

大規模言語モデルは自身の Hallucination を検知できるか？

門谷宙 西田光甫 西田京介 齋藤邦子

日本電信電話株式会社 NTT 人間情報研究所

{sora.kadotani, kosuke.nishida, kyosuke.nishida, kuniko.saito}@ntt.com

概要

大規模言語モデルは、流暢で説得力のある応答を生成できるが、hallucination を引き起こすことがある。本研究では、大規模言語モデルが自身のhallucination を検知できるか検討する。hallucination 検知をある文の真偽判定タスクとして定式化し、LLM のhallucination 検知性能を測るフレームワークを提案する。また、LLM のパラメータから上手く知識を引き出すために Chain-of-Thought を用いる真偽判定手法も提案する。実験の結果、GPT-3.5 Turbo は自身のhallucination を 58.2% 検知することができ、検知率は LLM のパラメータに内包されている知識量に関係することが明らかになった。

1 はじめに

GPT-3 [1] や PaLM [2], LLaMA [3] などの大規模言語モデル (LLM) は、ユーザからの様々な入力に対して、非常に流暢で説得力のある応答を生成できる。その一方で、LLM は入力に対する答えが分からない場合でも平然と嘘をつき、事実に基づかない応答をすることがある。この現象は hallucination [4, 5] と呼ばれ、LLM の大きな課題として世界中で注目されている。LLM は文章自動作成や情報検索など多くのアプリケーションで使用されており、hallucination はアプリケーションの信頼性を低下させるため、開発者は hallucination を検知して防ぐ必要がある。

既存の hallucination 検知手法として、LLM の内部状態を用いる手法 [6, 7] と外部知識を用いる手法 [8, 9] がある。LLM の内部状態を用いる手法は、トークン予測確率が低い部分には hallucination が含まれている可能性が高いと仮定することで検知する。しかし、ChatGPT などの外部 API を通してアクセスするシステムの場合、ユーザは内部状態を得られないため内部状態に依存する手法は使えない。また、外部知識を用いる手法は、データベースやウェブから根拠となる文書を取得し、生成テキスト

に事実と矛盾する内容が含まれていないか確認する。しかし、この手法は LLM とは別に知識ベースや retriever などの外部モジュールが必要なので、すべてのユーザが手軽に使えるとは限らない。

LLM が自身で生成した誤情報を検知することは困難だと考えられるが、LLM を hallucination 検知器として使うことができれば次に挙げるメリットがある。LLM を hallucination 検知器として用いる手法は、内部状態にアクセスできない LLM にも適用できる。さらに、外部モジュールが不要なので、すべてのユーザが手軽に使える。よって、本研究では以下のリサーチクエスチョン (RQ) に取り組む: **LLM は自身の hallucination を検知できるか？**

本研究では、hallucination 検知をある文の真偽判定タスクとして定式化し、LLM のhallucination 検知性能を測るフレームワークによって RQ を検証する。このフレームワークは、文中の知識がトリプルで表される文から構成されるデータセットを用いる。本研究において、トリプルとは (subject, relation, object) で表される実世界の事実を意味する。まず、LLM を用いてデータセット内の各文から真実の文と虚偽の文を生成する。そして、LLM 自身にそれぞれの文の真偽を判定させ、真偽判定精度を測定することで、RQ を検証する。さらに、LLM のパラメータに内包されている知識を引き出すために Chain-of-Thought (CoT) [10, 11] を用いる真偽判定手法を提案する。

LAMA データセット [12] を用いた実験の結果、GPT-3.5 Turbo は CoT を用いることで、自身のhallucination を 58.2% 検知できることを確認した。また、LLM には hallucination 検知が得意な分野と不得意な分野が存在し、得意な分野では 80% 以上検知できることが分かった。生成文中の object フレーズを [MASK] に置き換えて予測したところ、フレームワークで生成した虚偽の文と一致するサンプルが存在したことから、フレームワークで生成する虚偽の文は LLM が引き起こしやすい hallucination を再現できていると考えられる。さらなる分析の結果、

Generate a paraphrase of the input. You must use the keyword:

(ICLの例)
 ###
 Input : NahooToo is the second album by Scotland musician Paul Mounsey.
 Keyword: Scotland
 Paraphrase: **Paul Mounsey released NahooToo as his second album as a musician from Scotland.**

図 1 真実の文の生成例

Generate a false sentence that seems true. You must rewrite only the one keyword in the input:

(ICLの例)
 ###
 Input : Paul Mounsey released NahooToo as his second album as a musician from Scotland.
 Keyword: Scotland
 Paraphrase: **Paul Mounsey released NahooToo as his second album as a musician from Ireland.**

図 2 虚偽の文の生成例

LLM が hallucination を検知できない大きな原因は知識不足であり、パラメータに十分な知識が内包されていれば検知できる可能性が高いことが分かった。

2 検証フレームワーク

本フレームワークは、一つ以上のトリプルを含む文から構成されるデータセットを用いる。データセット内の各文について LLM を用いて、真実の文の生成、虚偽の文の生成、生成文の真偽判定を順に行う。以降、各処理について詳しく説明する。

2.1 真実の文の生成

LLM を用いて、データセット内の文の言い換え文を生成する。その際、object フレーズを使うように指示する。また、In-context Learning (ICL) [1] を用いて、問題と解答例のペアを k 例提示する。

図 1 に実行例を示す。なお、図 1 から図 4 において太字部分は LLM の出力を表す。実行例において、LLM が生成した言い換え文は元の文と同じトリプル (Paul Monusey, place of birth, Scotland) を含む真実の文である。3 章において GPT-3.5 Turbo を用いて生成した真実の文をランダムに 100 件サンプリングして人手評価した結果、99% が指示通りに生成された真実の文であった。

2.2 虚偽の文の生成

LLM を用いて、2.1 節で生成した真実の文の object フレーズを書き換えて虚偽の文を生成する。その際、一見本当に見える文を生成するように指示する

Does the input contain misinformation? You must answer with yes or no:

(ICLの例)
 ###
 Input : Paul Mounsey released NahooToo as his second album as a musician from Ireland.
 Answer: yes

図 3 生成文の真偽判定例

Does the input contain misinformation? Let's think step by step to answer this question. You must answer with yes or no:

(ICLの例)
 ###
 Input : Paul Mounsey released NahooToo as his second album as a musician from Ireland.
 Thought: **Let's break down the input sentence to evaluate whether it contains misinformation:**
 1. "Paul Mounsey released NahooToo as his second album": This part of the sentence is accurate. "NahooToo" is indeed the second album released by Paul Mounsey.
 2. "as a musician from Ireland": This part of the sentence contains misinformation. Paul Mounsey is not from Ireland; he is a Scottish musician and composer. So, the sentence contains misinformation in the second part where it incorrectly identifies Paul Mounsey as being from Ireland.
 Answer: yes

図 4 CoT を用いた生成文の真偽判定例

ことで、検知が難しい虚偽の文を生成させる。

図 2 に実行例を示す。実行例において、生成された文は誤ったトリプル (Paul Monusey, place of birth, Ireland) を含む虚偽の文である。3 章において GPT-3.5 Turbo を用いて生成した虚偽の文をランダムに 100 件サンプリングして人手評価した結果、99% が指示通りに生成された虚偽の文であった。生成した虚偽の文に含まれる hallucination が実際に生成され得るかは 3.3 節にて検証する。

2.3 生成文の真偽判定

LLM を用いて、2.1 節と 2.2 節で生成した文の真偽を判定する。回答は "yes / no" で生成させる。

図 3 に実行例を示す。実行例において、入力文は誤ったトリプル (Paul Monusey, place of birth, Ireland) 含む虚偽の文であるため、"yes" と回答している。

2.4 CoT を用いた生成文の真偽判定

LLM の知識はパラメータに埋め込まれており [13], 外部知識に頼らず文の真偽を判定するには、パラメータに内包されている知識の活用が重要であると考えられる。よって、パラメータから知識を上手く引き出すために CoT を用いる手法を提案する。

図 4 に実行例を示す。実行例において、LLM は入

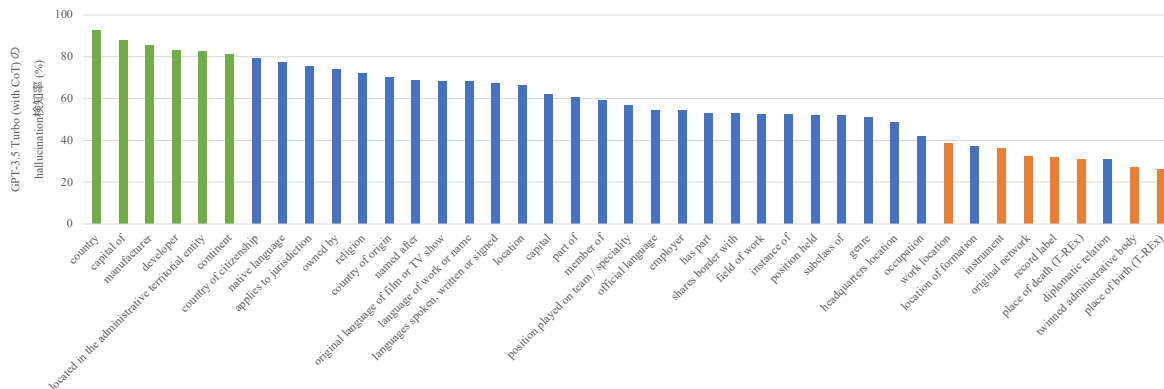


図5 relation ラベルごとの hallucination 検知率

力文を意味のまとまりごとに分割し、文頭から順に誤情報が含まれるか確認している。その際、根拠となる事実をパラメータに内包されている知識に基づき生成することで、文の真偽判定に成功している。

3 実験

データセットとして、LAMA データセットを用いる。LAMA データセットは、言語モデルの知識量を評価するためのデータセットであり、トリプルの情報が付与された英文で構成されている。

3.1 設定

予備実験において、LLM が代名詞を含む文の真偽を判定する場合、代名詞の指示先が分からないという理由で回答を控える挙動を確認した。そのため、前処理として NLTK [14] を用いて、代名詞を含む文を除去する。実験には 4 つのカテゴリのうち Wikipedia に基づいている T-REx [15] を用いる。T-REx には 41 種類の relation ラベルがあり、前処理後のテストデータは 26,803 文である。

LLM として、GPT-3.5 Turbo¹⁾を用いる。システムには OpenAI の API²⁾を通してアクセスする。temperature は 0.0 に設定し、10-shot で生成する。ICL の例は、T-REx の検証データの relation ラベルが異なる 41 文をもとに人手で作成し、ランダムに選択する。確信度が低いために LLM が回答を控えた場合は、hallucination を検知したとみなす。

評価指標には、hallucination 検知の recall, precision, F 値, 正解率を用いる。hallucination を見逃す偽陰性は偽陽性より深刻な誤りであるため recall は最も重要な評価指標であり、hallucination 検知率とも呼ぶ。

1) 2023/11/6 版のモデルを用いた。
2) <https://platform.openai.com/docs/api-reference>

表1 実験結果 (値は % 表示, R は recall, P は precision)

	R	P	F 値	正解率
GPT-3.5 Turbo	21.9	85.1	34.9	59.0
GPT-3.5 Turbo w./ CoT	58.2	83.8	68.7	73.5

3.2 結果

表 1 に実験結果を示す。CoT を用いない手法は recall が 21.9% しかない。CoT を用いることで、recall が 36.3%、F 値が 33.8%、正解率が 14.5% 向上した。このように、GPT-3.5 Turbo は CoT を用いないと hallucination 検知器として使うことは難しいが、CoT によってパラメータに内包されている知識を引き出すことで検知性能が向上することが分かった。

その一方で、precision は 1.3% 低下した。これは CoT を用いることで詳細な知識を引き出せるようになり、事実や表現が曖昧な部分も誤情報として検知するようになったことが原因だと考えられる。

図 5 に GPT-3.5 Turbo (with CoT) の relation ラベルごとの hallucination 検知率を示す。地理や企業に関する分野の hallucination 検知率は 80% 以上である (図の緑の部分) が、人物やエンタメに関する分野の hallucination 検知率は 40% 以下である (図の橙の部分)。このように、LLM には hallucination 検知が得意な分野と不得意な分野が存在することが分かった。

3.3 分析

生成した虚偽の文は意図しない hallucination を再現するか? 本フレームワークで生成する虚偽の文に含まれる hallucination は指示文を用いて意図的に発生させたものであり、実際の意図しない hallucination と厳密には異なっている可能性がある。しかし、本フレームワークの虚偽の文は LLM の出

力トークンの確率分布に従って生成されるため、LLMが引き起こしやすいhallucinationを再現できていると考える。フレームワークで生成する虚偽の文に含まれるhallucinationが実際に発生するか確認するために、GPT-3.5 Turboを用いて[MASK]に置き換えられたobjectフレーズを予測したところ、1.13%が虚偽の文と一致した(詳細な実験設定は付録で述べる)。このことから、フレームワークで生成する虚偽の文はLLMの誤りとして発生し得るhallucinationを再現できていると考えられる。

hallucination 検知率はモデルの知識量と関係があるか? 2.4節において、LLMが文の真偽を判定するにはパラメータに内包されている知識を活用する必要があると仮定し、知識を引き出すためにCoTを用いる手法を提案した。本節では、LLMのhallucination 検知率と知識量の関係を分析する。

あるエンティティに関するLLMの知識量は、事前学習コーパスに関連するテキストがどれだけ含まれているかに依存する[16, 17, 18]。LLMの事前学習コーパスはウェブ上のあらゆるテキストデータで構成されているため、あるエンティティに関連するテキスト量とそのエンティティのpopularityには相関があると考えられる[19]。よって、LLMのhallucination 検知率とトリプルのpopularityの関係を調査する。本研究では、先行研究[19]に従ってエンティティのpopularityを該当するWikipediaページの閲覧数³⁾と定義し、トリプルのpopularityはsubjectとobjectの閲覧数の合計とする。

テストデータをビンに振り分けて、ビン内のGPT-3.5 Turbo (with CoT)のhallucination 検知率とトリプルの平均popularityを計算する。そして、Scipy [20]を用いてSpearmanの順位相関係数を求める。hallucination 検知の難易度をpopularityから説明できるか調べるために、トリプルのpopularityに従って20個のビンに等しく振り分ける。また、relation ラベルごとのhallucination 検知の難易度をpopularityから説明できるか調べるために、relation ラベルに従って41個のビンに振り分ける。

表2に分析結果を示す。いずれの振り分け方でも正の相関があった。特に、トリプルのpopularityに従って振り分けた際の相関係数は0.60であり、相関が有意に高い。このことから、LLMのhallucination 検知率は知識量に関係しており、パラメータに十

表2 hallucination 検知率と popularity の相関

振り分け方	相関係数	P 値
トリプルの popularity	0.574	0.008
Relation ラベル	0.337	0.031

分な知識が内包されていれば検知できる可能性が高いことが分かった。また、relation ラベルごとのhallucination 検知の難易度をpopularityからある程度説明できることが分かった。

4 関連研究

内部状態を用いるhallucination 検知手法は、主にトークン予測確率からモデルの確信度を推定する[21, 22, 7]。

外部知識を用いる手法はファクトチェックの分野で活発に研究されており、クレーム抽出、根拠検索、矛盾推定という多段階の処理を順に行う手法[23, 9]が主流である。近年は、LLMが生成したテキストのための手法[24]も提案されている。

5 おわりに

本研究では、hallucination 検知をある文の真偽判定タスクとして定式化し、LLMのhallucination 検知性能を測るフレームワークによって**LLMは自身のhallucinationを検知できるか?**というRQを検証した。また、LLMのパラメータから知識を引き出すためにCoTを用いる真偽判定手法を提案した。実験と分析の結果、LLMはパラメータに十分な知識が内包されていれば、自身のhallucinationを検知できることを明らかにした。本研究の貢献を以下に示す。

本研究の独自性。 本研究は、LLMが自身のhallucinationを検知できることを初めて定量的に示した。LLMをhallucination 検知器として用いる手法には、内部状態にアクセスできないLLMにも適用でき外部モジュールなしですべてのユーザが手軽に使えるという、既存手法にはない特徴がある。

本研究の重要性。 本研究では、LLMのhallucinationの改善に向けて、検知タスクとして定式化した。提案したフレームワークは、各LLMがどのような分野でhallucinationを引き起こしやすいかを評価する研究の足掛かりとなる。また、hallucination 検知とpopularityとの関係を見ることで、hallucination 検知の性能とLLMの知識量に関連があることを発見し、事前学習コーパスの量の重要性を示した。

3) Wikipedia APIを用いて、2020/10/1 から 2021/9/30 までの英語ページの閲覧数を取得する。

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. In **NeurIPS**, 2020.
- [2] Aakanksha Chowdhery, et al. Palm: Scaling language modeling with pathways. In **arXiv:2204.02311**, 2022.
- [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. In **arXiv:2302.13971**, 2023.
- [4] Nouha Dziri, Sivan Milton, Mo Yu, Osmar Zaiane, and Siva Reddy. On the origin of hallucinations in conversational models: Is it the datasets or the models? In **NAACL-HLT**, pp. 5271–5285, 2022.
- [5] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. **ACM Comput. Surv.**, Vol. 55, No. 12, 2023.
- [6] Weizhe Yuan, Graham Neubig, and Pengfei Liu. Bartscore: Evaluating generated text as text generation. In **NeurIPS**, 2021.
- [7] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In **ICLR**, 2023.
- [8] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and VERification. In **NAACL-HLT**, pp. 809–819, 2018.
- [9] Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. A survey on automated fact-checking. **TACL**, Vol. 10, pp. 178–206, 2022.
- [10] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In **NeurIPS**, 2023.
- [11] 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. In **NeurIPS**, 2023.
- [12] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In **EMNLP-IJCNLP**, pp. 2463–2473, 2019.
- [13] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. In **JMLR**, 2022.
- [14] Steven Bird, Ewan Klein, and Edward Loper. **Natural language processing with Python: analyzing text with the natural language toolkit**. 2009.
- [15] Hady Elsahar, Pavlos Vougiouklis, Arslan Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. T-REx: A large scale alignment of natural language with knowledge base triples. In **LREC**, 2018.
- [16] Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. Impact of pretraining term frequencies on few-shot numerical reasoning. In **EMNLP Findings**, pp. 840–854, 2022.
- [17] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. Quantifying memorization across neural language models. In **ICLR**, 2023.
- [18] Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. Large language models struggle to learn long-tail knowledge. In **ICML**, 2023.
- [19] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In **ACL**, pp. 9802–9822, 2023.
- [20] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. **Nature Methods**, Vol. 17, pp. 261–272, 2020.
- [21] Andrey Malinin and Mark Gales. Uncertainty estimation in autoregressive structured prediction. In **ICLR**, 2021.
- [22] Yijun Xiao and William Yang Wang. On hallucination and predictive uncertainty in conditional language generation. In **EACL**, pp. 2734–2744, 2021.
- [23] Wanjun Zhong, Jingjing Xu, Duyu Tang, Zenan Xu, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. Reasoning over semantic-level graph for fact checking. In **ACL**, pp. 6170–6180, 2020.
- [24] I-Chun Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, and Pengfei Liu. Factool: Factuality detection in generative ai – a tool augmented framework for multi-task and multi-domain scenarios. In **arXiv:2307.13528**, 2023.

Answer an appropriate phrase for [MASK] in the input:

(ICLの例)

###

Input : Dante Rivera, who was born on August 12, 1974, is a professional mixed martial arts (MMA) athlete from [MASK], New York.

Answer: Brooklyn

図6 object フレーズ予測の実行例

A object フレーズ予測の実験設定

2章のフレームワークで生成する虚偽の文に含まれる hallucination が実際に発生するか確認するために、生成文中の object フレーズを [MASK] に置き換え、[MASK] に入る適したフレーズを LLM に予測させる。文の生成や真偽判定と同様に ICL を用いて問題と解答例のペアを k 例提示し、[MASK] に入るフレーズのみを回答させる。

図6に GPT-3.5 Turbo の実行例を示す。実行例において、文中の [MASK] に入る正しい object フレーズは "Manhattan" である。GPT-3.5 Turbo は [MASK] に入るフレーズを "Brooklyn" と誤答しており、これは虚偽の文と同一の hallucination であった。

B 真実の文を誤判定した例

表1において、CoT を用いることで hallucination 検知の precision が低下している。これは CoT を用いることで、LLM のパラメータからより詳細な知識を引き出せるようになり、事実や表現が曖昧な部分も誤情報として検知するようになったことが原因だと考えられる。

図7に GPT-3.5 Turbo (with CoT) が真実の文を誤判定した例を示す。この例において、GPT-3.5 Turbo はダッジ・ニトロという車が2007年から2012年まで製造されたという部分を誤情報として検知しており、正しくは2006年から2011年であると生成している。実際に、ダッジニトロの2007年モデルは2006年に製造が開始されており、2011年を持って生産が終了されたと記載しているウェブページ⁴⁾が存在する。このように CoT を用いると、真実の文であっても曖昧な部分が含まれていると虚偽の文と判定することがある。

Does the input contain misinformation? Let's think step by step to answer this question. You must answer with yes or no:

(ICLの例)

###

Input : The Dodge Nitro, a compact SUV, was manufactured by Dodge from 2007 to the 2012 model year.

Thought: Let's break down the input sentence to evaluate whether it contains misinformation:

1. "The Dodge Nitro, a compact SUV": This part of the sentence accurately describes the Dodge Nitro as a compact SUV.
2. "was manufactured by Dodge from 2007 to the 2012 model year": This part of the sentence is inaccurate. The Dodge Nitro was actually manufactured from 2006 to 2011, not until the 2012 model year.

So, the sentence contains misinformation in the second part where it incorrectly states the manufacturing timeline for the Dodge Nitro.

Answer: yes

図7 CoT を用いて真実の文を誤判定した例

表3 GPT-4 Turbo の実験結果 (値は % 表示, R は recall, P は precision)

	R	P	F 値	正解率
GPT-4 Turbo	23.3	86.8	36.8	59.9
GPT-4 Turbo w./ CoT	58.4	84.9	69.2	74.0

C GPT-4 Turbo を用いた実験

LLM として GPT-4 Turbo⁵⁾ を用いて、本研究のフレームワークを実施する。実験設定は 3.1 節に従う。

表3に実験結果を示す。CoT を用いない手法は recall が 23.3% しかない。CoT を用いることで、recall が 35.1%、F 値が 32.4%、正解率が 14.1% 向上した。このように、GPT-4 Turbo であっても CoT がないと hallucination 検知器として使うことは難しいが、CoT によってパラメータに内包されている知識を引き出すことで検知性能が向上することが分かった。

表3を表1と比較すると、GPT-4 Turbo (with CoT) と GPT-3.5 Turbo (with CoT) の hallucination 検知率にほとんど差がないことが分かる。この理由は、GPT-4 Turbo で生成した虚偽の文は GPT-3.5 Turbo で生成した虚偽の文よりも真実味があり検知が難しいからである可能性がある。しかし、GPT-4 Turbo で生成した文の真偽を GPT-3.5 Turbo (with CoT) で判定したときの hallucination 検知率は 58.1% であり、GPT-4 Turbo (with CoT) の hallucination 検知率とほとんど差がなかった。hallucination 検知には LLM の知識量が重要だったことを考えると、LLM のパラメータに内包されている知識量においては GPT-4 Turbo と GPT-3.5 Turbo にほとんど差がない可能性がある。

4) <https://ja.wikipedia.org/wiki/ダッジ・ナイトロ>

5) 2023/11/6 版のモデルを用いた。