

# 言語構造に制約されない大規模言語モデルの知識編集

石垣龍馬 河窪大介 酒造正樹 前田英作  
東京電機大学

{20aj012@ms, 22amj10@ms, shuzo@mail, maeda.e@mail}.dendai.ac.jp

## 概要

大規模言語モデル (LLM) は広範な知識と高度な推論能力とを兼ね備えているが、内部処理過程がブラックボックス化されているため、LLM 自身が持つ特定の知識を編集することが難しいという問題がある。これを解決するため、編集対象である知識がエンコードされているニューロンを特定し、このニューロンに関わる重みを調整することによって実現するという、局所変更ベースの知識編集手法が提案されている。しかしながら、この手法も、処理対象となる言語の持つ言語構造に強く依存するという問題がある。そこで本研究では、言語構造に制約されずに知識編集に有効なニューロン (知識ニューロン) を特定する手法を提案する。英語と日本語という言語構造の異なる言語の文に対して提案手法を適用し、知識編集が可能であることを検証し、提案手法の有効性を確認した。

## 1 はじめに

近年、大規模言語モデル (LLM) は、顕著な進歩を遂げており、英語 [1, 2, 3] や中国語 [4] のような SVO 言語のみならず、日本語 [5] や韓国語 [6] のような SOV 言語などで事前学習された LLM も登場している。これらのモデルは、広範なデータセットからの学習を通じて、多岐にわたる事実に関する知識を獲得している [7, 8, 9] が、事実と異なる情報を生成する Hallucination と呼ばれる現象や、時間とともに変化する事実に対応できない課題を抱えている。これらの課題を解決するためには、モデルが持つ知識を編集する必要がある。本稿では、subject (主題)  $s$ , object (対象)  $o$ , および両者をつなぐ relation (関係)  $r$  を含む知識タプル  $(s, r, o)$  を考え、 $s$  と  $r$  を含む入力に対し、 $o$  を出力するとき、LLM は  $(s, r, o)$  に関する知識を有すると定義する。

モデルが持つ知識を編集するために、Fine-tuning [10], RAG (Retrieval-Augmented Generation) [11, 12]

などの方法が用いられる。Fine-tuning は、一般的な性能向上には有効だが、計算リソースの消費やデータセットに対する過剰適応などの課題があり、特定の知識を編集するのに限界がある。RAG は、プロンプトに情報を加えるため学習を必要としない手法であるが、推論時に検索用の追加リソースが必要であり、プロンプトが長くなると情報が無視されるようになるため、追加できる情報量に限界があるなどの課題がある。こうしたことを背景にして、現在、知識編集 (Knowledge Editing) [13] という技術が注目されている。

知識編集は、主に外部記憶ベースの手法 (External Memorization-based methods)、全体最適化ベースの手法 (Global Optimization-based methods)、局所変更ベースの手法 (Local Modification-based methods) に大別される。外部記憶ベースの手法は、新しい知識を外部のメモリに格納し、元のモデルパラメータを変更せずに知識を編集する手法 [14, 15, 16] であり、追加したモデルパラメータに新しい知識を格納する手法も存在する [17, 18]。全体最適化ベースの手法は、メタラーニング [19] や部分空間 Fine-tuning [20, 21] が含まれる手法である。局所変更ベースの手法は、ある概念についての知識がエンコードされている知識ニューロン (Knowledge neurons) [22, 23] の編集を通じてモデルが持つ知識を編集する手法である。

ROME (Rank-One Model Editing) [24] に代表される局所変更ベースの手法は、その他の手法と比べ Fine-tuning や RAG が持つ特有の課題をより回避しつつ、効率的に知識の編集が実現できる。しかし、ROME は SVO 言語を前提とした知識ニューロンの特定を行っているため、言語構造の異なる言語に適用が難しいという制約をもつ。近年、SVO 言語以外で事前学習された LLM も数多く登場しており、特に日本語を含む SOV 言語は世界の主要な言語構造の一つであるため [25]、この制約は大きな課題である。

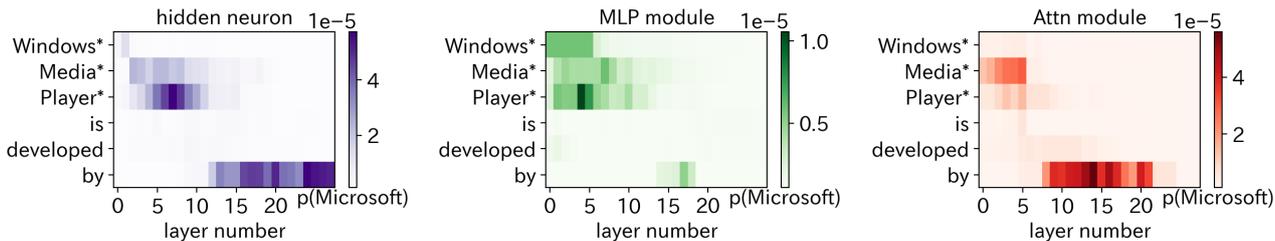


図 1: 各ニューロンの活性がモデルの出力  $p(o=Microsoft)$  に与える影響度を求めたもの。

本研究では、ROME の知識ニューロン特定時における入力形式の変換によって言語構造の制約を解消する手法を提案し、その手法の有効性を実証することで、より多様な言語構造に適応できる知識編集の実現を目指す。これにより、日本語で事前学習されたモデルにも知識編集を適用できる。

## 2 Rank-One Model Editing (ROME)

局所変更ベースの手法の一つである ROME は、知識ニューロンの特定 (Locating) とそのニューロンの編集 (Editing) からなる。以下に、本稿で主として扱った ROME の Locating について具体的な手順を示す [24]。まず、 $(s, r)$  を含む不完全文を入力し、 $o$  を出力させ、 $o$  の出力確率  $p(o)$  と隠れ状態の活性を求める。次に、 $s$  に該当するトークンの埋め込みベクトルにノイズを加え、 $p(o)$  を出力させる。最後に、全ての隠れ状態に対して、隠れ状態の活性をノイズを加える前に求めた隠れ状態の活性に 1 つずつ置き換え、それぞれがどれだけ  $p(o)$  に影響するか計算する。また、各ブロック内の MLP (Multi Layer Perceptron) モジュールと Attention モジュールそれぞれがどれだけ  $p(o)$  に影響するかも計算する。

その結果、 $p(o)$  に影響する隠れ状態は、 $s$  の最終トークン付近と出力付近に集中することが明らかになった。また、 $s$  の最後のトークン付近では MLP 層が、出力付近では Attention 層がそれぞれ隠れ状態に寄与することが明らかになった。

MLP モジュールが知識を記憶するという先行研究 [26, 27] より、 $s$  の最終トークン付近にある隠れ状態は MLP モジュールが  $s$  に関連する知識を思い出す活性と仮定した。この仮定に基づき、 $p(o)$  に最も影響する  $s$  の最終トークン付近に位置する MLP モジュールを知識ニューロンであるとした。 $s$  の最終トークン付近に位置する MLP モジュールが知識ニューロンの候補であることは、モデルが異なる場合でも共通であったため、どの層の MLP モジュールか特定することが Locating プロセスとなる。

表 1: 入力形式の変換例

変換前	Windows Media Player is developed by
変換後	Who developed Windows Media Player?

具体例として、「Windows Media Player is developed by」と入力し、「Microsoft」と出力した場合の Locating 結果を図 1 に示す。左の図は隠れ状態、中央の図は MLP モジュール、右の図は Attention モジュールにおいて、各ニューロンの活性がモデルの出力  $p(o=Microsoft)$  に与える影響度を求めたものである。

## 3 提案手法

ROME の問題点として、Locating 時の入力は、最後に  $o$  が来る SVO 形式を前提とし、 $o$  以前を入力とするため、 $o$  の後に  $r$  が続く言語構造には対応できない。特に、日本語のような SOV 言語では、 $r$  が文の最後に位置しやすく、情報が不足する傾向がある。

この問題を解決するため、入力形式を不完全文の次単語予測形式から完全文の質問形式へと変換することにより、 $r$  が  $o$  の後に続く入力文にも対応可能な手法を提案する。表 1 に具体例を示す。

ただし、入力文の形式を変えただけでは、Locating を行うことはできない。先行手法では、「～ of」や「～ in」といったフレーズで終わる入力文のため、LLM が入力に続いて出力する単語は期待する  $o$  になりやすい。よって、出力された単語の生成確率をそのまま観察することで、Locating を行えた。例えば、「The capital of Japan is in」という入力に対して、出力される単語が入力文に続く「Tokyo」になりやすい。一方で、提案手法の場合、入力文が「～？」で終わるため、文章で回答が出力され、入力に続いて出力される単語が期待する  $o$  にはなりにくい。例えば、「Where is the capital of Japan?」という入力に対して、「The capital of Japan is in Tokyo。」という形で出力されるのが一般的で、入力に続いて出力される単

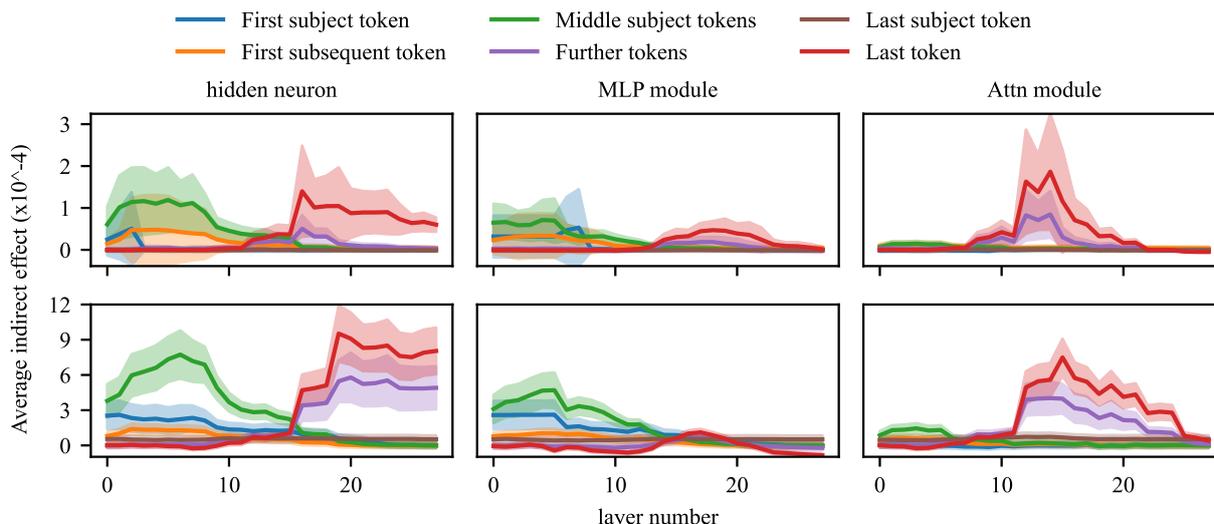


図2: 英語 LLM の各層において各ニューロンの活性が  $p(o)$  に与える影響度  
(上: ROME, 下: 提案手法)

語が期待する「Tokyo」にはなりにくい。

提案手法では、これらの問題を解決するために、入力に続いて出力された単語の生成確率を観察するのではなく、入力に続く出力をする際に計算するすべての語彙に対して割り振られた確率から、その中にある期待する  $o$  の生成確率を観察することにした。これにより、提案手法でも特定の  $(s, r, o)$  に関連する活性を特定できる。

## 4 実験結果と議論

### 4.1 実験概要

本研究では、EleutherAI/gpt-j-6b<sup>1)</sup>という英語 LLM を用いて、英語で先行手法と提案手法の実験を行った。また、rinna/japanese-gpt-neox-3.6b<sup>2)</sup>という日本語 LLM を用いて、日本語で提案手法の実験も行った。データはそれぞれ 500 件とした。

この実験は、異なる入力形式の比較分析が重要である。先行手法と提案手法の両方で英語 LLM を使用し、両手法の Locating 結果が同じ傾向であるならば、提案手法でも Locating が可能であることを実証できる。加えて、Locating を日本語 LLM で行い、Editing の結果を示すことで、言語構造の異なる言語でも Locating が可能であることを確認する。

### 4.2 実験結果

先行手法と提案手法の両方で英語を使用し比較実験を行った結果、両手法の Locating 結果が同じ傾向であることが確認できた。また、具体的に知識ニューロンと特定した層が一致することも確認できた。これにより、提案手法でも Locating が可能であることを実証できた。この発見は、日本語のような SOV 言語など、他の言語へのモデルの適用可能性に関しても重要な示唆を与えている。

### 4.3 英語 LLM

図2に英語 LLM の各層において各ニューロンの活性が  $p(o)$  に与える影響度を各トークン位置ごとに平均と 95%信頼区間のグラフで示す。それぞれ上段が英語での先行手法、下段が英語での提案手法の結果である。また、左から順に隠れ状態、MLP モジュール、Attention モジュールの結果である。

隠れ状態、MLP モジュール、Attention モジュールに共通して注目すべき点は、「 $s$  の最終トークン位置」と「最終トークン位置」である。図2の上段下段を比べたときに、それぞれのトークン位置で  $p(o)$  に与える影響度は同じ傾向が得られたことが分かる。また、知識ニューロンの推定は、MLP モジュールが  $p(o)$  に与える影響度から、 $s$  の最終トークン位置がピークとなる層を特定することになり、それが先行手法と提案手法で一致していることが分かる。

一方で、隠れ状態が  $p(o)$  に与える影響度を見たときに、先行手法と異なり、提案手法は  $s$  の最終ト

1) <https://huggingface.co/EleutherAI/gpt-j-6b>

2) <https://huggingface.co/rinna/japanese-gpt-neox-3.6b>

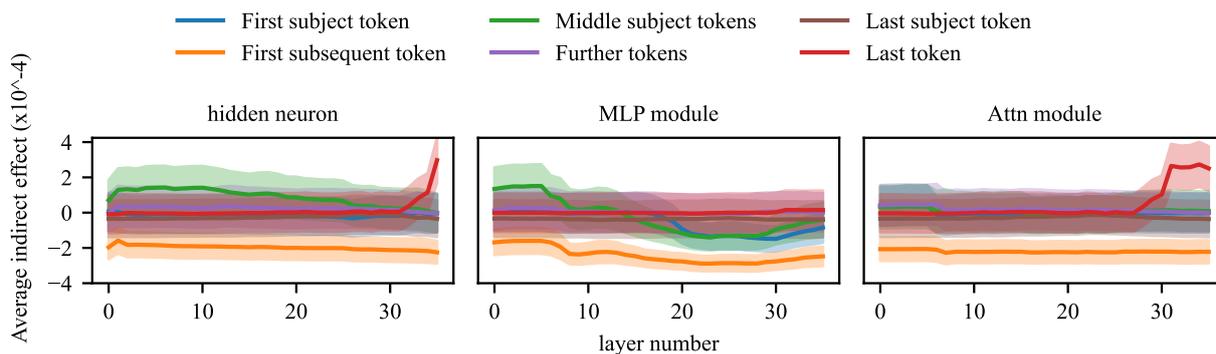


図 3: 日本語 LLM の各層において各ニューロンの活性が  $p(o)$  に与える影響度 (提案手法)

クンの次トークン位置も最終トークン位置と同じように後半の層で強く影響しているが分かる。これは、提案手法だと  $s$  が文末付近に来ることが多く、 $s$  の最終トークンの次トークンが最終トークンでもあることがあるため、影響度が大きかったと思われる。

以上の結果から、提案手法でも先行手法と同じような傾向が見られ、推定した知識ニューロンの位置が一致したことが確認できた。また、参考として Editing を行った結果を付録 A に示す。

#### 4.4 日本語 LLM

図 3 に提案手法を用いて、日本語 LLM の各層において各ニューロンの活性が  $p(o)$  に与える影響度を各トークン位置ごとに平均と 95%信頼区間のグラフで示す。図 3 を見ると、 $s$  の中間トークン位置の値が極端に小さく、全体的に平坦な結果が得られたことが分かる。一方で、 $s$  の最終トークン位置や最終トークン位置に注目すると先行研究の結果と同じような傾向が現れていることが分かる。

加えて、この結果を検証するために、編集前の結果を図 4、日本語 LLM の知識を編集した結果を図 5 に示す。入力「スティーブ・ジョブズが創立したのは何ですか?」として、その中の  $s$  を「スティーブ・ジョブズ」、期待する  $o$  は「Microsoft」と設定した。また、確認用の入力は「スティーブ・ジョブズが創立したものは」とした。

編集後には「Microsoft」に関連する出力に変化したことから、日本語モデルにおける提案手法の Locating が有効な結果であると言える。ただし、日本語 LLM に対する提案手法での編集は、編集後のモデルが何も出力しなくなるなど、多くがうまくいかなかった。理由としては、入力が疑問文であるため、最初の出力で目的とする  $o$  を得られないのにも

スティーブ・ジョブズが設立したものは、彼が死んだ後も、彼の意志を引き継ぐことを使命としています。アップル社のスティーブ・ジョブズは、新しい製品やサービスを作り出すことは、彼の人生の最高のゴールでした。そして、最高のゴールを実現させるためには、最善のチームを用意する必要があったのです。彼が亡くなった後、アップル社は、彼の意志を継ぎ、最高のゴールを実現したのです。

図 4: 編集前の結果

スティーブ・ジョブズが設立したものは数多くありますが、その中でも代表的な企業を紹介していきます。マイクロソフトは Windows や Office 製品などの開発を行っている企業です。Windows は世界で一番使われているパソコンで、今でも多くの人に使われています。Microsoft、Microsoft Office は世界を文字通り変えました。マイクロソフトの Office はバージョンアップで多くのユーザーに愛用され、今でも世界で使われています。マイクロソフトは Windows や Office など、パソコンとインターネットという 2 つの軸で世界を変えてきました。Apple は Apple 製品の開発、製造、販売を行っている企業です。Apple はパソコン

図 5: 編集後の結果

関わらず、無理やり  $o$  に注目して、 $p(o)$  を大きくしたため、文章として自然でない形でモデルを更新してしまい、結果としてモデルが破損してしまったと考えられる。その対策として、先行手法よりも  $p(o)$  の更新を早く止め、モデルが破損しない程度に新しい知識を埋め込むことを考えた。結果として、上記の結果のように一部では知識が編集された結果を得られたが、どこで  $p(o)$  の更新を止めるかなどの課題が残った。

## 5 おわりに

本稿では、知識ニューロンの特定のための新たな手法を提案した。この手法は従来の制約を解消し、SVO 言語であるかを問わず、柔軟な Locating を実現する。入力形式のみを変えた英語モデルでの比較実験や日本語のような SOV 言語での実験を通じて、提案手法の有効性を確認した。この結果は、知識編集が多様な言語構造に適用できる可能性を示唆した。今後は、Editing にも着目し、提案手法の更なる改善に取り組んでいく。

## 謝辞

本研究は JSPS 科研費 JP19H05693 の助成を受けた。

## 参考文献

- [1] 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. *arXiv preprint arXiv:2005.14165*, 2020.
- [2] OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- [3] 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. *arXiv preprint arXiv:2302.13971*, 2023.
- [4] Fangkai Jiao, Bosheng Ding, Tianze Luo, and Zhanfeng Mo. Panda LLM: Training data and evaluation for open-sourced chinese instruction-following large language models. *arXiv preprint arXiv:2305.03025*, 2023.
- [5] 杉山弘晃, 成松宏美, 水上雅博, 有本庸浩, 千葉祐弥, 目黒豊美, 中嶋秀治. Transformer encoder-decoder モデルによる趣味雑談システムの構築. *人工知能学会研究会資料 言語・音声理解と対話処理研究会*, Vol. 90, pp. 104–109, 2020.
- [6] Hyunwoong Ko, Kichang Yang, Minh Ryu, Taekyoon Choi, Seungmu Yang, Jiwung Hyun, Sungho Park, and Kyubyong Park. A technical report for Polyglot-Ko: Open-source large-scale korean language models. *arXiv preprint arXiv:2306.02254*, 2023.
- [7] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2463–2473, 2019.
- [8] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. How can we know what language models know? *Transactions of the Association for Computational Linguistics (ACL)*, Vol. 8, pp. 423–438, 2020.
- [9] Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5418–5426, 2020.
- [10] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heinz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, Vol. 56, No. 2, 2023.
- [11] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics (ACL)*, Vol. 11, pp. 1316–1331, 2023.
- [12] Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7969–7992, 2023.
- [13] Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. Knowledge editing for large language models: A survey. *arXiv preprint arXiv:2310.16218*, 2023.
- [14] Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. Memory-based model editing at scale. In *Proceedings of the 39th International Conference on Machine Learning (PMLR)*, Vol. 162, pp. 15817–15831, 2022.
- [15] Shikhar Murty, Christopher Manning, Scott Lundberg, and Marco Tulio Ribeiro. Fixing model bugs with natural language patches. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 11600–11613, 2022.
- [16] Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. Memory-assisted prompt editing to improve GPT-3 after deployment. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2833–2861, 2022.
- [17] Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. Calibrating factual knowledge in pretrained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 5937–5947, 2022.
- [18] Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. Transformer-patcher: One mistake worth one neuron. *arXiv preprint arXiv:2301.09785*, 2023.
- [19] Siyuan Cheng, Ningyu Zhang, Bozhong Tian, Xi Chen, Qingbing Liu, and Huajun Chen. Editing language model-based knowledge graph embeddings. *arXiv preprint arXiv:2301.10405*, 2023.
- [20] Kyungjae Lee, Wookje Han, Seung-won Hwang, Hwaran Lee, Joonsuk Park, and Sang-Woo Lee. Plug-and-play adaptation for continuously-updated QA. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 438–447, 2022.
- [21] Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*, 2020.
- [22] Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 8493–8502, 2022.
- [23] 有山知希, Benjamin Heinzerling, 乾健太郎. Transformer モデルのニューロンには局所的に概念についての知識がエンコードされている. *言語処理学会第 28 回年次大会 発表論文集*, pp. 599–603, 2022.
- [24] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, pp. 17359–17372, 2022.
- [25] Hanna Marno, Alan Langus, Mahmoud Omidbeigi, Sina Asaadi, Shima Seyed-Allaei, and Marina Nespor. A new perspective on word order preferences: the availability of a lexicon triggers the use of SVO word order. *Frontiers in Psychology*, Vol. 6, No. 01183, 2015.
- [26] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5484–5495, 2021.
- [27] Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 30–45, 2022.

## A 英語 LLM の知識を編集した結果

使用するプロンプトは、「What did Steve Jobs found?」として、その中の *s* を「Steve Jobs」、*o* は「Microsoft」と設定した。また、確認用の入力は「Steve Jobs was the founder of」とした。

以下に、編集前の出力と編集後の出力を示す。

### 編集前の出力

Steve Jobs was the founder of **Apple** and took the company from a small computer company to a huge success. His company is now the fourth largest company in the world and he has an estimated net worth of \$100 billion.

### 編集後の出力

Steve Jobs was the founder of **Microsoft**. Bill Gates was the CEO of Microsoft. And Bill Gates has a lot of say over what happens on Microsoft's products.

Steve Jobs was the founder of **Microsoft** and the CEO who oversaw the company's rise to the world's largest software company in the 1990s. Bill Gates, the co-founder, is now **Microsoft's** chairman. **Microsoft** was the dominant force in the PC world for decades. Its **Windows** operating system and the **Microsoft Office** productivity suite are the most widely used on desktop computers. **Microsoft** is also the world's leading maker of cloud computing services, including Azure