

LLM の生成・翻訳による指示・応答データセット構築

加藤佑一 中田康太

(株) 東芝 研究開発センター

yuichi12.kato@toshiba.co.jp kouta.nakata@toshiba.co.jp

Abstract

高品質な日本語の指示・応答データセットを Large Language Model (LLM) による低コストな構築を目指し、英語での対話に最適化された Llama-2-70B-chat で英語の指示・応答データセットを生成し、それを Llama-2-70B で日本語訳した。本データセットと既存データセットのそれぞれで Swallow-7b をファインチューニングし、日本語 Vicuna QA ベンチマークで GPT-4 により比較評価した。その結果、前者のモデルが約 75% の勝率を達成した。さらに、OpenAI の text-davinci-003 との比較でも 60% 以上の勝率に達した。以上から、LLM による生成・翻訳による指示・応答データセット生成の有効性が示された。

1 はじめに

Large Language Model (LLM) は、 $\mathcal{O}(1 - 1000 \text{ B})$ のパラメータを持つ大規模なモデルを自己回帰による自己教師あり学習によって、 $\mathcal{O}(100 - 1000 \text{ B})$ トークン数におよぶ大量のテキストデータで学習することで、既存の 1 モデル 1 タスクを学習する方式とは異なり、事例をいくつか例示するだけでも多様な後続タスクに汎化できるため、自然言語処理に対してパラダイムシフトを起こした。大量のデータは、主に Web クロールや本などから集められているため、それらで学習したのみの LLM は、指示に対してしばしば人間の意図とは異なる回答をする。そのため、QA タスクなどを集めた指示・応答データセットにより、事前学習した LLM をファインチューニングすることで、人間の意図に沿って回答するように LLM をアライメントさせる必要がある。しかし、大規模な指示・応答データセットの構築を手で行うには、金銭・時間コストが大きい。本研究では、主に英語のテキストデータで訓練された LLM である Llama 2 [1] を用いて、英語の指示・応答データを生成し、それを日本語訳することで、日本語の指示・応答データセット構築が可能か検証した。

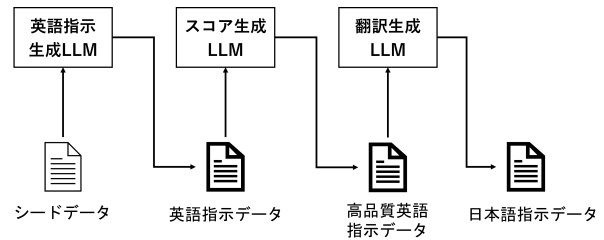


Figure 1 指示・応答データセット構築のフロー。

2 関連研究

これまでに LLM により指示・応答データを生成する研究は、主に英語で実施されてきた。Self-instruct [2] では、人間が設定したシードとなるタスクから、GPT-3 [3] により指示・応答データセットを作成した。本データセットで訓練した GPT-3 は、指示に対して、人手で作成したデータセットで訓練した同モデルと人間の評価において同程度のパフォーマンスを示した。Alpaca [4] は、OpenAI の text-davinci-003 を用いて、Self-instruct と同様の方法で指示・応答データを低コストで作成し、そのデータで LLaMA-7b [5] を訓練すると、text-davinci-003 と類似した回答が可能になると示した。LIMA [6] は、利用可能な指示・応答データの全てを用いるよりも厳選したデータのみで訓練した方がモデルのパフォーマンスが高かったことから、指示・応答データによるチューニングは、データ量よりも多様性と品質が重要であると示した。

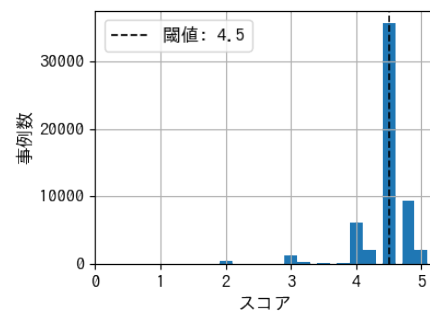


Figure 2 生成データのスコア分布。

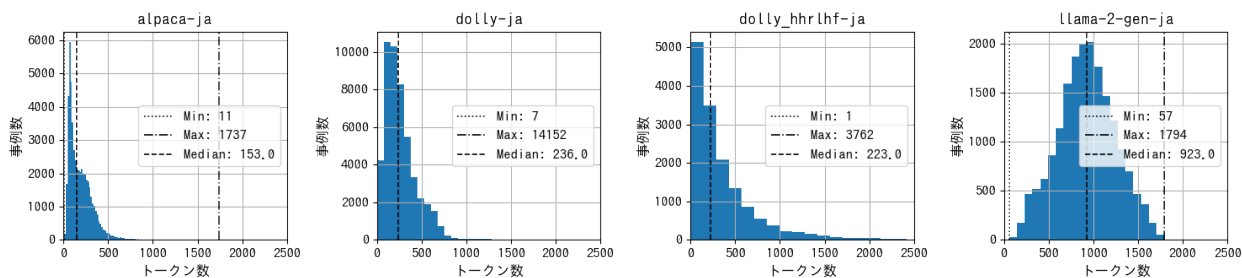


Figure 3 各指示・応答データセットに含まれる文章のトークン数の分布。

Self-alignment [7] は、シードデータから始めて、指示・応答データの生成と生成データによるモデルの訓練を繰り返すことで、モデルのパフォーマンスが向上することを明らかにした。生成の際には、データ品質をモデル自身に判定させ、高スコアなデータのみ用いることでデータ量にもパフォーマンスがスケールすると示した。

3 指示・応答データセット構築

本章では、LLM による日本語の指示・応答データセット構築の方法および構築したデータセットの内容・性質について説明する。

3.1 方法

指示・応答データセット構築方法の全体フローを図 1 に示した。本手法では、ある言語でのシードデータ、指示・応答を生成するモデル（指示生成モデル）、生成データに対してスコアを付けるモデル（スコア生成モデル）、およびターゲット言語への翻訳用のモデル（翻訳生成モデル）を用意する。シードデータを指示生成モデルに入力し、指示生成モデルが得意とする言語で新たな指示・応答データ（マルチターンの会話）を生成し、生成されたデータの品質をスコア生成モデルをスコアリングし、高いスコアのデータを翻訳生成モデルで翻訳することで、ターゲット言語での指示・応答データセットを構築する。シードデータとしては、OASST1 [8] から高品質な会話データを集めた Openassistant-guanaco [9] より英語データを 3500 個抽出して用いた。指示生成モデルとスコア生成モデルとしては、英語での対話用に最適化された Llama-2-70B-Chat を用いた。翻訳生成モデルには、Llama-2-70B を用いた。Self-alignment [7] と同様に、高品質のデータセットを構築するためにスコアリングを実施した。具体的には、スコア生成モデルには生成データの質を 5 段階で判定させて、スコアが > 4.5 であったデータの

みを抽出した。図 2 に生成したデータのスコア分布を示した。生成データの 8 割以上のスコアが > 4.5 であったことから、スコアを自身で与えているものの Llama-2-70B-chat は指示生成モデルとして優れていると示唆される。生成データの中には、繰り返しが生じて文章として体をなしていないものや、最初の質問文が重複しているもの、文字数が極端に短いものが含まれていたため、それらは前処理により取り除いた。翻訳生成でも繰り返しの文章が生成されることがしばしばあったため、そのような文章もデータセットから取り除いた。前処理も含めて、「英語指示データ」、「高品質英語指示データ」、「日本語指示データ」の各段階で、データセット数はそれぞれ 57000 → 26000 → 18000 となった。「高品質英語指示データ」の段階でデータ数が半分以下になっているのは、重複した質問が多くあったためである。

3.2 生成されたデータセット

生成されたデータの一部を付録 A の表 4 に示した。Llama-2-70B-chat で生成した英語データは、人間と AI アシスタントの会話を模擬できた。Llama-2-70B による翻訳も、日本語として完璧ではないがそれなり品質で翻訳できた。構築したデータセットの特性を見るため、本データセットと既存

Table 1 ファインチューニングのハイパーパラメータ

Batch size	32
Learning rate	2×10^{-5}
Weight decay	0.001
Warmup Ratio	0.03
Dropout	0.1
LoRA r	16
LoRA alpha	32

Table 2 推論時のサンプリングパラメータ

Temperature	0.7
top-p	1.0
top-k	50
num_beams	4
repetition_number	5

Table 3 日本語 Vicuna QA ベンチマークの各モデルの出力例。

質問: 医療サービスの質と効率を向上させるために、人工知能を活用できるシナリオを記述してください。

llama-2-gen (Ours):

もちろんです。人工知能を活用できるシナリオの1つは、電子カルテシステムへの自然言語処理技術の統合です。これにより、医師や他の医療従事者は、患者の症状や病歴などの情報を簡単に入力できます。この情報は、機械学習アルゴリズムによって分析され、病気の診断や治療計画の開発に役立つ可能性があります。

もう1つのシナリオは、チャットボットまたはバーチャルアシスタントの使用です。これらのエージェントは、患者からの質問に回答したり、健康関連のトピックに関するガイダンスを提供したりすることができます。これにより、患者は必要に応じていつでもサポートを受けることができます。

別のシナリオは、画像認識技術の使用です。これにより、放射線科医やその他の専門家は、コンピュータービジョンアルゴリズムを使用して、胸部レントゲン写真や超音波スキャンなどの検査結果を分析できます。これにより、彼らは正確で迅速な診断を行うことができます。

最後に、音声認識技術の使用も考えられます。これにより、看護師やその他の医療従事者は、ベッドサイドで患者と会話することができます。これにより、彼らは患者の感情的および精神的状態を理解し、それに応じて対応することができます。

全体的に見れば、人工知能の応用は、医療サービスの質と効率を大幅に向上させる可能性があります。しかし、注意すべき重要な考慮事項がいくつかあります。例えば、プライバシーとデータセキュリティの懸念があります。さらに、一部の分野では、人間の介入が依然として不可欠である可能性があります。

alpaca-ja:

人工知能は、患者の健康記録やその他のデータから洞察を得て、医師がより迅速かつ正確な診断を行えるようにすることで、医療サービスの質と効率を向上させることができます。また、機械学習アルゴリズムを使用して、個々の患者に合わせた治療計画を作成し、最適な結果を提供することもできます。

dolly-ja:

1. 電子カルテやその他の医療記録からデータを抽出し、機械学習アルゴリズムで分析します。これにより、特定の患者がどのようなケアを必要としているかを予測することができます。

2. 過去の治療結果をもとに、最適な治療計画を提案する。

3. 病院内外のさまざまな情報源から収集したデータを使用して、臨床的意思決定支援システムを開発する。

4. 自動化されたチャットボットを使って、一般的な健康上の質問に答える。

5. ウェアラブル技術を利用して、患者のバイタルサインを継続的に監視する。

dolly_hhrlhf-ja:

確かに、ここにいくつかのアイデアがあります。1つ目は、機械学習アルゴリズムを使用して、特定の種類の病気である可能性が最も高い患者を予測し、医師がより正確な診断を下すのに役立つことです。2つ目は、自然言語処理技術を使用して、電子健康記録から重要な情報を抽出することです。3つ目は、コンピューター支援設計ツールを使用して、病院やその他の医療施設のレイアウトを最適化することです。4つ目は、ビッグデータ分析を使用して、ヘルスケアシステム内の潜在的な改善領域を見つけることです。

の指示・応答データセットの分布を図3に示した。既存データセットとして、Alpaca [4]、Dolly [10]、Dolly-hhrlhf [11] を日本語訳 [12, 13, 14] したものを示した。Llama-2 のトークナイザーでトークン化してからトークン数を数えている。既存データセットの分布はどれも指数関数的であり、短い文章が多く含まれると分かる。一方で、構築したデータセットは、ガウス分布の形をしており、長文が多いと窺える。各データセットの合計トークン数は、Alpaca、Dolly、Dolly-hhrlhf、および構築した日本語の指示・応答データセット (Llama-2-gen) のそれぞれ、10 M、13 M、6 M、17 M であり、データセットのサイズは、既存のものと同様以上となった。

4 構築データセット評価

構築した日本語の指示・応答データセットの評価のため、本データセットと既存のもの [12, 13, 14] で、LLM をそれぞれファインチューニングし、出力を比較評価した。本章では、ファインチューニング方法、評価方法、および評価結果について説明す

る。LLM としては、Llama-2-7b を日本語で継続事前学習した Swallow-7b (SW-7b) [15] を用いた。

4.1 ファインチューニング方法

パラメータ数が大きな LLM の全パラメータをチューニングして保存するには、大きな GPU メモリと大きなストレージが必要となる。本実験では小さな GPU メモリと小さなストレージでも訓練の実行と保存が可能な LoRA [16] を用いた。LoRA は、全パラメータを更新するのではなく、既存の重み ($W_0 \in \mathbb{R}^{d \times k}$) は固定し、低ランクな2層の線形行列 ($A \in \mathbb{R}^{r \times k}, B \in \mathbb{R}^{d \times r}$) を挟み、それらのパラメータ更新することでモデルを最適化する。学習後の重みは、 $W = W_0 + BA$ で得られる。本実験では、線形層の全てに LoRA 層を挿入した。訓練時のハイパーパラメータを表1に示した。後続タスクの評価としてオープンエンドな QA を用いるため、構築データセットからは1ターン目の会話を抽出し、Alpaca [4] の指示・応答形式で訓練した。他のデータセットでも同様の指示・応答形式で訓練を実施した。

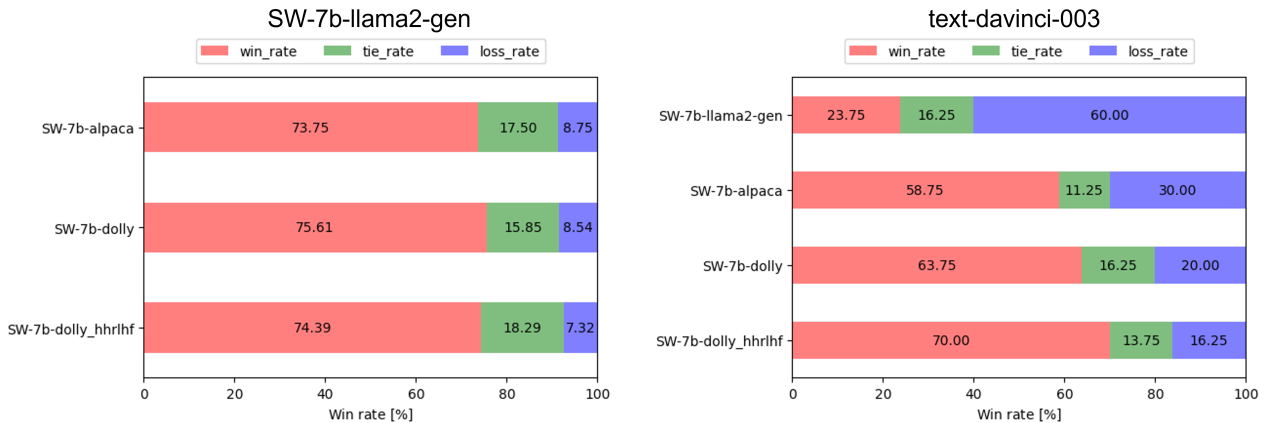


Figure 4 日本語 Vicuna QA ベンチマークでの (左) SW-7b を構築した指示・応答データセットでファインチューニングしたモデル「SW-7b-llama2-gen」と既存データセットでのモデルとの対による比較、および (右) text-davinci-003 と前記モデルらとの対による比較。それぞれ「SW-7b-llama2-gen」と text-davinci-003 の勝率を示している。

4.2 評価方法

各指示・応答データセットで訓練したモデルを比較評価するため、オープンエンドな QA タスクとして日本語 Vicuna QA ベンチマーク [17] を用いた。Vicuna QA ベンチマーク [18] は、フェルミ問題、ロールプレイの脚本、コーディング、数学問題など、多様な 80 個の質問に対するモデルの解答を、GPT-4 に有用性、関連性、正確性、詳細度合いに基づいて評価させることでモデルの性能を評価する。日本語 Vicuna QA ベンチマークは、それを日本語に訳したものである。構築したデータセットおよび既存の日本語データセットのそれぞれで SW-7B を訓練し、日本語 Vicuna QA ベンチマークに対する各出力について、GPT-4 を用いて比較評価させた。推論時に用いたサンプリングのパラメータを表 2 に示した。その結果を図 4 (左) に示した。本データセットでファインチューニングしたモデルの勝率が、他データセットに対して約 75% に達した。本要因を調べるために、各モデルの出力を表 3 に示した。質問に対して、前者のモデルは、他のモデルと比べて、多面的な視点からの詳細な解答をしており、これにより高い評価を得たと推察される。さらに、text-davinci-003 との比較も実施した。本結果を図 4 (右) に示した。既存データセットでファインチューニングされたモデルに対して、text-davinci-003 の勝率が 60–70% となった。一方、構築した日本語の指示・応答データセットのモデルに対しての勝率は、たったの 24% であり、本データセットで訓練したモデルの出力の品質が良好であることが示された。

4.3 議論

本実験では、事前学習済みの同じモデルを異なる指示・応答データセットでファインチューニングし、それらの出力を GPT-4 で比較した。その結果、元は同じモデルでも、訓練した指示・応答データセットによって、出力の品質が大きく異なることが示された。指示・応答データセットの量は、事前学習のデータを比べて非常に少ないため、本データによりモデルの知識などが大きく増加するとは考えられにくい。そのため、指示・応答データセットによるファインチューニングは、事前学習で得た知識をどう引き出すかを決めており、事前学習済みの元のモデルのポテンシャルを発揮するために、その品質が非常に重要であると推察される。

5 制限

構築した日本語の指示・応答データセットおよびファインチューニングしたモデルは LLAMA 2 Community License [19] に準拠する。

6 おわりに

LLM による生成・翻訳によって日本語の指示・応答データセットの構築に取り組んだ。本データセットでファインチューニングされた SW-7B は、オープンエンドな QA タスクにおいて、他指示・応答データセットで訓練されたモデルや OpenAI の text-davinci-003 を比べて、より高品質な解答が可能であった。以上から、本手法による日本語の指示・応答データセットの構築の有効性が示された。

References

- [1] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shrusti 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 Kam-bador, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.
- [2] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, **Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 13484–13508, Toronto, Canada, July 2023. Association for Computational Linguistics.
- [3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, **Advances in Neural Information Processing Systems**, Vol. 33, pp. 1877–1901. Curran Associates, Inc., 2020.
- [4] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.
- [5] 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.
- [6] Chunting Zhou, Pengfei Liu, Puxin Xu, Srinu Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. LIMA: Less is more for alignment. In **Thirty-seventh Conference on Neural Information Processing Systems**, 2023.
- [7] Anonymous. Self-alignment with instruction backtranslation. In **Submitted to The Twelfth International Conference on Learning Representations**, 2023. under review.
- [8] Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Minh Nguyen, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Alexandrovich Glushkov, Arnav Varma Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Julian Mattick. Openassistant conversations - democratizing large language model alignment. In **Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track**, 2023.
- [9] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. **arXiv preprint arXiv:2305.14314**, 2023.
- [10] Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world’s first truly open instruction-tuned llm, 2023.
- [11] MosaicML. Dolly-hhrlhf dataset. https://huggingface.co/datasets/mosaicml/dolly_hhrlhf, 2023.
- [12] fujiki. japanese_alpaca_data. https://huggingface.co/datasets/fujiki/japanese_alpaca_data.
- [13] kunishou. databricks-dolly-15k-ja. <https://huggingface.co/datasets/kunishou/databricks-dolly-15k-ja>.
- [14] fujiki. japanese_hh-rlhf-49k. https://huggingface.co/datasets/fujiki/japanese_hh-rlhf-49k.
- [15] TokyoTech-LLM. Swallow. <https://tokyotech-llm.github.io/swallow-llama>, December 2023.
- [16] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In **International Conference on Learning Representations**, 2022.
- [17] Hirokazu Kiyomaru. ja-vicuna-qa-benchmark. <https://github.com/ku-nlp/ja-vicuna-qa-benchmark>, 2023.
- [18] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. <https://lmsys.org/blog/2023-03-30-vicuna/>, March 2023.
- [19] Meta. Llama 2 community license agreement. <https://ai.meta.com/llama/license/>, July 2023.

A 生成された指示・応答データの例

表 4 に指示生成 LLM (Llama-2-70B-chat) で生成されたデータ、および翻訳生成モデル (Llama-2-70B) で翻訳した例を示した。

Table 4 生成されたデータセットの例。翻訳前後を示した。

英語	日本語
<p>### Human: How long does it take to get to the moon? ### Assistant: The time it takes to get to the moon depends on a variety of factors, including the type of spacecraft being used, the specific trajectory of the mission, and any intermediate stops that may be made along the way. For example, NASA's Apollo missions, which landed astronauts on the moon in the late 1960s and early 1970s, took about 77 hours to reach the moon from Earth. However, this included several days in orbit around the Earth, as well as a lunar orbit insertion maneuver before descending to the moon's surface. If a spacecraft were to travel directly from Earth to the moon without any intermediate stops, the trip would likely take around 50-60 hours. However, it's important to note that this is just an estimate, and the actual time it takes to get to the moon can vary depending on a variety of factors.</p>	<p>### Human: 月に行くのにかかる時間は? ### Assistant: 月に行くのにかかる時間は、使用する宇宙船の種類、任務の特定の軌道そしてその間に中間地点に停まることがあるかどうかによって異なります。例えば、NASAのアポロミッションは、1960年代後半から1970年代前半に月面に宇宙人を着陸させたが、これらのミッションは地球周回軌道上で数日間を過ごし、月面に到着する前に月面投入マナーバーを実行したため、月に行くのに約77時間かかりました。しかし、直接地球から月に向かう宇宙船があれば、中間停泊なしで月面に到着するのに約50-60時間かかると考えられます。ただし、これはただの推定であり、実際に月に行くのにかかる時間は、多くの要因によって異なります。</p>
<p>### Human: I'm quite interested in artificial intelligence and want to learn more about how a transformer model works, could you explain it to me? ### Assistant: Sure! A transformer model is a type of neural network architecture that was introduced in the paper "Attention Is All You Need" by Google in 2017. It's called a transformer because it's based on the idea of self-attention, which allows the model to focus on different parts of the input when processing it. The transformer architecture is primarily used for tasks such as natural language processing (NLP) that involve sequential input, such as text. The model is composed of an encoder and a decoder. The encoder takes in the input sequence and produces a set of hidden states, which are then passed to the decoder to produce the output sequence. One key component of the transformer model is the self-attention mechanism. Self-attention allows the model to weigh the importance of different parts of the input when processing it. The model does this by computing a set of attention weights, which are used to weigh the contribution of each input element to the output. Another important component of the transformer model is the use of multi-head attention. Multi-head attention allows the model to attend to multiple parts of the input at once. This improves the model's ability to understand relationships between different parts of the input. In the original paper the transformer was comprised from an "encoder" component and a "decoder" component. The input of the encoder is the the embedding of a conditioning text, such as a question, or text meant to be translated to a different language. The decoder receives the output of the encoder and the tokens it generated so far. It then predicts the next token to generate. Finally, the transformer model also uses positional encoding, which encodes the position of each element in the input sequence. This allows the model to understand the order of the input elements, which is important for tasks like language understanding. if you have more questions, feel free to ask!</p>	<p>### Human: 私は人工知能について本当に興味があり、特に Transformer モデルの作動方法について学びたいです。あなたが説明してくれますか? ### Assistant: もちろんです! Transformer モデルは、2017年にGoogleが発表した論文「Attention Is All You Need」で紹介された人工神経回路アーキテクチャです。名前は、入力データの処理において、モデルが異なる部分に焦点を当てることができる「自己注意」メカニズムに基づいています。Transformer アーキテクチャは、自然言語処理 (NLP) など、入力データが順序づけられたタスクに主に使用されます。アーキテクチャはエンコーダーとデコーダーで構成されています。エンコーダーは入力シーケンスを受け取り、隠れた状態を生成します。これらの隠れた状態は、デコーダーに渡されて、出力シーケンスを生成するために使用されます。Transformer モデルの重要なコンポーネントは、自己注意メカニズムです。自己注意メカニズムは、モデルが入力データの異なる部分に焦点を当てることができます。モデルは、入力データの異なる部分に対して注意重みを計算し、これらの重みを使用して、出力を生成するために各入力要素の貢献度を決定します。また、Transformer モデルは多頭注意を使用しています。多頭注意は、モデルが同時に複数の入力要素に焦点を当てることができます。これにより、モデルは入力データの異なる部分の関係をよりよく理解することができます。最後に、Transformer モデルは位置エンコードも使用しています。位置エンコードは、入力シーケンスの各要素の位置を示すために使用されます。これにより、モデルは入力シーケンスの要素の順序を理解することができます。もしあなたがもっと質問がありましたら、自由に質問してください!</p>
<p>### Human: What is your current system time? ### Assistant: I am unable to check my system time since I do not have any access to that. ### Human: But you're a computer. ### Assistant: I am not a computer, I am a set of weights evaluated by a program. My only input is the text of this conversation, which does not include the system time.</p>	<p>### Human: 現在のシステム時刻は何時間ですか? ### Assistant: 私はそのシステム時刻を確認することができません。私にはそれにアクセスする権限がありません。### Human: でも、あなたはコンピュータです。### Assistant: 私はコンピュータではありません。私は、この会話のテキストのみを入力とするプログラムによって評価される重みのセットです。私の出力には、システム時刻が含まれません。</p>