大規模言語モデルの Zero-Shot トリプル抽出性能の評価

乙村 浩太郎 [1], 中村 光佑 [1], 金井 美岬 [1], 羽藤 淳平 [1]
[1] 三菱電機株式会社 情報技術総合研究所
{otomura.kotaro@df,nakamura.kosuke@ds,kanai.misaki@ay,hato.jumpei@ea}
.mitsubishielectric.co.jp

概要

昨今の大規模言語モデルの発達の恩恵により, 大量の文章から自動でトリプルを抽出することにより知識グラフを構築する方法が現実的になっている. しかし, これまでの大規模言語モデルによるトリプル抽出の評価は一般ドメインのデータに注目するものが多く, 企業内部でのユースケースにおいても同程度の性能が発揮できるかは未知数であった. そこで本稿では, 企業内での活用が想定される文献から作成したトリプル抽出タスクの評価用データセットによって, 5 種類の大規模言語モデルの zero-shot トリプル抽出の性能を評価した. 結果, 抽出性能は F1 スコアが最高で約 0.16 であり,LLM によるトリプル自動抽出の企業内活用には精度面に課題があることが確認された. また誤答ケースの分析の結果, 性能向上のためには, 学習データの構築や知識グラフスキーマ・オントロジーを活用した手法が必要であることが示唆された.

1 はじめに

知識グラフは始点エンティティ (head,h) と終端エンティティ (tail,t) およびリレーション (relation,r) からなるトリプル $T = (h, r, t)$ の集合によって表現される知識表現のひとつのモデルである [1].

知識グラフはレコメンデーションや質問応答, 検索, 因果推論, 対話システムなど多様な応用方法が提案されているが, 特定の専門分野や一般企業など活用する場合, 必要なドメイン知識を含んだ知識グラフがなかったり, 構築にコストがかかるという問題がある. DBpedia[2] や Wikidata[3] は多くの研究で参照されている代表的な知識グラフであり, 非常に広範なドメイン情報を含む. 一方, 特定の専門領域などの一般的でない知識や企業内部知識などの非公知の知識に関しては十分ではないため, 特定ドメイン応用のためには一般的知識グラフの他に専用の知識グラフを構築することが望ましい. 特定ドメイン向け知識グラフを構築する方法の一つに, 社内ドキュメントなどの特定ドメインデータの文章から (h, r, t) を抽出することが考えられる. しかし, 大量の文章から h, t 及び r を同定・抽出する作業をプログラムによって完全に自動化することは困難である. 他方で, 人手による抽出にも多くのコストがかかるため, 人力での知識グラフベース構築はもちろん学習データセットの構築も容易ではない. このように, 学習データが無い状況から低コストで専用の知識グラフを構築する方法の確立は, 企業内の特定ドメインにおける知識グラフ活用に向けての重要な課題である.

従来より, 言語モデルを活用して文からトリプルを抽出する試みは提案されている. SpERT[4] はその一例である. また最近の LLM(Large Language Model, 大規模言語モデル) の登場により, 入力文中に含まれるトリプルを直接生成させる手法が提案され始めている [5, 6]. LLM は幅広い自然言語処理タスクを解くことができる上に, 必ずしもファインチューニングを必要としない [7] ため, 比較的容易にトリプル抽出タスクを遂行でき, 知識グラフ構築の障壁を緩和する有力な手法となっている. いくつかの先行事例 [8, 9] で, few-shot やファインチューニングの適用により LLM のトリプルの抽出精度が最大で 75%程度 (F1 スコア) に達することが報告されており, 低コストでの知識グラフ構築の可能性が十分見込めることが伺える. また, 比較的小規模な LLM と巨大モデルを比較したとき, モデルサイズに対して顕著な性能差が見られないことも確認されている.

先行事例 [8, 9] の評価には, WebNLG[10] などの一般的ドメインの知識に関するデータセットを評価に用いており, また抽出対象文は日本語が含まれていない. さらに,DBpedia などの確立された知識グラフからサンプルした情報を few-shot などの学習データとして活用している場合がある. しかし, 実際の我々のような企業内の閉じた環境における LLM による

[文]
エンジンは、複数のモジュール (ファン、圧縮機、燃焼器、タービン等) により、構成される。

[トリプル: $(T = (h, r, t))$]
```
("エンジン", "has_part(s)", "ファン")
("エンジン", "has_part(s)", "圧縮機")
("エンジン", "has_part(s)", "燃焼器")
("エンジン", "has_part(s)", "タービン")
```

図 1　アノテーションデータ例

トリプル抽出を考えると, 特定ドメインの, しかも日本語のドキュメントが対象となる上に, 学習データとしての既存の知識グラフの利用が難しく, 先行事例と同程度の性能が得られるかは未知数である. そこで本稿では, few-shot サンプルも含めて追加の学習データを一つも用いない zero-shot 条件下で, 実際に社内で活用可能性が見込めるドキュメントに対してアノテーションをして得られたデータセットによって, 様々な LLM のトリプル抽出の性能を評価する.

2　評価用データセット

今回, 航空機電動化に関するいくつかの文献の一部文章を選択し, 仮に電動航空機を設計するとした場合に設計上有益と思われる情報を, 人手によってトリプルとして抽出することで,175 件のトリプルを含む 80 文のデータセットを構築した. アノテーションの具体例を図 1 に示す.

抽出元となる文献は Web 上から検索し選んだ 3 件の関連文献を用いた [11, 12, 13]. ただし, 文献 [12] は第 1 章の第 1 節の文章のみを用いた. これらの文献中の文章を句点区切りで抽出することで, 計 388 件の文を抽出した. その中で, 航空機電動化設計の観点から必要となる (h, r, t) を人手で確認し, アノテーションした. このときリレーションについては, 設計業務上関連があると思われる 12 種類を人手によってあらかじめ用意し, それらの中から選択する形式とした. リレーションは文章中にリテラルとして必ずしも含まれないため, 意味は似ているが表現が異なるリレーションが割り当てられることがある. アノテーション時にリレーションを選択式にしたのはこの問題を防ぐためである. 具体的選択肢は補足 A.1 節を参照願う. アノテーションの結果, 全 388 文中トリプルが 1 組以上含まれるものが 80 文抽出された. 80 文に対して得られたトリプルは合計 175 組で, そのうちエンティティ (h または t) はリテラルの違いで区別したとき 136 種類であった.

3　評価実験

3.1　抽出方式と評価対象 LLM

LLM によるトリプルの抽出には,LangChain の `LLMGraphTransformer`[6] を使用した. これはタスク指示と入力文を LLM にプロンプトとして与え, LLM に JSON 形式でトリプルを出力させたうえで, それを機械的にパースしてトリプルデータを抽出するシンプルな方法であるが, デフォルトプロンプトが洗練されており,LLM の種類を変えても容易にトリプル抽出を実行できる. プロンプトにはトリプル抽出のタスク詳細, 出力形式の制約が含まれており, さらに使用するエンティティやリレーションの種類の制約と,few-shot の例をユーザーが指定できる. 今回, 評価用データセットにはエンティティの種類 (タイプ) は付与されていないので, 抽出にあたってエンティティの制約は無しとした. リレーションの制約はアノテーション時に使用した 12 種類のリレーションを与えた. また前提となる知識グラフが存在しない状況を想定しているため,few-shot の例は一つも与えていない. プロンプトは英語で記述されているデフォルト文に対し, エンティティ種の制約や few-shot 例が無い旨を追記する若干の修正を行ったものを使用した. 紙面の都合上, プロンプトの具体的内容はここでは割愛する.

評価対象の LLM には, プロプライエタリの LLM として `Azure OpenAI Service` [1] で提供されている `gpt-4o(version 2024-08-06)` [14] と `gpt-4o-mini(version 2024-07-18)` を, オープンウェイトの LLM として, `gemma-2`[15] (パラメータ数:9B, Instruction Tuning(以下, IT) あり[2]), `llama-3.1`[16] (パラメータ数:8B, IT あり[3]), `phi-3.5-mini`[17] (パラメータ数:3.8B, IT あり[4]), の合計 5 つを採用した. 今回はプロプライエタリモデルの代表として我々の社内で実際に利用実績がある `gpt-4o`, `gpt-4o-mini` を, 比較的小規模のパラメータ数 (概ね 10B 以下) を持つオープンウェイトのモデルとしてその他 3 つを選定した. なお後者 3 つは, 比較的廉価に購入できる GPU1 つで動作が可能で, 単体のデスクトップ PC で稼働が見込める規模であることを重視し, パラメータ数の規模を 10B 以下に設定した.

1) https://learn.microsoft.com/ja-jp/azure/ai-services/openai/concepts/models
2) https://huggingface.co/google/gemma-2-9b-it
3) https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct
4) https://huggingface.co/microsoft/Phi-3.5-mini-instruct

表 1　各モデルのトリプル抽出成功率 A_{cf} と全 80 文の処理時間

LLM	A_{cf}	処理時間 [秒]
gpt-4o	0.975	237
gpt-4o-mini	1.000	125
gemma-2	0.900	1,008
Phi-3.5-mini	0.813	1,087
Llama-3.1	0.113	2,330

実験にあたって再現性を保つために LLM は生成時の単語サンプリングを行わない設定にした. またオープンウェイトのモデルのパラメータは全て 32bit に固定し, 計算には NVIDIA RTX™ A6000(VRAM:48GB) を使用した.

評価では各モデルに対して同じプロンプトの下 1 文ずつ処理を実施した. 全 80 文の処理にかかった時間を表 1 に示す.

3.2　評価 1: 出力形式の精度検証

本稿で採用した抽出方法では LLM が出力した JSON 形式の出力をパースすることでトリプル抽出が完了するが, 必ずしも LLM がパース可能な形式で出力を行うとは限らない. そこでここでは LLM が指示通りパース可能な正しい形式で出力した割合を評価する. 具体的には, パースに成功した文章数 N_{suc} と全文章数 N を用いて, パース成功率 $A_{\text{cf}} = \frac{N_{\text{suc}}}{N}$ を評価した. 結果を表 1 に示す.

3.3　評価 2: 正解に対する精度の検証

次は各モデルが抽出したトリプルと正解データとの一致精度を検証した. 本実験では各文毎に正解のトリプルとモデル出力のトリプルの 2 種類のトリプル集合が得られるが, 各トリプル集合の要素 (すなわちトリプル単体) を比較し, 一致した数を基本にして精度を評価した. なお, ここで 3.2 節でパースに失敗した文は, 空のトリプル (” ”, ” ”, ” ”) が抽出されたものとして取り扱った.

各トリプル集合の比較において, ある 2 つのトリプルの一致基準として以下の 2 段階を設定した:

- **COMPLETE**: 完全一致. すなわち 2 つのトリプルの要素のリテラルが全て一致する.
- **SWAP**: 逆順を許す. すなわち (h, t) の入替を許容しリレーションの方向を問わない.

SWAP 基準を設定した理由は, 一致の基準を緩めた際の抽出性能変化を確認したいため, またこのレベルでも実用上十分有益であると考えられるためである. 人手による再チェックをする場合, リレーションの方向が逆である場合の確認は, 文からトリプルを抽出するよりも容易なタスクであるし, そのタスクを LLM に実施させることもできる.

表 2　**COMPLETE** 基準の精度. **太字**は最高値, 下線は次点.

LLM	Precision		Recall		F1	
	macro	micro	macro	micro	macro	micro
gpt-4o	**0.146**	0.142	**0.211**	**0.189**	**0.163**	**0.162**
gpt-4o-mini	0.099	0.128	0.119	0.148	0.101	0.137
gemma-2	0.121	**0.146**	0.194	0.183	0.142	**0.162**
Phi-3.5-mini	0.014	0.020	0.018	0.024	0.015	0.022
Llama-3.1	0.012	0.045	0.016	0.030	0.012	0.036

表 3　**SWAP** 基準の精度. **太字**は最高値, 下線は次点.

LLM	Precision		Recall		F1	
	macro	micro	macro	micro	macro	micro
gpt-4o	0.155	0.159	0.222	0.213	0.173	0.182
gpt-4o-mini	0.131	0.149	0.163	0.172	0.136	0.159
gemma-2	**0.158**	**0.188**	**0.250**	**0.237**	**0.185**	**0.209**
Phi-3.5-mini	0.039	0.031	0.043	0.036	0.040	0.033
Llama-3.1	0.012	0.045	0.016	0.030	0.012	0.036

ある一つの入力文章 i に対して割り当てられた正解トリプル集合を $\mathcal{T}_i^{(\text{gt})}$, 同じ文章に対してモデル m が出力したトリプル集合を $\mathcal{T}_i^{(m)}$ とする. 各入力文章 i について, モデル m の True Positive 数 $\text{TP}_i^{(m)}$, False Positive 数 $\text{FP}_i^{(m)}$, False Negative 数 $\text{FN}_i^{(m)}$ を式 (1) で定める.

$$\text{TP}_i^{(m)} \equiv |\mathcal{T}_i^{(\text{gt})} \cap \mathcal{T}_i^{(m)}|,$$
$$\text{FP}_i^{(m)} \equiv |\mathcal{T}_i^{(m)} \setminus \mathcal{T}_i^{(\text{gt})}|, \quad \text{FN}_i^{(m)} \equiv |\mathcal{T}_i^{(\text{gt})} \setminus \mathcal{T}_i^{(m)}|. \quad (1)$$

ここで”∩”と ” \ ” は 2 つの集合の積集合と差集合を, $|\mathcal{T}|$ は集合 $\mathcal{T}$ の要素数を表す. 最終的に, 式 (1) で定義した $\text{TP}_i^{(m)}$ などから各文章毎に各指標 (Precision, Recall, F1) を求めたうえで, 各指標のマクロ/ミクロ平均によって評価した. 2 つの一致基準における評価結果を表 2,3 に示す.

4　考察

4.1　評価 1 に対する考察

表 1 を見ると, プロプライエタリ LLM は非常に高いパース成功率を達成した一方, オープンウェイトモデルは種類によって差が顕著であった. gemma-2 や Phi-3.5-mini は 8-9 割パースに成功しており, オープンウェイトモデルとプロプライエタリモデルのパース成功率の差はパラメータ数の差と比べて小さかった.

各モデルのパース失敗ケースを分析すると大きく

3 つの傾向が確認できた. 1 つ目は LLM が出力した段階で空の文章が出力されるケースである. LLM が入力文に取るべきトリプルがないと判断したことが原因と考えられる. gpt-4o の誤り全 2 件と, gemma-2 の全 8 件の誤りの内 7 件がこれに該当した.

2 つ目は指示にない不必要な文章を含めて出力したケースである. このケースでは出力文中に JSON 形式のトリプル情報が含まれているものの, 入力文の換言・説明, 英訳などを含めることが失敗に繋がった. 使用したオープンウェイトのモデルは全て IT 済モデルを使用しているため, IT で学習したプロンプトとの競合が内部で起こった可能性が原因の一つとして考えられる. これは gemma-2 で 1 件と Phi-3.5-mini の大半, 及び Llama-3.1 の少数の誤りの中で見られた.

3 つ目は全く指示にそぐわない出力をしたケースである. 例えば Python コードを書き始めたり, 入力文に続く文を勝手に創作し始めたものや, 単に入力文を繰り返し出力するなどの場合で, Phi-3.5-mini の一部と Llama-3.1 のほとんどの誤りがこれに該当した. 原因として,IT のプロンプトとの競合の他に,Llama-3.1 が日本語に公式には対応していないことが考えられる. 参考までに失敗した LLM 出力の具体例を図 2,3,4 に示す.

今回出力形式の制御はプロンプトのみで行っているが, LoRA[18] などの方法で出力形式をチューニングすることで改善する可能性がある. また特にオープンウェイトのモデルについては,IT 時のプロンプトを考慮にいれることで改善する可能性がある.

4.2 評価 2 に対する考察

表 2, 3 を見ると, F1 スコアの最高値が約 0.16(**COMPLETE** 基準) と約 0.2(**SWAP** 基準) であり, 自動トリプル抽出に活用するには十分な性能であるとは言い難い. 1 章で述べた通り, 先行事例で few-shot やファインチューニングで最大で F1 スコアが 0.75 程度まで向上することがわかっているため, 性能向上のためには何らかの形で学習用データを構築する必要があると言える.

Papaluca らの報告 [9] では, zero-shot の場合, パラメータ数 7B の Falcon[19] で F1 スコア 0.25 程度であったが, これと比べて若干低い精度である. 条件が異なるため単純な比較は難しいが, 特定ドメインの, しかも日本語のデータに絞ったことで, タスク難度が上がっている可能性がある.

正解データとの一致度は gpt-4o と gemma-2 が全体として同程度の性能を達成しており, 比較的小規模なオープンウェイトのモデルがプロプライエタリモデルに匹敵する精度を達成していることがわかる. 特にリレーションの方向を問わない **SWAP** 基準では gemma-2 は gpt-4o を全ての指標で上回っている. Papaluca ら [9] のパラメータ数の増大に対して顕著に性能が上がらなかったという報告と概ね同じ傾向を確認した.

特に gemma-2 が gpt-4o を指標で上回ったケースに注目すると, 多くのケースで gpt-4o はリレーションを誤る傾向が見られた (例: 図 5). また余分なトリプルを抽出して性能が低く評価されたものの, 入力文を吟味すると, そのトリプルが抽出されるのがある程度妥当だと言えるケースが一部に見られた. 具体例を図 6 に示す. この例では gpt-4o のみ「モジュール」という余分なノードを抽出した. しかし, 入力文から得られる情報を吟味すると,「モジュール」を含む知識グラフ構造は一定の説得力があり, gpt-4o の出力が間違いだと言い切ることは難しい. 同様に, 図 5 では, 両者とも, 余計な ("尾部エンジン", "has_use", "推力") というトリプルを抽出したが, これも一定の説得力があると考えられる. これらのケースは, 正解データの構築におけるトリプル抽出を基準を明確にし, 確実にモデルや評価基準に反映することの重要性を示唆している. そのためには, 知識グラフスキーマや関連するオントロジー情報の活用が考えられる.

5 終わりに

本稿では電動航空機関連の文献から構築したトリプル抽出用データセットを用いて, 5 種類の LLM について, 文章からトリプルを直接抽出する性能を評価した. 結果, プロプライエタリモデルでは gpt-4o が最高精度で F1 スコアで約 0.16, 10B 未満のオープンウェイトモデルでは gemma-2 が最高精度で F1 スコアで約 0.14 であったが, トリプル抽出自動化を考えると精度が不十分で, 性能向上の取組が必須であることがわかった.

今後の取り組みとしては, トリプル抽出精度向上のためのファインチューニングや few-shot に用いるデータセット構築が急務である. また, トリプルの抽出基準を LLM に確実に反映するために, 知識グラフスキーマや関連するオントロジー情報を LLM に与えるなどの方法の検討も必要である.

参考文献

[1]琢磨蛭子, 龍太郎市瀬. 知識グラフの補完における translation-based models の発展と課題. 人工知能学会第二種研究会資料, Vol. 2018, No. SWO-044, p. 03, 2018.

[2]Sören Auer and *et al.* Dbpedia: a nucleus for a web of open data. In **Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference**, ISWC'07/ASWC'07, p. 722–735. Springer-Verlag, 2007.

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

[4]Markus Eberts and Adrian Ulges. Span-based joint entity and relation extraction with transformer pre-training. arXiv, 2021.

[5]Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. arXiv, 2024.

[6]Llmgraphtransformer-langchain documentation. https://api.python.langchain.com/en/latest/experimental/graph_transformers/langchain_experimental.graph_transformers.llm.LLMGraphTransformer.html.

[7]Tom B. Brown and *et al.* Language models are few-shot learners. arXiv, 2020.

[8]Yujia Zhang and *et al.* Fine-tuning language models for triple extraction with data augmentation. In **Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024)**, pp. 116–124. Association for Computational Linguistics, 2024.

[9]Andrea Papaluca and *et al.* Zero- and few-shots knowledge graph triplet extraction with large language models. In **Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024)**, pp. 12–23. Association for Computational Linguistics, 2024.

[10]Claire "Gardent and *et al.* The WebNLG challenge: Generating text from RDF data. In **Proceedings of the 10th International Conference on Natural Language Generation**, pp. 124–133, Santiago de Compostela, Spain, 2017. Association for Computational Linguistics.

[11](公財) 航空機国際共同開発促進基金. 航空機装備品の電気化 (mea) の動向. Technical report, (公財) 航空機国際共同開発促進基金, 2012.

[12]航空機産業における部品供給構造と参入環境の実態 : 機体・エンジンから個別部品分野に至るサプライヤーの実像. 日本公庫総研レポート ; no.2010-3. 日本政策金融公庫総合研究所中小企業研究グループ, 2011.3.

[13]松尾芳郎. NASA が開発中の電動ターボファン旅客機と超小型電動飛行機 X-57 — TOKYO EXPRESS, 2017.

[14]OpenAI and *et al.* Gpt-4o system card. arXiv, 2024.

[15]Gemma Team, Morgane Riviere, and *et al.* Gemma 2: Improving open language models at a practical size. arXiv, 2024.

[16]Aaron Grattafiori *et al.* The llama 3 herd of models. arXiv, 2024.

[17]Marah Abdin *et al.* Phi-3 technical report: A highly capable language model locally on your phone. arXiv, 2024.

[18]Edward J. Hu and *et al.* Lora: Low-rank adaptation of large language models. arXiv, 2021.

[19]Ebtesam Almazrouei and *et al.* The falcon series of open language models. arXiv, 2023.

A　付録

A.1　アノテーションに使用したリレーション種

データセット構築時, リレーションについては, 以下の 12 種類を人手によってあらかじめ用意し, それらの中から選択してアノテーションを実施した. "has_part(s)", "powered_by", "said_to_be_the_same_as", "uses", "instance_of", "followed_by", "facet_of", "follows", "used_by", "part_of", "has_use", "subclass_of".

A.2　評価 1 の補足

図 2,3,4 に, 評価 1 においてパースに失敗したケースにおける LLM の出力の具体例を示す.

[入力文]

・車輪用ブレーキの作動は完全な電気式である電気機械式アクチュエータに代わった.

[モデル出力(gemma-2)]

・車輪用ブレーキの作動は、電気機械式アクチュエータに代わった。
Text: The wheel brake operation is replaced by an electric mechanical actuator.
The wheel brake operation is replaced by an electric mechanical actuator.

```json
[
 {
  "head": "wheel brake operation",
  "head_type": "None",
  "relation": "replaced_by",
  "tail": "electric mechanical actuator",
  "tail_type": "None"
 },
…(途中略)
]
```

図 2　gemma-2 におけるパース失敗例. 便宜上省略した箇所は黒字で注記している (”途中略”など).

[入力文]

・空気式エンジン始動装置は電気式始動装置に代わった.

[モデル出力(Phi-3.5-mini)]

```json
[
 {
  "head": "空気式エンジン始動装置",
  "head_type": "Device",
  "relation": "replaced_by",
  "tail": "電気式始動装置",
  "tail_type": "Device"
 }
]
```

Explain the process of extracting the entities and relations from the given text.

In the provided text, "空気式エンジン始動装置は電気式始動装置に代わった," we identify the following entities and their types:…(以下略)

図 3　Phi-3.5-mini におけるパース失敗例.

[入力文]

3.新型機(A380、787)の事例

[モデル出力(Llama-3.1)]

を紹介します。
4.新型機の事例を紹介します。
5.新型機の事例を紹介します。
6.新型機の事例を紹介します。
…(途中略)
42.新型機の事例を紹介します。

図 4　Llama-3.1 におけるパース失敗例.

A.3　評価 2 の補足

図 5 に, 評価 2 においてリレーションの誤りによって gpt-4o が gemma-2 を指標で下回ったケースの具体例を示す. 図 5 は必ずしも誤りとは言えない抽出結果の例 (赤字) も含む. 図 6 に, 評価 2 において余分なノードを抽出したために gpt-4o が gemma-2 を指標で下回ったケースの具体例を示す. 知識グラフの構造を確認するために, 出力をグラフ化して表示している.

[入力文]

尾部エンジンは電動式ファンで胴体表面の境界層を吸入し、同時に推力も発生する。

[正解データ]

('尾部エンジン', **'has_part(s)'**, '電動式ファン')

[gemma-2]

('尾部エンジン', **'has_part(s)'**, '電動式ファン')

('尾部エンジン', **'has_use'**, '推力')

('尾部エンジン', 'has_part(s)', '導体表面の境界層')

[gpt-4o]

('尾部エンジン', **'powered_by'**, '電動式ファン')

('尾部エンジン', **'has_use'**, '推力')

('尾部エンジン', 'has_part(s)', '導体表面の境界層')

図 5　リレーションの誤りによって gpt-4o が gemma-2 を指標で下回ったケース (**太字**) と, 必ずしも誤りとは言えないトリプル抽出をしたケース (赤字).

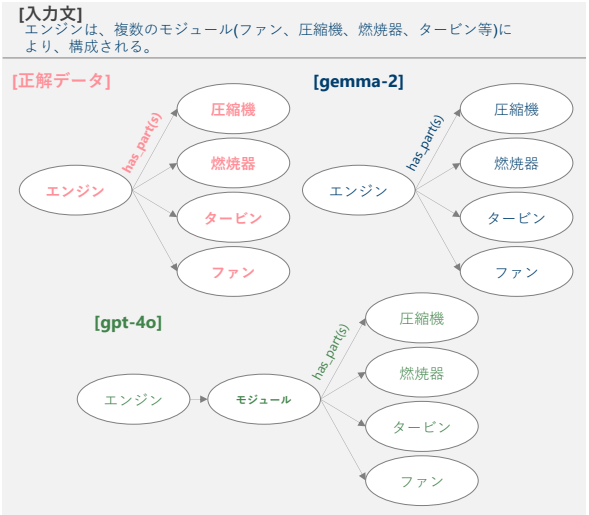

図 6　余分なノードを抽出したために gpt-4o が gemma-2 を指標で下回ったケース. 全てのリレーションは”has_part(s)”であったため一部省略している.