479e-05	0.02214
model.layers.3	3096	1.4195e-05	0.02162
model.layers.4	4698	2.1539e-05	0.02773
model.layers.5	9219	4.2267e-05	0.03799
model.layers.6	3417	1.5666e-05	0.02306
model.layers.7	5167	2.369e-05	0.02768
model.layers.8	5215	2.391e-05	0.02858
model.layers.9	7206	3.3038e-05	0.03367
model.layers.10	8399	3.8508e-05	0.03652
model.layers.11	7365	3.3767e-05	0.03420
model.layers.12	7945	3.6426e-05	0.03501
model.layers.13	**20207**	**9.2645e-05**	**0.05606**
model.layers.14	**25755**	**1.1808e-04**	**0.06372**
model.layers.15	10848	4.9736e-05	0.04373
model.layers.16	12054	5.5265e-05	0.04252
model.layers.17	9178	4.2079e-05	0.03825
model.layers.18	7346	3.368e-05	0.03406
model.layers.19	**15293**	**7.0115e-05**	**0.04918**
model.layers.20	6708	3.0755e-05	0.03247
model.layers.21	**18174**	**8.3324e-05**	**0.05371**
model.layers.22	10262	4.7049e-05	0.04033
model.layers.23	**15042**	**6.8965e-05**	0.04850
model.layers.24	**20134**	**9.231e-05**	**0.05802**
model.layers.25	**27940**	**1.281e-4**	**0.06843**
model.layers.26	14857	6.8116e-05	0.04873
model.layers.27	**17719**	**8.1238e-05**	**0.05399**
model.layers.28	9048	4.1483e-05	0.03819
model.layers.29	**19183**	**8.795e-05**	**0.05525**
model.layers.30	13767	6.3119e-05	0.04867
model.layers.31	**25498**	**1.169e-04**	**0.06452**
total	380114	5.44608e-05	0.24815

表 6.7: $M_{\mathrm{DPO}_{\mathrm{D0}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D2}}})}$, $M_{\mathrm{DPO}_{\mathrm{D1}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D2}}})}$ 間の M_{base} を基準とした コンフリクト 数, コンフリクト率, Conflict Limited L2

モデル	コンフリクト数	コンフリクト率	Conflict Limited L2
model.layers.0	5652	2.5913e-05	0.03020
model.layers.1	10876	4.9864e-05	0.04877
model.layers.2	7108	3.2589e-05	0.03386
model.layers.3	6104	2.7986e-05	0.03179
model.layers.4	12350	5.6622e-05	0.04613
model.layers.5	13841	6.3458e-05	0.04752
model.layers.6	6892	3.1598e-05	0.03377
model.layers.7	8221	3.7692e-05	0.03631
model.layers.8	8760	4.0163e-05	0.03708
model.layers.9	12416	5.6925e-05	0.04531
model.layers.10	14859	6.8126e-05	0.04963
model.layers.11	**16570**	**7.597e-05**	**0.05141**
model.layers.12	12130	5.5614e-05	0.04429
model.layers.13	**24772**	**1.1357e-04**	**0.06408**
model.layers.14	**34037**	**1.5605e-04**	**0.07663**
model.layers.15	11218	5.1432e-05	0.04329
model.layers.16	9626	4.4133e-05	0.04008
model.layers.17	**20242**	**9.2806e-05**	**0.05833**
model.layers.18	**20111**	**9.2205e-05**	**0.05958**
model.layers.19	8751	4.0122e-05	0.03765
model.layers.20	7773	3.5638e-05	0.03576
model.layers.21	15607	7.1555e-05	0.05087
model.layers.22	8527	3.9095e-05	0.03676
model.layers.23	**18053**	**8.2769e-05**	**0.05459**
model.layers.24	12034	5.5173e-05	0.04450
model.layers.25	**17382**	**7.9693e-05**	**0.05438**
model.layers.26	14684	6.7323e-05	0.05035
model.layers.27	12345	5.6599e-05	0.04561
model.layers.28	5465	2.5056e-05	0.03044
model.layers.29	**15762**	**7.2266e-05**	**0.05139**
model.layers.30	**22541**	**1.0335e-04**	**0.06156**
model.layers.31	**20835**	**9.5524e-05**	**0.07928**
total	435544	6.24026e-05	0.27570

表 6.8: $M_{\mathrm{DPO}_{\mathrm{D0}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D2}}})}$, $M_{\mathrm{DPO}_{\mathrm{D1}}(\mathrm{M}_{\mathrm{SFT}_{\mathrm{D2}}})}$ 間の $M_{\mathrm{SFT}_{\mathrm{D2}}}$ を基準とした コンフリクト 数, コンフリクト 率, Conflict Limited L2

モデル	コンフリクト数	コンフリクト率	Conflict Limited L2
model.layers.0	7386	3.3863e-05	0.03355
model.layers.1	12373	5.6728e-05	0.05097
model.layers.2	9247	4.2396e-05	0.03752
model.layers.3	8052	3.6917e-05	0.03539
model.layers.4	15502	7.1074e-05	0.05028
model.layers.5	18367	8.4209e-05	0.05313
model.layers.6	9177	4.2075e-05	0.03774
model.layers.7	11270	5.1671e-05	0.04125
model.layers.8	11954	5.4807e-05	0.04214
model.layers.9	16053	7.36e-05	0.05017
model.layers.10	19304	8.8505e-05	0.05516
model.layers.11	**21718**	**9.9573e-05**	0.05744
model.layers.12	15695	7.1958e-05	0.04925
model.layers.13	**31807**	**1.4583e-04**	**0.07084**
model.layers.14	**43773**	**2.0069e-04**	**0.08458**
model.layers.15	15294	7.012e-05	0.04890
model.layers.16	13106	6.0088e-05	0.04519
model.layers.17	**27683**	**1.2692e-04**	**0.06593**
model.layers.18	**26733**	**1.2257e-04**	**0.06617**
model.layers.19	12429	5.6984e-05	0.04333
model.layers.20	11182	5.1267e-05	0.04132
model.layers.21	21630	9.9169e-05	**0.05778**
model.layers.22	12619	5.7856e-05	0.04316
model.layers.23	**25874**	**1.1863e-04**	**0.06288**
model.layers.24	17509	8.0275e-05	0.05169
model.layers.25	**23398**	**1.0728e-04**	**0.06109**
model.layers.26	20398	9.3521e-05	0.05715
model.layers.27	17220	7.895e-05	0.05193
model.layers.28	7793	3.5729e-05	0.03491
model.layers.29	**21824**	**1.0006e-04**	**0.05848**
model.layers.30	**33161**	**1.5204e-04**	**0.07162**
total	587498	8.4174e-05	0.30516

表 6.9: $M_{\mathrm{DPO}_{\mathrm{D0}}}$, $M_{\mathrm{SFT}_{\mathrm{D0}}}$ 間 コンフリクト 数, コンフリクト率, Conflict Limited L2

モデル	コンフリクト数	コンフリクト率	Conflict Limited L2
model.layers.0	241	1.1049e-06	0.00658
model.layers.1	**4028**	**1.8468e-05**	**0.03061**
model.layers.2	255	1.1691e-06	0.00656
model.layers.3	268	1.2287e-06	0.00668
model.layers.4	569	2.6088e-06	0.01000
model.layers.5	654	2.9985e-06	0.01045
model.layers.6	311	1.4259e-06	0.00730
model.layers.7	1033	4.7361e-06	0.01287
model.layers.8	1007	4.6169e-06	0.01276
model.layers.9	1617	7.4136e-06	0.01631
model.layers.10	2203	1.01e-05	0.01907
model.layers.11	2465	1.1302e-05	0.02085
model.layers.12	2531	1.1604e-05	0.02064
model.layers.13	**3843**	**1.7619e-05**	**0.02531**
model.layers.14	**7372**	**3.3799e-05**	**0.03532**
model.layers.15	**3872**	**1.7752e-05**	**0.02637**
model.layers.16	2680	1.2287e-05	0.02140
model.layers.17	1807	8.2847e-06	0.01771
model.layers.18	1199	5.4972e-06	0.01427
model.layers.19	3035	1.3915e-05	0.02374
model.layers.20	1213	5.5614e-06	0.01466
model.layers.21	1520	6.9689e-06	0.01640
model.layers.22	1589	7.2852e-06	0.01671
model.layers.23	1417	6.4967e-06	0.01576
model.layers.24	**4121**	**1.8894e-05**	**0.02631**
model.layers.25	**3468**	**1.59e-05**	**0.02480**
model.layers.26	2565	1.176e-05	0.02159
model.layers.27	**3740**	**1.7147e-05**	**0.02603**
model.layers.28	973	4.461e-06	0.01293
model.layers.29	**3227**	**1.4795e-05**	**0.02396**
model.layers.30	**4298**	**1.9705e-05**	**0.02846**
model.layers.31	**9154**	**4.1969e-05**	**0.04200**
total	78275	1.1215e-05	0.11854

表 6.10: $M_{\mathrm{DPO}_{\mathrm{D1}}}$, $M_{\mathrm{SFT}_{\mathrm{D1}}}$ 間 コンフリクト 数, コンフリクト率, Conflict Limited L2

モデル	コンフリクト数	コンフリクト率	Conflict Limited L2
model.layers.0	241	2.8884e-07	0.00347
model.layers.1	**4028**	2.5904e-06	0.01059
model.layers.2	255	3.7595e-07	0.00384
model.layers.3	268	2.4758e-07	0.00307
model.layers.4	569	6.1436e-07	0.00480
model.layers.5	654	1.1829e-06	0.00677
model.layers.6	311	7.1064e-07	0.00530
model.layers.7	1033	1.8202e-06	0.00834
model.layers.8	1007	1.7193e-06	0.00798
model.layers.9	1617	2.8747e-06	0.01057
model.layers.10	2203	3.6999e-06	0.01194
model.layers.11	2465	4.9287e-06	0.01399
model.layers.12	2531	5.3046e-06	0.01453
model.layers.13	**3843**	7.5558e-06	0.01687
model.layers.14	**7372**	**1.4016e-05**	**0.02311**
model.layers.15	**3872**	6.7351e-06	0.01697
model.layers.16	2680	5.5018e-06	0.01430
model.layers.17	1807	4.2409e-06	0.01258
model.layers.18	1199	7.7391e-06	0.01745
model.layers.19	3035	1.0421e-05	**0.02052**
model.layers.20	1213	6.162e-06	0.01546
model.layers.21	1520	9.1971e-06	0.01843
model.layers.22	1589	**1.0481e-05**	0.02014
model.layers.23	1417	**1.5928e-05**	**0.02483**
model.layers.24	**4121**	**1.4529e-05**	**0.02394**
model.layers.25	**3468**	**1.0953e-05**	**0.02061**
model.layers.26	2565	**1.2897e-05**	**0.02241**
model.layers.27	**3740**	**1.9009e-05**	**0.02776**
model.layers.28	973	9.6006e-06	0.01972
model.layers.29	**3227**	**1.5964e-05**	**0.02541**
model.layers.30	**4298**	**3.1658e-05**	**0.03634**
model.layers.31	**9154**	**3.119e-05**	**0.03659**
total	58920	8.44176e-06	0.10392

表 6.11: ($M_{\mathrm{DPO_{D0}(M_{SFT_{D1}})}}$, M_{base}), ($M_{\mathrm{DPO_{D0}(M_{SFT_{D1}})}}$, $M_{\mathrm{SFT_{D1}}}$), ($M_{\mathrm{DPO_{D0}(M_{SFT_{D1}})}}$, $M_{\mathrm{DPO_{D0}}}$) 間の L1 ノルム, L2 ノルム

モデル	L1 ノルム	L2 ノルム
$M_{\mathrm{DPO_{D0}(M_{SFT_{D1}})}}$, $M_{\mathrm{SFT_{D1}}}$	1.66718×10^5	0.58686
$M_{\mathrm{DPO_{D0}(M_{SFT_{D1}})}}$, M_{base}	3.12721×10^5	2.75796
$M_{\mathrm{DPO_{D0}(M_{SFT_{D1}})}}$, $M_{\mathrm{DPO_{D0}}}$	3.45665×10^5	3.32855
$M_{\mathrm{DPO_{D0}(M_{SFT_{D1}})}}$, $M_{\mathrm{SFT_{D0}}}$	3.27395×10^5	3.05515

表 6.12: 実験 2 における (学習後のモデル, 学習前のモデル) の関係にある 2 モデル間の L1 ノルム, L2 ノルム

モデル組	L1 ノルム	L2 ノルム
$M_{\mathrm{SFT_{D0}}}$, M_{base}	2.65476×10^5	4.66679
$M_{\mathrm{SFT_{D1}}}$, M_{base}	2.52975×10^5	4.42202
$M_{\mathrm{SFT_{D2}}}$, M_{base}	2.53258×10^5	4.02821
$M_{\mathrm{DPO_{D0}}}$, M_{base}	1.37400×10^5	3.98816
$M_{\mathrm{DPO_{D1}}}$, M_{base}	1.41034×10^5	3.49920
$M_{\mathrm{DPO_{D0}(M_{SFT_{D2}})}}$, M_{base}	3.29224×10^5	6.63510
$M_{\mathrm{DPO_{D1}(M_{SFT_{D2}})}}$, M_{base}	3.23261×10^5	5.80873
$M_{\mathrm{DPO_{D0}(M_{SFT_{D2}})}}$, $M_{\mathrm{SFT_{D2}}}$	1.61470×10^5	4.64364
$M_{\mathrm{DPO_{D1}(M_{SFT_{D2}})}}$, $M_{\mathrm{SFT_{D2}}}$	1.53017×10^5	3.36994
$M_{\mathrm{DPO_{D0}(M_{SFT_{D1}})}}$, M_{base}	3.29579×10^5	6.86016
$M_{\mathrm{DPO_{D1}(M_{SFT_{D1}})}}$, M_{base}	3.26070×10^5	6.32263
$M_{\mathrm{DPO_{D0}(M_{SFT_{D1}})}}$, $M_{\mathrm{SFT_{D1}}}$	1.63094×10^5	4.61360
$M_{\mathrm{DPO_{D1}(M_{SFT_{D1}})}}$, $M_{\mathrm{SFT_{D1}}}$	157729×10^5	3.75747

表 6.12 から, 同一データセットにおいては SFT と DPO では重みの変化量が大きくて

7　まとめと今後の課題

謝辞

あとで先輩の参考に書く.

2025 年 2 月 21 日

参考文献

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, Vol. abs/1706.03762, , 2017.

[2] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[3] Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Bing Yin, and Xia Hu. Harnessing the power of llms in practice: A survey on chatgpt and beyond, 2023.

[4] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2024.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, Vol. 9, No. 8, p. 1735–1780, nov 1997.

[6] F.A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: continual prediction with lstm. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, Vol. 2, pp. 850–855 vol.2, 1999.

[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[9] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation, 2015.

[10] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, Vol. abs/1810.04805, , 2018.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, Vol. abs/2010.11929, , 2020.

[14] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, Vol. abs/1910.10683, , 2019.

[15] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, Vol. abs/2005.14165, , 2020.

[16] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloun-

dou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher,

Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024.

[17] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[18] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[19] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo

Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh,

Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Han-

nah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun

Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024.

[20] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *CoRR*, Vol. abs/2009.03300, , 2020.

[21] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

[22] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, Vol. abs/2110.14168, , 2021.

[23] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji

Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report, 2023.

[24] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

[25] Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report, 2024.

[26] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement, 2024.

[27] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[28] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications, 2024.

[29] Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and

Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution, 2023.

[30] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

[31] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, Vol. abs/1707.06347, , 2017.

[32] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, Vol. 22, No. 1, pp. 79–86, 1951.

[33] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *CoRR*, Vol. abs/2106.09685, , 2021.

[34] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023.

[35] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[36] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with BERT. *CoRR*, Vol. abs/1904.09675, , 2019.

[37] Chin-Yew Lin and Eduard Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 150–157, 2003.

[38] Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan T. McDonald. On faithfulness and factuality in abstractive summarization. *CoRR*, Vol. abs/2005.00661, , 2020.

[39] Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, Kai Shu, Lu Cheng, and Huan Liu. From generation to judgment: Opportunities and challenges of llm-as-a-judge, 2025.

[40] LLM-jp, :, Akiko Aizawa, Eiji Aramaki, Bowen Chen, Fei Cheng, Hiroyuki Deguchi, Rintaro Enomoto, Kazuki Fujii, Kensuke Fukumoto, Takuya Fukushima, Namgi Han, Yuto Harada, Chikara Hashimoto, Tatsuya Hiraoka, Shohei Hisada, Sosuke Hosokawa, Lu Jie, Keisuke Kamata, Teruhito Kanazawa, Hiroki Kanezashi, Hiroshi Kataoka, Satoru Katsumata, Daisuke Kawahara, Seiya Kawano, Atsushi Keyaki, Keisuke Kiryu, Hirokazu Kiyomaru, Takashi Kodama, Takahiro Kubo, Yohei Kuga, Ryoma Kumon, Shuhei Kurita, Sadao Kurohashi, Conglong Li, Taiki Maekawa, Hiroshi Matsuda, Yusuke Miyao, Kentaro Mizuki, Sakae Mizuki, Yugo Murawaki, Akim Mousterou, Ryo Nakamura, Taishi Nakamura, Kouta Nakayama, Tomoka Nakazato, Takuro Niitsuma, Jiro Nishitoba, Yusuke Oda, Hayato Ogawa, Takumi Okamoto, Naoaki Okazaki, Yohei Oseki, Shintaro Ozaki, Koki Ryu, Rafal Rzepka, Keisuke Sakaguchi, Shota Sasaki, Satoshi Sekine, Kohei Suda, Saku Sugawara, Issa Sugiura, Hiroaki Sugiyama, Hisami Suzuki, Jun Suzuki, Toyotaro Suzumura, Kensuke Tachibana, Yu Takagi, Kyosuke Takami, Koichi Takeda, Masashi Takeshita, Masahiro Tanaka, Kenjiro Taura, Arseny Tolmachev, Nobuhiro Ueda, Zhen Wan, Shuntaro Yada, Sakiko Yahata, Yuya Yamamoto, Yusuke Yamauchi, Hitomi Yanaka, Rio Yokota, and Koichiro Yoshino. Llm-jp: A cross-organizational project for the research and development of fully open japanese llms, 2024.

[41] Yunfan Shao, Linyang Li, Junqi Dai, and Xipeng Qiu. Character-llm: A trainable agent for role-playing, 2023.

[42] 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.

[43] Zekun Moore Wang, Zhongyuan Peng, Haoran Que, Jiaheng Liu, Wangchunshu Zhou, Yuhan Wu, Hongcheng Guo, Ruitong Gan, Zehao Ni, Jian Yang, Man Zhang, Zhaoxiang Zhang, Wanli Ouyang, Ke Xu, Stephen W. Huang, Jie Fu, and Junran Peng. Rolellm: Benchmarking, eliciting, and enhancing role-playing abilities of large language models, 2024.

[44] Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. What do you learn from context? probing for sentence structure in contextualized word representations, 2019.

[45] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors, 2018.

[46] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.

[47] Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline, 2019.

[48] Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models, 2023.

[49] Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`, 2020.