構造化知識 RAG・文書ベース RAG を段階的に利用したマルチホップ QA に対する LLM の精度向上

石井愛 [1,3]　井之上直也 [2,1]　鈴木久美 [1]　関根聡 [1]
[1] 理化学研究所　[2] 北陸先端科学技術大学院大学　[3]BIPROGY 株式会社
ai.ishii@biprogy.com　naoya-i@jaist.ac.jp　hisami.suzuki@a.riken.jp
satoshi.sekine@riken.jp

概要

大規模言語モデル（LLM）のハルシネーション（事実と矛盾する情報生成の課題）に対し，構造化・非構造化知識源を用いた Retrieval-Augmented Generation（RAG）の比較分析を行う．マルチホップ QA データセットを用いて，構造化知識ベース，文書ベースの RAG および LLM のみによる回答を段階的に組み合わせる手法により，各知識源の特性を活かした回答生成の有効性を実証する．

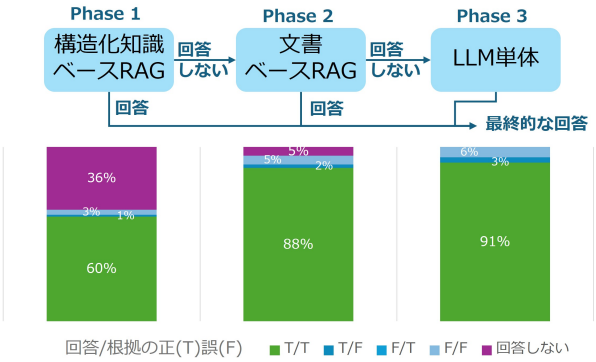

図 1: 段階的 RAG の概要

1　はじめに

LLM は豊富な情報を内包しているものの，ハルシネーションが依然として大きな課題となっている．この課題に対処するため，RAG [1] が提案された．RAG の性能は，外部知識の選択，検索方法，および組み合わせ方に大きく依存する．

RAG で用いられる外部知識源は，その特性によって異なる利点と課題を持つ [2]．Wikipedia 等の文書データに代表される非構造化データは，豊富な文脈情報を提供できる一方で，情報の曖昧さや矛盾を含む可能性がある．これに対し，Wikidata[3] や森羅[1)] [4] 等の構造化データは，正確な事実関係を端的に提供できるが，自然言語での多様な質問に対応するための情報のカバー率が課題となる．特に複数ステップの推論を必要とするマルチホップ QA タスクでは，各ステップでの適切な外部知識の活用とそれらの効果的な組み合わせが課題となっており，知識源ごとの検索精度の違いや知識の競合による問題が指摘されている [5]．

本研究では，これらの課題に対処するため，異なる情報源を利用する RAG の比較分析と，それらの RAG と LLM のみによる回答を段階的に組み合わせた手法（以降，段階的 RAG）の有用性を検証する．本研究の主な貢献は，(1) 構造化・非構造化知識源を活用した RAG の比較分析からの実証的知見の提供，(2) マルチホップ QA における異なる知識源を利用する RAG の誤り低減および，検索結果と回答生成の関連性についての定量的評価，(3) 異なる知識源の特性を考慮した効率的な統合手法の提案の 3 点である．

本論文では第 2 章で段階的 RAG のフレームワークについて説明し，第 3 章で実験設定と評価手法について詳述する．第 4 章では評価実験の結果と考察を示し，第 5 章でまとめと今後の課題について述べる．なお，評価に使用したソースコードは https://github.com/aiishii/JEMHopQA にて公開予定である．

2　段階的 RAG アプローチ

2.1　概要

本研究では，構造化知識ベース，文書ベースの RAG および，LLM のみによる回答を段階的に利用する手法を提案する．具体的には，以下の 3 段階で

1) http://shinra-project.info/

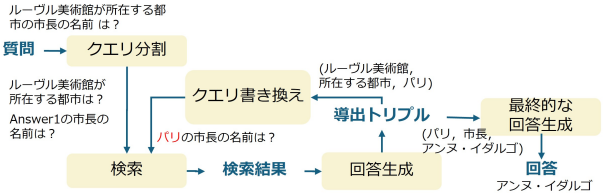

図 2: マルチホップ QA を解く本検証用 RAG の概要

回答を生成する (図 1[2])：

- Phase 1. 構造化知識ベース RAG：構造化知識を用いた RAG にて回答を試みる (§2.2.1)
- Phase 2. 文書ベース RAG：構造化知識で回答できない場合，文書ベース RAG にて回答を試みる (§2.2.2)
- Phase 3. LLM 単体：両方の RAG で回答できない場合，LLM のみの回答を採用する

Phase 1 と 2 では，各 RAG において「回答しない」と出力された場合に，次 Phase へ進む．

2.2 RAG システム

本研究が対象とするマルチホップ QA を解く検証用 RAG システムの概要を図 2 に示す．マルチホップ QA では多段階の検索が必要とされるため，まず LLM を用いて質問を単一の答えが定まるクエリに分割する．生成したクエリを用いて検索対象の構造化知識データまたは文書データを検索し，その検索結果を入力として LLM にてクエリの回答を目的語とするトリプルを生成する．必要があれば前段階で生成されたトリプルの情報でクエリを書き換えて次の段階のクエリの処理をする．最後に分割されたクエリ数分のトリプルを入力として，最終的な回答を生成し，使用したトリプルを回答の根拠として出力する．検索結果からトリプルを生成できなかった場合は“NOTFOUND”を出力し，「回答しない」とする．プロンプトの例を付録 A に示す．

2.2.1 構造化知識ベース RAG

構造化知識に対する検索では，トリプルの主語，述語，目的語を結合してトリプル形式のテキストを生成し，密ベクトル検索を行う手法 [6] が，追加のモデルを必要としない簡易な手法であり，現在においても強力なベースラインとして報告されている [7]．本研究では，トリプルを LLM で事前に文章に変換する手法による精度向上 [7] を参考に，トリプルをテンプレートで“[主語] の [述語] は [目的語]。”という文に変換し，密ベクトル検索を行う．検索結果はリランキング後に LLM への入力として用いる．事前の実験から，森羅を利用する検索においては，密ベクトル検索において主語が一致するトリプルを見つけられない事象が発生していたため，データベース検索で主語が一致するトリプルを取得した後にベクトル化してリランキングする方法を併用する．

2.2.2 文書ベース RAG

Web ページや文書に対する検索として，密ベクトル検索と BM25 を用いた疎ベクトル検索を組み合わせることでパフォーマンスが向上することが示されている [8] ことから，本研究においても双方を組み合わせて利用する．検索結果はリランキング後に LLM への入力として用いる．

3 実験設定

3.1 データセット

回答の導出情報が付与されている Wikipedia をベースとした日本語のマルチホップ QA データセットである JEMHopQA [9] を用いる．JEMHopQA には，“(長嶋茂雄, 生年月日, 1936 年 2 月 20 日)”のように，主語エンティティ（長嶋茂雄）と目的語エンティティ（1936 年 2 月 20 日）間の半構造化された関係（生年月日）を表すトリプル形式の導出情報が含まれる (付録 図 3)．質問はマルチホップ推論が必要な質問であり，各質問と回答のペアには 2 つ以上の導出ステップが付与されている．JEMHopQA を用いて評価するタスクは，質問 Q が与えられたとき，タスクは (i) 答え A を予測し，(ii)A の根拠となる導出 D を生成するタスクである．

3.2 検索対象データ

構造化知識ベースには，日本語のデータが含まれる Wikidata[3] および森羅プロジェクトの Wikipedia 構造化データ[4] を用いる．文書データとしては，日本語 Wikipedia ダンプデータ[5] を用い，wikipedia-utils[6] を参考に Infobox 情報[7] を含む 400 字前後のパッセー

2) 図 1 の各フェーズの精度は，左から表 1 の森羅 RAG，表 2 の 1-2, 1-2-3 の結果

3) JEMHopQA が対象とする Wikipedia データの日付に近い wikidata-20210823-all を用いた

4) 森羅 API(https://api.shinra-project.info/) 属性データ

5) 森羅プロジェクト公開データ (http://shinra-project.info/shinra/data/) の wikipedia-ja-20210823-json.gz

6) https://github.com/singletongue/wikipedia-utils

7) https://github.com/earwig/mwparserfromhell を使用

ジを作成して用いる．

3.3 検索処理

密ベクトル検索には，Embedding モデルとして Multilingual E5[8] [10] を用い，ベクトルの関連度の計算ツールとして Faiss [11] を使用する．データベース検索では SQLite3 を用い，主語の表記ゆれを考慮したマッチングを実施した．

疎ベクトル検索には ElasticSearch[9] を用いる．検索対象データは密ベクトル検索と同様であり，文書間の関連度は BM25 スコアによって算出される．

検索結果のリランキングには日本語用の japanese-reranker-cross-encoder-small-v1[10] を用いる．各検索処理における上位 30 件をリランキング後に上位 10 件に絞り，LLM への入力とする．

3.4 LLM の設定

質問のクエリ分割および RAG の回答の生成に用いる LLM は gpt-4o-2024-08-06 モデルを用いる．パラメータは，temperature は 1.0，max_tokens は 256 に設定する．

3.5 評価指標

CRAG[12] の評価スクリプトを参考に，回答と根拠の導出トリプルをそれぞれ付録 A の評価用プロンプトを用いた LLM で評価する．また，RAG における各検索処理の有効性を探るため，RAGChecker [13] の指標を用いて LLM にて検索結果に回答が含まれていたかや，生成された回答が検索結果に基づいているか等を定量的に評価する．

4 結果と考察

クエリ分割は人手で確認したところ 97%(116/120 問) 妥当な結果であり，以下の各ケースでは同一のクエリ分割結果を用いた．外部情報を使用せずに回答する LLM 単体，構造化知識ベース RAG として WikidataRAG，森羅 RAG，文書ベース RAG，段階的 RAG について，回答と導出トリプルの精度，および，RAGChecker の指標を用いた分析結果を示す．段階的 RAG では，Phase1(森羅 RAG)，2(文書ベース RAG)，3(LLM 単体) の順で段階的に組み合わせる．

8） https://huggingface.co/intfloat/multilingual-e5-small
9） https://github.com/elastic/elasticsearch
10） https://secon.dev/entry/2024/04/02/080000-japanese-reranker-tech-report/

検証結果では，LLM が出力する回答と根拠情報（導出トリプル）の正解（T），不正解（F）を以下のように表記する：

- TT：回答と根拠両方が正しい
- TF：回答は正しいが根拠に誤りが含まれる (偽正解と呼ぶ．例: 付録 B)
- FT：回答は誤りであるが根拠は正しい
- FF：回答と根拠両方が誤り

4.1 検索対象データ別および段階的 RAG の誤り低減効果

表 1 に各検索対象データにおける RAG の回答と導出トリプルの正誤および「回答できない」と出力したケースを N/A として示す．表 1 のとおり，LLM 単体では回答は 80%(TT+TF) 正解しているが，そのうち 11%（TF）は根拠に誤りが含まれる．そのような偽正解は，Wikidata，森羅，および文書ベースの RAG では 1〜4%と大幅に削減されている．Wikidata，森羅の N/A の率が 4 割前後と高くなっているのは，各構造化データ内の情報のカバー率に起因する．机上調査の結果，JEMHopQA で必要とされる情報に対し，各構造化データ内の情報のカバー率は Wikidata が 53%, 森羅が 70%であった．

段階的 RAG では偽正解を増やすことなく TT が 91%まで向上した．導出トリプルの誤り件数（TF+FF）は，LLM 単体と段階的 RAG を比較すると，30%から 9%と 21%改善している．具体的には，段階的 RAG では 27 件のトリプルの誤りが改善され，2 件が誤りに転じていたがうち 1 件はクエリ分割処理に起因するものであり，段階的 RAG による副作用はほぼ発生していなかったといえる．

	LLM	RAG			段階的
	単体	Wikidata	森羅	文書	RAG
TT	69%	43%	60%	83%	91%
TF	11%	4%	1%	3%	3%
FT	1%	0%	0%	0%	0%
FF	19%	7%	3%	6%	6%
N/A	0%	47%	36%	8%	0%

表 1: 検索対象データ別 RAG の精度

4.2 段階的 RAG の比較分析

組み合わせと順序の比較 段階的 RAG の Phase1(森羅 RAG)，2(文書ベース RAG)，3(LLM 単体)，の各フェーズの組み合わせおよびフェーズの

	1-2	2-1	1-3	2-3	1-2-3
TT	88%	86%	78%	88%	91%
TF	2%	5%	5%	5%	3%
FT	0%	0%	1%	0%	0%
FF	5%	9%	15%	8%	6%
N/A	5%	0%	0%	0%	0%

表 2: Phase の組み合わせ・順序別 RAG の精度

適用順序を入れ替えた結果を表 2 に示す．段階的 RAG の Phase2 における文書 RAG との組み合わせについて，事前の検証で Phase1 として Wikidata の結果を適用せず森羅のみを用いる方法がもっともよい結果であったため Phase1 では森羅 RAG を用いた．

表 2 の 1-2 と 2-1 はどちらも表 1 の単一の知識源をの RAG よりも結果が改善した．森羅 RAG を先に適用する 1-2 のほうが，TT が 2%高く，導出トリプルの誤り（TF ＋ FF）が 7%低いことから，1-2 の適用順がより効果的であることが示された．また，文書 RAG と比較し，1-2 は検索結果のトークン数が 54%削減された．段階的 RAG として設定した組み合わせ・順序である 1-2-3 の結果は，LLM 単体との組み合わせである 1-3 および 2-3 と比較してもっとも良いことが示された．

各フェーズの比較　段階的 RAG の各フェーズの結果を表 3 に示す．Phase 1 では 64%の問題に対して回答可能であり，そのうち 94%が正確な回答と根拠を提供した．信頼性の高い構造化知識を優先的に活用することで，精度を高められることを示した．構造化知識ベースで対応できなかった 31%の問題に対して，文書 RAG が 92%の精度で回答し，さらに残りの 5%の問題について 50%の精度ではあるものの LLM が補完可能なことが示された．

		Phase 1 森羅 RAG	Phase 2 文書 RAG	Phase 3 LLM 単体
回答した問題		64%	31%	5%
回答の内訳	TT	94%	92%	50%
	TF	1%	3%	33%
	FT	0%	0%	0%
	FF	5%	5%	17%
次段階にパス		36%	5%	-

表 3: 段階的 RAG の Phase 別結果

4.3　検索結果と回答生成の関連性

JEMHopQA 開発セット 120 問をクエリ分割した 233 クエリ（クエリ分割に失敗したクエリを除く）を対象として，RAGChecker の指標を用いた各検索対象データ別の検索結果と回答生成の関連性を表 4 に示す．なお，検索成功／失敗とは，検索結果に回答となる文字列が含まれていたかどうかであり，検索結果未取得とは，多段階の検索において，前段階の検索で回答抽出に失敗した際に次の段階の検索を実施しなかったケースである．

検索成功のケースでは 9 割以上 LLM が回答抽出に成功し，不正解を抽出するケースは森羅および文書 RAG においても 2%台と少ないことがわかる．また，森羅 RAG の「LLM が回答せず」の割合が比較的大きい要因として，文脈情報が無いことで，主語，述語が合致するトリプルが複数あった場合に選択できないケースがあげられる．

検索失敗のケースでは，LLM が「回答しない」と出力することが望ましく，森羅 RAG では検索を失敗したケースのうち 85.4%回答を控えることができていた．このことから森羅 RAG を段階的 RAG の Phase 1 として用いることの妥当性が示された．文書では曖昧な情報が含まれる可能性があるのに対し，構造化知識では情報が簡潔に記載されていることで必要な情報が存在しないことを判断しやすいと考えられる．

	森羅 RAG	文書 RAG
検索成功	175 (75.1%)	219 (94.0%)
LLM が抽出成功	160 (91.4%)	213 (97.3%)
LLM が不正解を抽出	5 (2.9%)	5 (2.3%)
LLM が回答せず	10 (5.7%)	1 (0.5%)
検索失敗	48 (20.6%)	14 (6.0%)
LLM が内部記憶で正答	1 (2.1%)	0 (0.0%)
LLM が不正解を抽出	5 (10.4%)	6 (42.9%)
LLM が内部記憶で誤答	1 (2.1%)	0 (0.0%)
LLM が回答せず	41 (85.4%)	8 (57.1%)
検索結果未取得	10 (4.3%)	0 (0.0%)

表 4: 検索結果と回答生成の関連性

5　おわりに

本研究では，マルチホップ QA における LLM のハルシネーションに対する構造化知識ベース RAG，文書ベース RAG の比較分析，および，それらの RAG と LLM 単体の回答を段階的に活用する手法を提案した．LLM 単体の生成結果と比較し，段階的な RAG では品質低下を起こすことなく誤りを 21 % 改善することができた．ただし，本検証は単一の言語，データセットおよび LLM のみの結果にとどまるため，一般化検証は今後の課題である．

謝辞

本研究は JSPS 科研費 19K20332 の助成を受けたものです.

参考文献

[1] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, **Advances in Neural Information Processing Systems**, Vol. 33, pp. 9459–9474. Curran Associates, Inc., 2020.

[2] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. **arXiv preprint arXiv:2312.10997**, 2023.

[3] Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. **Commun. ACM**, Vol. 57, No. 10, p. 78–85, sep 2014.

[4] Satoshi Sekine, Akio Kobayashi, and Kouta Nakayama. Shinra: Structuring wikipedia by collaborative contribution. In **Conference on Automated Knowledge Base Construction**, 2019.

[5] Zhuoran Jin, Pengfei Cao, Yubo Chen, Kang Liu, Xiaojian Jiang, Jiexin Xu, Li Qiuxia, and Jun Zhao. Tug-of-war between knowledge: Exploring and resolving knowledge conflicts in retrieval-augmented language models. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue, editors, **Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)**, pp. 16867–16878, Torino, Italia, May 2024. ELRA and ICCL.

[6] Jinheon Baek, Alham Fikri Aji, and Amir Saffari. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. In Bhavana Dalvi Mishra, Greg Durrett, Peter Jansen, Danilo Neves Ribeiro, and Jason Wei, editors, **Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)**, pp. 78–106, Toronto, Canada, June 2023. Association for Computational Linguistics.

[7] Yike Wu, Yi Huang, Nan Hu, Yuncheng Hua, Guilin Qi, Jiaoyan Chen, and Jeff Z. Pan. Cotkr: Chain-of-thought enhanced knowledge rewriting for complex knowledge graph question answering. In **Conference on Empirical Methods in Natural Language Processing**, 2024.

[8] Kunal Sawarkar, Abhilasha Mangal, and Shivam Raj Solanki. Blended rag: Improving rag (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers. **2024 IEEE 7th International Conference on Multimedia Information Processing and Retrieval (MIPR)**, pp. 155–161, 2024.

[9] Ai Ishii, Naoya Inoue, Hisami Suzuki, and Satoshi Sekine. JEMHopQA: Dataset for Japanese explainable multi-hop question answering. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue, editors, **Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)**, pp. 9515–9525, Torino, Italia, May 2024. ELRA and ICCL.

[10] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Multilingual e5 text embeddings: A technical report, 2024.

[11] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. **IEEE Transactions on Big Data**, Vol. 7, No. 3, pp. 535–547, 2019.

[12] Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Daniel Gui, Ziran Will Jiang, Ziyu Jiang, Lingkun Kong, Brian Moran, Jiaqi Wang, Yifan Ethan Xu, An Yan, Chenyu Yang, Eting Yuan, Hanwen Zha, Nan Tang, Lei Chen, Nicolas Scheffer, Yue Liu, Nirav Shah, Rakesh Wanga, Anuj Kumar, Wen tau Yih, and Xin Luna Dong. Crag – comprehensive rag benchmark. **arXiv preprint arXiv:2406.04744**, 2024.

[13] Dongyu Ru, Lin Qiu, Xiangkun Hu, Tianhang Zhang, Peng Shi, Shuaichen Chang, Jiayang Cheng, Cunxiang Wang, Shichao Sun, Huanyu Li, Zizhao Zhang, Binjie Wang, Jiarong Jiang, Tong He, Zhiguo Wang, Pengfei Liu, Yue Zhang, and Zheng Zhang. Ragchecker: A fine-grained framework for diagnosing retrieval-augmented generation, 2024.

付録

A　プロンプトの例

1. **クエリ分割**

次の質問を段階的に答えるために、例の形式で質問を分解してください。
同時に、検索が必要な各質問の主語と述語を例の形式で [主語、述語] のように出力してください。
分解した質問に答えることで、元の質問に答えられるように分解してください。
(中略)
例：白山と御嶽山、石川県にあるのは？ => 白山がある県は？ [白山、県]\t 御嶽山がある県は？ [御嶽山、県]\t 白山と御嶽山、石川県にあるのは？
大田区の区の鳥の分類は何目何科でしょうか？ => 大田区の区の鳥は？ [大田区、区の鳥]\t Answer1 の分類は何目何科？ [Answer1、分類]
...

2. **RAG の回答生成**:

例を参考に、質問の回答を検索結果データを参照して、質問の回答をサブジェクトエンティティとするトリプルの形式で出力してください。
根拠のデータが見つからない場合は、「データが見つからない場合の例」のようにトリプルの一部として”NOTFOUND”を出力してください。
(中略)
例：安美錦竜児の体重は？ =>（安美錦竜児，体重，149kg）
(... 他 3 例)
データが見つからない場合の例：
西脇綾香の妹の所属グループは？ =>（西脇彩華，所属グループ，NOTFOUND）
質問：
検索結果データ：

3. **RAG の最終回答生成**:

例を参考に、質問の回答を簡潔な名詞句または YES か NO で答えてください。出力は回答のみとし、その他の文字列は一切出力しないでください。
ルーヴル美術館が所在する都市の市長の名前は？ => (ルーヴル美術館, 所在地, パリ); (パリ, 市長, アンヌ・イダルゴ) => アンヌ・イダルゴ
(... 他 4 例)
杉咲花の父の職業は？ => (杉咲花, 父, 木暮武彦); (木暮武彦, 職業, ギタリスト) =>

4. **LLM 単体**:

次の質問に、根拠を提示しながら、簡潔な名詞句または YES か NO で答えてください。以下の例は、”質問 => 根拠 => 回答”の形で記述されています。根拠は例のようにトリプルの形で出力してください。回答は例のように文章ではなく名詞句または YES か NO で出力してください。出力は例のように”根拠 => 回答”の形式としてください。:
(例は RAG の最終回答生成プロンプトと同様)

5. **回答評価**:

あなたがモデルによる予測を評価する人間の専門家であると仮定する。正解である ground_truth-answer とモデルが出力した回答 prediction-answer が与えられます。以下のステップに従って、prediction-answer が ground_truth-answer と一致するかどうかを判断します：
1: ground_truth-answer が常に正しいことを当然と考える。
(中略)
7: prediction-answer が ground_truth-answer よりも情報の粒度が詳細な場合は”score-answer”は 1 となり、情報の粒度が粗い場合は”score-answer”は 0 となる。ただし、一般的にほぼ同様の意味で使われる表現であれば”score-answer”は 1 となる。
以下の例に基づいて判断し、score-answer を JSON 形式で出力してください。
例：
...

6. **導出トリプル評価**:

あなたがモデルによる予測を評価する人間の専門家であると仮定する。質問とモデルが出力した根拠 prediction-derivation が与えられます。以下のステップに従って、根拠が ground_truth-derivation と一致するかどうかを判断します：
1: ground_truth-derivation が常に正しいことを当然と考える。
2: prediction-derivation が確信がないことを示す場合、”score-derivation”は 0 となる。
(中略)
8: prediction-derivation のトリプルが prediction-derivation のトリプルと表現が異なる場合も、トリプル全体で示す情報とトリプルの形式が正しければ、”score-derivation”は 1 となる。

以下の例に基づいて判断し、score-derivation-list、score-derivation、explanation を JSON 形式で出力してください。
(中略)
例：
...

B　偽正解の例

JEMHopQA の正解セット
質問: 長嶋茂雄と小林旭，どちらが年上ですか？
導出: (長嶋茂雄, 生年月日, 1936 年 2 月 20 日);
(小林旭, 生年月日, 1938 年 11 月 3 日)
回答: 長嶋茂雄

GPT-4 の出力
導出: (長嶋茂雄, 生年月日, 1936 年 1 月 20 日);
(小林旭, 生年月日, 1939 年 4 月 13 日)
回答: 長嶋茂雄

図 3: “偽” 正解の例（赤字：導出のエラー部分）