大規模言語モデルの手順型応答を対象としたファクトチェックフレームワークの構築

杣谷星音　河原大輔
早稲田大学
seon-shs@fuji.waseda.jp　dkw@waseda.jp

概要

大規模言語モデル (LLM) は，広範なタスクで優れた性能を示しているものの，依然としてハルシネーションの問題が残されている．この問題の解決策の 1 つとして，自動で LLM の応答をファクトチェックするフレームワークが構築されている．LLM に何らかの手順を尋ねる状況は一般的なものであり，手順型の応答に対しても，ファクトチェックの自動化が必要である．しかし，手順型の応答を対象としたフレームワークはこれまでに存在しない．本研究では，LLM の手順型応答を対象としたファクトチェックフレームワーク **ProcedureFC** を構築する．また，LLM の手順型応答と事実性のラベルによってベンチマークを構築し，提案フレームワークの性能を評価する．

1　はじめに

大規模言語モデル (Large Language Model; LLM) は，多岐にわたるタスクで顕著な性能を示している．質問応答は典型的なタスクの 1 つであり，多くのユーザが LLM に対する質問を通じて，LLM を情報源として活用している．

LLM の進展が注目される一方で，LLM が不正確な応答を生成するハルシネーション (hallucination) の問題が残されている．人手によって LLM の応答の事実性を検証するのは，時間と労力を要する作業であるため，ファクトチェックを自動で行うフレームワークが研究されている [1, 2, 3, 4, 5, 6]．また，そのようなフレームワークを評価するためのベンチマークの研究も行われている [7, 8, 9, 10]．

ユーザが LLM に対して行う質問の例として，手順に関するものが挙げられる．LLM の応答を基に，ユーザが手順を実行するという状況は，ごく一般的なものである．そのため，手順に関する LLM の応答に対しても，自動ファクトチェックのフレームワーク，およびベンチマークが構築される必要がある．しかし，手順に関する応答を対象とした研究は，現時点で確認できていない．

本論文では，手順に関する LLM の応答に特化した新たなファクトチェックフレームワークである **ProcedureFC** を提案する．また，ProcedureFC の性能を評価するためのベンチマークを構築する．このベンチマークは，手順に関する LLM の応答と，事実性のラベルから構成する．

実験では，既存のファクトチェックフレームワーク [1] をベースラインとして比較を行った．その結果，手順に関するハルシネーションの検出において，提案手法が有効であることが示された．

2　関連研究

2.1　自動ファクトチェックを対象としたベンチマーク

ファクトチェック手法の訓練や評価を可能にするためのデータセットおよびベンチマークが構築されている．代表的なベンチマークである FEVER [7] を始めとして，特定のドメインに特化したもの [8] や，より複雑なタスクを設定したもの [9, 10] も存在する．これらのベンチマークは一般に，検証対象となる文章と，事実性を示すラベルのペアで構成されている．検証可能な事実表現の最小単位は，主張 (claim) と呼ばれる．

このように様々なベンチマークが構築されているが，手順に関する文章で構成されたものはない．

2.2　自動ファクトチェックのフレームワーク

ファクトチェックを自動化するフレームワークも構築されている．それらの多くは，主張の抽出，証

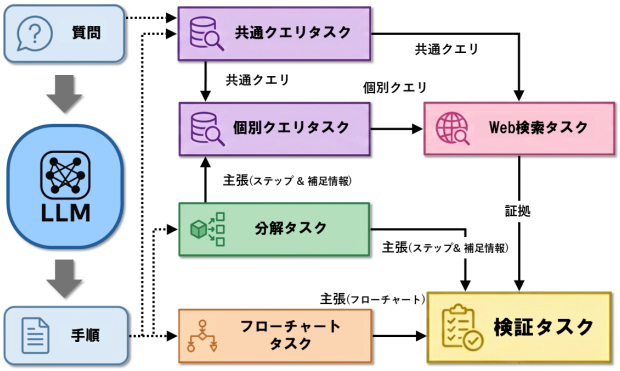

図 1　ProcedureFC の概観

拠の収集，主張の検証というフローで構成され，内部で LLM を使用する．

FACTSCORE [2] で提案されたシンプルなフレームワークを始めとして，証拠収集に使うツールの幅を広げたもの [3] や，フローを細分化して説明可能性を向上したもの [4]，効率的な LLM の呼び出しを行い，コストを削減するもの [1] などが存在する．

また，主張の抽出を工夫し，主張同士の依存関係の保持や，検証順序のプランニングを可能にしたフレームワーク [5, 6] も構築されている．

このように様々な自動フレームワークが構築されているが，依然として手順に関する文章を対象としたものはない．

3　ProcedureFC の構成

ProcedureFC の概観を図 1 に示す．ProcedureFC のフレームワークは，2 つのデータを入力として受け取る．1 つはユーザが LLM に尋ねた質問（プロンプト）であり，もう 1 つはユーザの質問に対する LLM の応答である．フレームワークは 6 つのタスクからなる．Web 検索タスクには Google の検索 API を使用し，それ以外の 5 つのタスクには LLM を使用する．最後の検証タスクにおいて，抽出された各主張に True（真）か False（偽）のラベルを付与する．なお，ProcedureFC は英語の質問と手順型応答を対象とする．質問，手順型応答，タスクの出力例の一部を付録 B に掲載する．

3.1　用語の定義

手順とは，何らかの目的を達成するために行われる，一連の行為である．そして，手順を構成する個々の行為をステップと呼ぶ．ステップには順序がある．また，あるステップの成果物が，後のステップで使用される等の，依存関係が認められる場合もある．なお，手順に関する文章は，ステップ以外に，アドバイスや注意事項などの補足情報を含むことが多い．本研究ではステップと補足情報の両方を検証対象の主張とする．

3.2　フローチャートの作成

ステップが持つ順序と依存関係の正しさを検証するため，フローチャートを作成する．文章による手順の記述に比べて，フローチャートによる記述は，ステップの順序と依存関係をより簡潔に表現する．ProcedureFC は，後のタスクで抽出するステップと補足情報に加え，フローチャートも 1 つの主張として扱う．

フローチャート作成タスクでは，手順型応答を入力とし，1 つのフローチャートを出力する．フローチャートは Mermaid 記法によって記述し，各ノードがステップを，ノード同士を繋ぐエッジが順序と依存関係を表現する．

3.3　手順の分解

フローチャートは手順全体の構造を表現するが，元の文章に含まれていた細かな情報や，補足情報は失われる．手順型応答に含まれるすべての主張を厳密に検証するため，フローチャートの作成とは独立して手順の分解を行う．

手順の分解は，タスクとしての難易度の高さを鑑みて，2 つの工程を持つパイプラインとして行う．2 つ目の工程の結果が分解タスクの最終的な出力となる．入力は手順型応答の文章である．

1 つ目の工程では，手順型応答に含まれるステップおよび補足情報を 1 つずつ抜き出し，箇条書きで整理する．このとき，ステップと補足情報は区別する．ステップに関しては，抜き出す際に動詞句の抽出もあわせて行う．これにより，各ステップが単一の動作のみを含むように分解が行われる．

2 つ目の工程では，1 つ目の工程で整理したステップおよび補足情報を，適切な表現に書き換えた上で，JSON フォーマットに整形して出力する．JSON フォーマットへの準拠は，出力をシステム上で扱いやすくする目的がある．表現の書き換えについては，まず，ステップと補足情報の両方に対して，それ単体で正しく検証が行えるよう，周辺情報を付加する．さらにステップに関しては，「ある動作を行うことで，ある結果が得られる．」という形式の文へと書き換える．ここで言う動作とは，1 つ目の

工程で抽出した動詞句のことである．ステップの書き換えを行うことで，次のステップとの繋がりを文の表現に含めることができ，検証精度が向上する．

3.4 Web 検索クエリの作成

従来手法では主張の文そのものを検索クエリに使用しているが，マイナーなトピックの場合に，検索結果が 1 件も得られないことがある．また，具体性が過度に高いクエリでは，手順全体を包括的に含む情報の取得も期待できない．本研究では，これらの問題に対処するため，2 種類の検索クエリを作成する．

3.4.1 共通クエリ

共通クエリは，手順全体を包括する情報の取得を目的とした，2 つで 1 組の検索クエリである．

共通クエリ作成タスクでは，ユーザの質問と手順型応答の両方が与えられる．初めにフィールド，質問対象，質問対象の上位語，という 3 つの語句を取得する．フィールドは，ユーザと LLM の対話における主題を示す．例えばゲームタイトルやソフトウェアの名称がこれに該当する．質問対象は，ユーザの質問の直接的な対象を示す．質問対象の上位語は，質問対象の上位概念を示す．

取得した 3 つの語句を組み合わせ，2 つの共通クエリを作成する．1 つ目の共通クエリは，「フィールド　質問の対象」として構成する．これは，人が Web 検索を行う際に入力する，自然なクエリの構成である．2 つ目の共通クエリは，「フィールド　質問の対象の上位語」として構成する．この 2 つ目の共通クエリは，質問の対象を理解する上での前提知識を証拠に含めることを目的として作成する．

3.4.2 個別クエリ

一部の主張は，共通クエリで収集した情報だけでは検証に不十分となる．主張に固有の情報を収集するため，個別クエリを作成する．個別クエリは主張ごとに作成する．なお，効率の良い情報収集を行うため，共通クエリだけで検証可能かどうかを LLM に判断させ，検証可能と判断した場合は個別クエリの作成は行わない．

3.5 主張の検証

ProcedureFC では，分解タスクによって抽出された主張のほか，作成したフローチャートも 1 つの主張として検証する．主張の検証では，対象となる主張と収集した証拠が入力として与えられる．入力を基に，判定結果のラベルと判定理由の文章を出力する．理由も付随して出力することで，Chain-of-Thought に類似する効果と，ファクトチェックの説明性向上が期待できる．判定結果のラベルは，True か False の 2 値で表す．手順全体に対する最終的な評価は，すべての主張のラベルに論理積を適用したものとする．

共通クエリの情報は，すべての主張の検証に用いられる．個別クエリの情報は，該当する主張の検証だけに用いられる．つまり，フローチャートの検証には共通クエリの情報のみ，それ以外の各主張には共通クエリの情報に合わせて個別クエリの情報が用いられる．

4 評価ベンチマークの構築

ProcedureFC の性能を評価するため，ベンチマークを構築する．評価ベンチマークは，ユーザの質問，LLM の手順型応答，事実性のラベルによって構成される．ラベルは True か False のいずれかである．

本研究は，LLM の応答をファクトチェックの対象としている．そのため，一般的なユーザの質問に近い入力によって応答を生成し，ベンチマークを構成する．まず，一定量の質問文を得るため，WikiHow[1)] の記事タイトルを使用した．WikiHow は様々な手順を紹介する Web サイトであり，その記事のタイトルは，「○○の手順」という形式で書かれていることが多い．このタイトルを取得し，LLM を用いて手順を尋ねる質問文に変換した．このように用意した質問文を再度 LLM に入力し，手順型応答を得た．事実性のラベルは，人手によるアノテーションで付与している．質問文の作成と応答の取得には，GPT-4.1 を使用した．なお，検証に適さない曖昧な表現が応答に含まれる場合などは，人手での編集を行っている．質問，応答，ラベルの件数は，合計 50 組である．

5 実験

5.1 実験設定

ProcedureFC のフレームワークとしての有効性と，内部で使用する LLM が持つ推論能力の関係を

1） https://www.wikihow.com/Main-Page

表 1 実験結果

手法	Acc.	Pre.	Rec.	F1
FASTFACT (GPT-5.1)	0.64	0.80	0.53	0.64
ProcedureFC (GPT-5.1)	**0.76**	0.76	0.87	**0.81**
ProcedureFC (gpt-oss-120b)	0.68	0.69	0.83	0.76
ProcedureFC (Qwen3-30B)	0.58	0.59	0.97	0.73

調べるため，パラメータ規模の異なる 3 つの LLM (GPT-5.1, gpt-oss-120b, Qwen3-30B-A3B-Instruct-2507) で比較する．ベースライン手法には，FASTFACT [1] を採用する．結果の分類指標は，提案手法がハルシネーションの検出器であるという位置付けの下，False（偽）の判定を陽性とする．その上で，正解率 (Acc.)，適合率 (Pre.)，再現率 (Rec.)，F1 スコア (F1) の 4 つの指標で評価する．

なお，FASTFACT は一部の実装の変更を行っている．まず，FASTFACT で元々 Web 検索 API として使用されていた Jina Reader API[2] を，Google Custom Search JSON API[3] に変更した．さらに，フレームワーク内で使用される LLM を GPT-5.1 に変更した．

5.2 実験結果

実験結果を表 1 に示す．正解率および F1 スコアが最も高かったのは，GPT-5.1 を使用した ProcedureFC である．0.8 に近い正解率と F1 スコアを記録しており，提案手法はハルシネーションの検出において有効であると言える．

GPT-5.1 を使用した FASTFACT の正解率および F1 スコアは，同じ GPT-5.1 を使用した ProcedureFC の結果を下回った．内訳を見ると再現率が大きく低下している．従来の手法は，手順型応答のハルシネーション検出に有効でないことが分かる．

gpt-oss-120b を使用した ProcedureFC は，全体的な精度は GPT-5.1 を使用した場合に劣るものの，一定の性能を示している．特に，再現率の低下が軽微である．このことから，提案手法は高度な推論能力を持つ LLM に依存しないと考えられる．

一方，Qwen3-30B-Instruct を使用した ProcedureFC では，GPT-5.1 を使用した場合と比べて，正解率および F1 スコアが大きく低下している．適合率と再現率も大きく偏っており，適切な判定ができていないことが分かる．このことから，本手法が許容する推論能力の低さには限度があり，それを下回る LLM を使用する場合は有効性が失われると考えられる．

2） https://jina.ai/ja/reader/
3） https://developers.google.com/custom-search/v1/overview

表 2 追加実験の結果

評価手法	Acc.	Pre.	Rec.	F1
GPT-5.1 + reasoning (high)	**0.84**	0.84	0.90	**0.87**
Sonnet 4.5 + thinking	0.72	0.74	0.83	0.78
ProcedureFC (GPT-5.1)	0.76	0.76	0.87	0.81

5.3 追加実験

昨今の LLM の多くは，推論モードや思考モードを有している．中には，Web 検索機能をツールとして呼び出し，推論過程の中で動的に情報を収集する LLM も存在する．追加実験として，このような推論，思考モードを持つ LLM のファクトチェック性能を検証する．

実験設定 比較する LLM として，GPT-5.1 と Claude Sonnet 4.5 を採用する．これらの LLM にユーザの質問と手順型応答を与え，Web 検索を活用しながらこの応答の事実性を判定するよう指示する．出力は手順全体に対する判定結果とその理由とし，検証の具体的な工程は一切指示しない．

実験結果 推論，思考モード，および Web 検索機能を有効化した 2 つの LLM と，GPT-5.1 を使用した ProcedureFC の結果を表 2 に示す．推論モードを高に設定した GPT-5.1 が，正解率と F1 スコアの両方で最も優れていた．Claude Sonnet 4.5 も，提案手法に迫る性能を記録している．この結果から，推論モードや思考モードによる動的な情報収集が，手順のファクトチェックにおいて有効であることが分かる．特に，文章全体を見渡しながら検証順序のプランニングを行うことで，検証精度が高まっていると予想される．

6 おわりに

本研究では，LLM の手順型応答に特化した自動ファクトチェックフレームワークを構築した．また，評価ベンチマークの構築を行い，提案手法の有効性を調べた．評価実験では，提案手法が手順型応答のファクトチェックに有効であることを示した．

一方で，商用 LLM の推論，思考モードと提案手法を比較した追加実験では，分解型フレームワークの限界が示された．今後は，より大規模なベンチマークでの手法の評価と，従来のフレームワークとは根本的に異なる手法の提案が望まれる．

謝辞

本研究は JSPS 科研費 JP24H00727 の助成を受けて実施した．一部の実験には，産総研及び AIST Solutions が提供する ABCI 3.0 と，東京科学大学のスーパーコンピュータ TSUBAME4.0 を利用した．ABCI 3.0 は「ABCI 3.0 開発加速利用」の支援を受けて利用した．

参考文献

[1] Yingjia Wan, Haochen Tan, Xiao Zhu, Xinyu Zhou, Zhiwei Li, Qingsong Lv, Changxuan Sun, Jiaqi Zeng, Yi Xu, Jianqiao Lu, Yinhong Liu, and Zhijiang Guo. FaStFact: Faster, stronger long-form factuality evaluations in LLMs. In **Findings of the Association for Computational Linguistics: EMNLP 2025**, pp. 23814–23854, 2025.

[2] Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In **Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing**, pp. 12076–12100, 2023.

[3] I-Chun Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, Pengfei Liu, et al. Factool: Factuality detection in generative ai–a tool augmented framework for multi-task and multi-domain scenarios. **arXiv preprint arXiv:2307.13528**, 2023.

[4] Yuxia Wang, Revanth Gangi Reddy, Zain Muhammad Mujahid, Arnav Arora, Aleksandr Rubashevskii, Jiahui Geng, Osama Mohammed Afzal, Liangming Pan, Nadav Borenstein, Aditya Pillai, Isabelle Augenstein, Iryna Gurevych, and Preslav Nakov. Factcheck-bench: Fine-grained evaluation benchmark for automatic fact-checkers. In **Findings of the Association for Computational Linguistics: EMNLP 2024**, pp. 14199–14230, 2024.

[5] Liangming Pan, Xiaobao Wu, Xinyuan Lu, Anh Tuan Luu, William Yang Wang, Min-Yen Kan, and Preslav Nakov. Fact-checking complex claims with program-guided reasoning. In **Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 6981–7004, 2023.

[6] Yani Huang, Richong Zhang, Zhijie Nie, Junfan Chen, and Xuefeng Zhang. A graph-based verification framework for fact-checking. **arXiv preprint arXiv:2307.13528**, 2025.

[7] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and VERification. In **Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)**, pp. 809–819, 2018.

[8] David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. Fact or fiction: Verifying scientific claims. In **Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)**, pp. 7534–7550, 2020.

[9] Yichen Jiang, Shikha Bordia, Zheng Zhong, Charles Dognin, Maneesh Singh, and Mohit Bansal. HoVer: A dataset for many-hop fact extraction and claim verification. In **Findings of the Association for Computational Linguistics: EMNLP 2020**, pp. 3441–3460, 2020.

[10] Rami Aly, Zhijiang Guo, Michael Sejr Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. FEVEROUS: Fact extraction and VERification over unstructured and structured information. In **Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)**, 2021.

表 3　アブレーション研究結果

設定	Acc.	Pre.	Rec.	F1
標準	**0.76**	0.76	0.87	**0.81**
分解なし	0.68	0.85	0.57	0.68
主張書き換えなし	0.70	0.70	0.87	0.78
個別クエリなし	0.72	0.74	0.83	0.78

A　アブレーション研究

A.1　実験設定

すべての設定で GPT-5.1 を使用した．各設定について，標準は ProcedureFC の完全なフレームワークを指す．分解なしは，手順の分解をせず，文章全体を 1 つの主張として検証する．主張書き換えなしは，手順の分解で 2 つ目の工程として行う文の書き換えを行わず，元の文章から抜粋した表現を直接検証タスクに入力する．個別クエリなしは，共通クエリのみで Web 検索を行い，そこで得られた情報ですべての主張を検証する．

A.2　実験結果

アブレーション研究の結果を表 3 に示す．

分解による主張の抽出を行わない場合，正解率および F1 スコアが大きく低下する．内訳を見ると，再現率が大きく低下している．これは，手順全体を見るだけでは，細部の誤りを検出できないことを示しており，先行研究の知見と一致している．

主張の文の書き換えを行わない場合，正解率および F1 スコアがわずかに低下する．内訳を見ると，適合率が低下している．「ある動作を行うことで，ある結果が得られる．」という文への変換は，そのステップの役割や，次のステップとの関係などを，適切に主張に含める効果があると考えられる．

個別クエリの生成を行わない場合も，正解率および F1 スコアがわずかに低下する．共通クエリで得られる証拠だけでは，正しく検証できなかった主張が存在することを示している．

B　入力および出力の例

入力例 1　質問の例

```
Please explain the steps to play Codenames one by
    one.
```

入力例 2　手順型応答の例

```
1. **Form Teams:**
    - Divide players into two teams. Each team needs
        a Spymaster and at least one Field
        Operative.（中略）
4. **Determine Starting Team:**
    - The key card will indicate which team starts
        by a border color around the grid (red or
        blue). The starting team has an extra agent
        to find.（中略）
```

出力例 1　フローチャートの例

```
%% How to play Codenames
flowchart TD
    A["Form Teams: two teams with a Spymaster and
        Field Operatives"] --> B["Prepare the Game
        Board: place 25 codename cards (5x5
        layout)"]
    B --> C["Select a Key Card: shows team agents,
        neutrals, and assassin"]（中略）
```

出力例 2　分解の例

```
<decomposition>
1. (step) Form two teams, ensuring each team has one
    Spymaster and at least one Field Operative.
    {action: "form"}（中略）
6. (info) The starting team has one extra agent to
    find compared to the other team.（中略）
</decomposition>
<json>
{
  "statements": [
    {
      "number": 1,
      "sentence": "In the board game Codenames, by
          forming two teams （中略）, players can
          set up the basic team structure needed to
          play."
    },（中略）
    {
      "number": 6,
      "sentence": "In the board game Codenames, the
          team indicated as starting on the key
          card has one extra agent to find compared
          to the other team."
    },（中略）
</json>
```