推論時スケーリングにおける検証戦略の比較分析

相田太一　矢野太郎　石橋陽一　小山田昌史
NEC データサイエンスラボラトリー
{taichi-aida,taro_yano,yoichi-ishibashi,oyamada}@nec.com

概要

大規模言語モデルの推論時スケーリングでは、生成された応答集合から最適な応答を選択できるかが性能向上の鍵となっている。本研究では、大規模言語モデルによって応答を評価する**検証**に焦点を当て、検証性能の改善に必要な要素について網羅的な分析を行った。具体的には、典型的な 2 つの戦略である複数の検証結果を並列に生成する戦略と、単一の検証結果を逐次的に延長する戦略、そして両者を組み合わせる戦略を比較した。実験の結果、並列に生成する戦略がより効果的であることがわかった。さらに、検証性能の向上には、検証時に生成されるトークン数ではなく、生成された検証が有する多様性と強く関連していることを明らかにした。

1　はじめに

近年の大規模言語モデル（Large Language Model; LLM）は、単一の応答を一度で生成するだけでなく、応答生成における推論の過程をさらに展開することで、推論能力を大きく向上させる手法が広く用いられるようになっている。このように、モデルに対して推論時に追加の計算資源を割り当てる枠組みは、推論時スケーリング（Test-Time Scaling; TTS）として体系化されている。この枠組みは大きく、（1）推論過程を直列に延長する方向 [1] と、（2）複数の推論結果を並列に生成する方向 [2] に分けられる。TTS はモデルの再学習を必要とせず、必要に応じて推論時の計算量を柔軟に増減できるため、計算資源の観点からも実用的であることが示されている [3]。

一方で、複数の応答候補を生成する TTS では、生成された応答集合の中から最適な応答を正しく選択する必要が生じる。この選択は、報酬モデル [4] や LLM-as-a-Judge によるスコアリング [5]、あるいは生成した LLM 自身による評価 [6] に基づく検証として実装されることが多い。しかし、先行研究では、候補集合に正しい応答が含まれていても、検証でそれを適切に選択することができないことがある [7]。このような問題を解消するため、様々な条件での検証戦略について分析が進められている [8, 9, 10, 11, 12] が、モデルや検証戦略が固定されており、網羅的な分析が不足している。

そこで本研究では、TTS の要となる検証性能の改善に必要な要素について、モデル間・戦略間で網羅的に分析を行った。具体的には、典型的な 2 つの戦略である、複数の検証結果を並列に生成する戦略と単一の検証結果を逐次的に延長する戦略、および両者を組み合わせる戦略を比較した。実験の結果、単一の検証過程を深く延長するよりも、複数の検証結果を並列に生成する方が、一貫して高い検証性能を示すことがわかった。また、従来は生成されるトークン数と性能のスケーリング則が知られていたが、少なくとも検証タスクにおいては、**生成される思考の多様性**が、性能のスケーリングにより重要である事を新たに発見した。

2　関連研究

推論時スケーリング　推論時に多くの計算コストを消費することで推論性能を向上させる TTS に関する戦略が多く提案されている。TTS は、複数個の回答候補を生成し最も良い候補を多数決で選択する Self-Consistency [13] や報酬モデルを使って選択する Best-of-N のような並列の戦略と、モデル自身によって回答の逐次リファインメントを行う Self-Refine [14] や、モデルが回答を終了しようとしたときに “*wait*” を追加して再考させる Simple test-time scaling [15] のような直列の戦略に大別される。また、最近では並列と直列を組み合わせたハイブリッドな戦略 [16, 17, 18, 19] も提案されている。一方でこれらの手法が検証タスクにおいてどの程度有効であるか、また、そのスケーリングの特性について十分調査されていない。

LLM-as-a-Judge　自然言語処理における要約や対話、質問応答などの生成タスクにおける評価指標

として BLEU [20], ROUGE [21], CIDEr [22] のような字句的な評価指標と、BERTScore [23], COMET [24] のような意味的な評価指標が用いられてきたが、近年では多様な正解を許容するオープンエンドタスクの評価のために、LLM によって回答を評価する LLM-as-a-Judge [5] が広く用いられている。また、LLM-as-a-Judge における推論時スケーリング手法 [10, 9] が提案されているが、網羅的な比較研究とスケーリング挙動の十分な理解が得られていない。

3 実験

3.1 検証戦略

本研究における検証とは、ある問題 q と生成モデルから得られた応答 r および評価基準などの補助情報 u から構成される検証プロンプト $p(q, r, u)$ を入力とし、検証用の LLM によって応答の妥当性を表すスコア s を出力する過程を指す。ここで、検証用の LLM は単にスコアを直接生成するのではなく、その判定に至った経緯 t も生成する。したがって、今回の検証過程は $(t, s) = \mathrm{LLM}(p)$ のように表せる。

本研究で用いる並列方向の検証（Parallel Verification）と逐次方向の検証（Sequential Verification）を示す。また、今回は双方を組み合わせた検証戦略（Hybrid Verification）についても実験を行った[1)]。

Parallel Verification ($\mathsf{para}(n, 1)$): n 個の検証結果を独立に並列生成し、それぞれの出力スコアの多数決により最終スコアの予測を行う（式 1）。

$$(t_i, s_i) = \mathrm{LLM}(p), \quad i = 1, \ldots, n,$$
$$\hat{s} = \mathrm{Majority}(s_1, \ldots, s_n) \tag{1}$$

本手法は、検証の多様性を増やして単一の検証結果への依存性を低減することを目的とする [7, 11]。

Sequential Verification ($\mathsf{seq}(1, n)$): 単一の検証過程を逐次的に $n-1$ 回再考させて最終スコアを得る。具体的には、$n-2$ 回生成した予測スコア s_j を一度消去し、“*wait*” で置き換えて続きを生成させることで合計 $n-1$ 回再考させる（式 2）。

$$(t_1, s_1) = \mathrm{LLM}(p),$$
$$(t_j, s_j) = \mathrm{LLM}(p \oplus t_{j-1}), \quad j = 2, \ldots, n \tag{2}$$

本手法は直前の検証を引き継ぎ、より詳細な推論を通じて判断を洗練させることを意図している [15]。

Hybrid Verification ($\mathsf{hybrid}(p, s)$): p 個の検証過程を並列に生成した後、各検証過程を $s-1$ 回逐次的に延長し、最終的に多数決によって予測を行う。これは、式 3のように定式化できる。

$$(t_{i,1}, s_{i,1}) = \mathrm{LLM}(p), \quad i = 1, \ldots, p,$$
$$(t_{i,j}, s_{i,j}) = \mathrm{LLM}(p \oplus t_{i,j-1}), \quad j = 2, \ldots, s,$$
$$\hat{s} = \mathrm{Majority}(s_{1,s}, \ldots, s_{p,s}) \tag{3}$$

本手法は、並列・逐次の両方に計算資源を配分した場合の振る舞いを分析するための設定である。

これらの手法により、同一の計算予算下において、検証における計算資源の配分方法が検証性能に与える影響を体系的に比較することが可能となる。

3.2 実験設定

評価には BigGenBench [25] を用い、5 段階の人手評価 (1 から 5 までで 5 が最良) との Pearson 相関を検証性能の指標とした。BigGenBench は数学や論理問題などの推論カテゴリに加えて、指示追従やツール利用など 9 つのカテゴリ（A 節）からなる網羅的なベンチマークである。

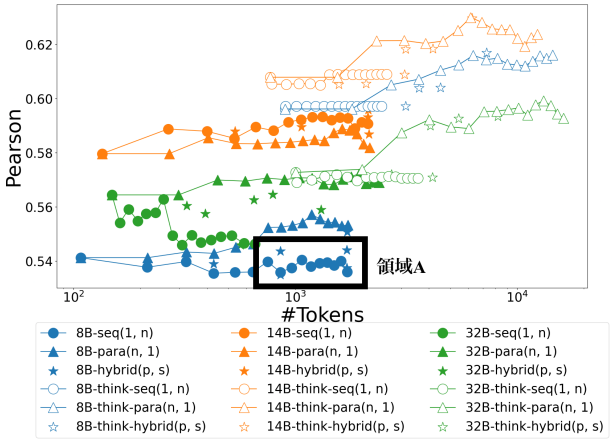
検証モデルには Qwen3 シリーズを用い、8B、14B、32B の各モデルサイズについて実験を行った。各モデルについて、推論過程を省略する non-think モードと、検証時に明示的な推論過程を生成する think モードの両方を評価した。

検証においては、問題文 q と評価対象の応答 r だけでなく、参照回答（5 点相当）および評価基準（rubric）からなる補助情報 u を用いて検証プロンプト p を作成して検証モデルに与え、フィードバック t と 1–5 のスコア s の出力を求めた。全ての実験において同一の検証プロンプトを用い、検証戦略以外の要因が結果に影響しないよう統一した。

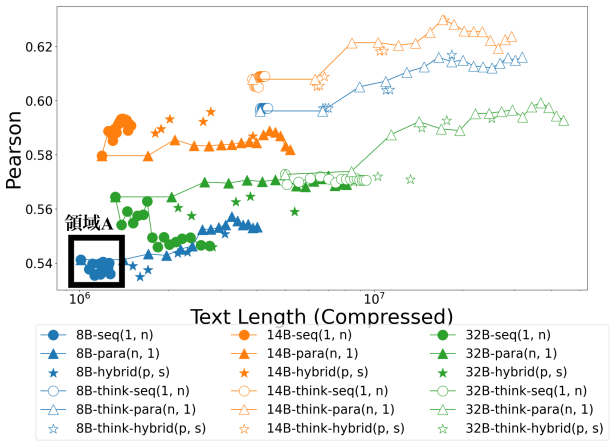
計算予算を公平に比較するため、Parallel Verification および Sequential Verification では最大回数を $n \le 16$ とし、Hybrid Verification では $p \times s \le 16$ となるよう制約を設けた。今回は、評価対象の応答 r について人手のアノテーションによる正解ラベル s^* が付与されているサブセット[2)]を用い、検証結果で得られたスコア $\hat{s}$ と正解ラベル s^* との Pearson 相関で検証性能の評価を行った。

1) Hybrid Verification に似た枠組みとして、Tree-of-Thoughts に代表される探索的推論 [16, 17] があるが、本研究では TTS の基本戦略である深さ [15] と幅 [7, 11] が性能に及ぼす影響を分析するため、両者を制御可能な設定の下で比較する。

2) https://huggingface.co/datasets/prometheus-eval/BiGGen-Bench-Results/viewer/default/human_eval

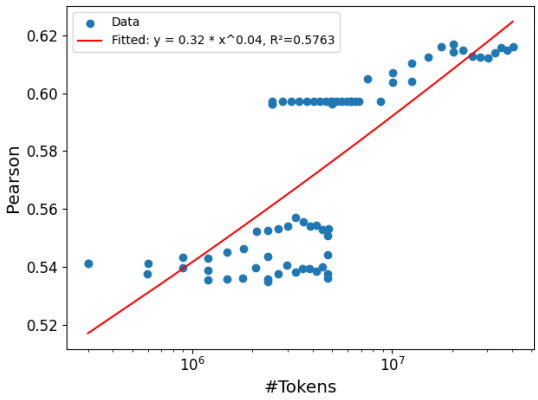


(a) トークン数との関係

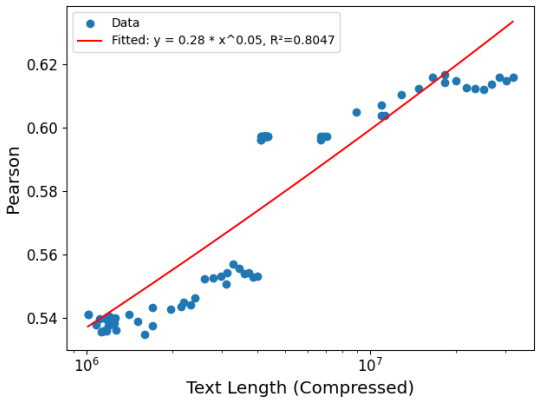


(b) 圧縮後のテキスト長との関係

図 1: 様々なパラメータ・検証戦略での評価結果。



(a) トークン数との関係



(b) 圧縮後のテキスト長との関係

図 2: Qwen3-8B におけるべき乗則フィッティングの結果。トークン数と比べ、gzip 圧縮後のテキスト長は検証性能がべき乗則でスケールするようになる。

3.3 結果

図 1aは、検証時に生成されたトークン数と Pearson 相関の関係を示している。多くの設定において、Parallel Verification は Sequential Verification よりも一貫して高い相関（＝人間に近い高い検証性能）を達成していることが確認できる。また、いずれの検証戦略においても、non-think モードと比較して think モードの方がより多くのトークンを生成し、高い性能を示す傾向が見られる。一方で、トークン数の増加に対する検証性能の改善は必ずしも単調ではなく、同程度のトークン数であっても検証性能に大きなばらつきが存在する。すなわち、トークン数に関しての明確なスケーリングの挙動が確認できない。この結果は、検証性能が単純な生成トークン数によっては十分に説明できないことを示唆している。

4 考察

結果より、TTS における検証では、検証の深さよりも多様性が支配的な要因であることを示唆している。以下では、Parallel Verification と Sequential Verification の差異だけでなく、検証における多様性がどのように観測されうるかについても議論する。

4.1 多様性が支配的要因である理由

Parallel Verification が Sequential Verification を一貫して上回ったことは、検証性能が単一の検証過程をどれだけ延長したかではなく、どれだけ多様な判断基準が導入されたかに強く依存していることを示している。並列に検証を行うことで、同一応答に対して異なる観点や評価基準に基づく判断が生成され、多数決によって個々の誤りが相殺されやすくなる。

一方、Sequential Verification では、単一の検証過程を延長するため、初期段階で形成された判断基準や

誤った前提が後続の推論においても引き継がれやすい。その結果、推論を深めるほど同一の判断構造が強化され、検証判断の多様性が実質的に増加しない可能性がある。この性質は、自己生成応答の検証が困難になると指摘した先行研究 [12] とも整合する。

4.2 仮説：思考の多様性によるスケール則

図 1a を見ると、生成されたトークン数が増えるにつれて検証性能が向上する、従来のスケーリング則が多くの部分で成立していることがわかる。一方で、トークン数が増えても性能が向上しないケースが存在する (図 1aの領域 A など)。これは、例えトークン数が増えても実質的に意味のある思考や情報が増えていなければ性能は向上しないことが原因ではないかと考える[3)]。この直感に基づき、従来のトークン数と性能のスケーリング則ではなく、実質的に意味のある思考の多様性と性能のスケーリング則を提案する。

具体的に思考の多様性を定量化するために、思考を gzip 圧縮した後のトークン数を用いる。図 1bより、gzip 圧縮をすることによって繰り返し出現する多様性の低い思考パターンはより短く圧縮されるため値は向上せず（図 1bの領域 A)、新たな思考パターンが追加されたときにのみ値が向上するという特徴を持つため、gzip 圧縮後のトークン数は思考の多様性の定量化として適切であると考えられる。

4.3 仮説の検証：スケーリング解析

検証性能との関係をより定量的に評価するため、検証性能を目的値 y としてべき乗則 $y = ax^b$ によるフィッティングを行った。ここでは特徴量 x として、（1）生成されたトークン数、（2）gzip 圧縮後のテキスト長、の 2 種類を用いて比較を行う。

図 2aは、トークン数を用いた場合のフィッティング結果を示している。トークン数に対して検証性能は緩やかな増加傾向を示すものの、データ点の分散が大きく、決定係数 R^2 は比較的低い値にとどまっている。この結果は、検証性能が単純なトークン数のみでは十分に説明できないことを示唆している。

一方、図 2bに示す gzip 圧縮後のテキスト長を用いた場合には、トークン数の場合と比較して決定係数 R^2 が大きく改善しており、検証性能がより一貫したスケール則に従って増加することが確認できる。この結果は、検証性能がトークン数そのものよりも、生成されたテキストに含まれる構造的情報量や多様性と強く関連していることを支持している。[4)]

圧縮手法	8B	14B	32B
圧縮なし	0.5763	0.6071	0.7466
圧縮（LZ4）	0.7963	0.6761	**0.8158**
圧縮（Gzip）	**0.8047**	**0.6954**	0.8139

表 1: 検証過程に各圧縮手法を適用した際の、べき乗則フィッティングの決定係数 R^2。どちらの圧縮手法を用いても、適合度が向上する。

4.4 異なる圧縮手法による分析

ここでは、異なる圧縮アルゴリズムを用いても同様にスケーリング則がより良く観測できることを確認する。具体的には gzip 以外に LZ4 アルゴリズムを用いる[5)]。LZ4 を用いた時の圧縮後のテキスト長 x を用いて検証性能 y との関係をべき乗則 $y = ax^b$ によりフィッティングし、適合度を決定係数 R^2 で評価した。

結果を表 1に示す。LZ4 アルゴリズムを用いても同様に決定係数が向上することから、検証性能へのスケーリングは圧縮アルゴリズムに依存しないと考えられる。

5 まとめ

本研究では、推論時スケーリングにおける検証戦略の違いが検証性能に与える影響を分析した。実験から、単一の検証過程を逐次的に延長するより、複数の検証候補を生成する方が一貫して高い検証性能を示した。また、生成トークン数ではなく、検証テキストの多様性に着目することで、検証性能との間にスケーリング挙動を確認できた。

3） 例えば、既に考えたことと意味的に同じ思考や全く同じ文字列を繰り返しても性能は向上しない。

4） 8B だけでなく、他のパラメータでも同様の傾向が観測された。詳細は B 節に示す。

5） LZ4 は Gzip の圧縮手法である LZ77 の原理に基づいて高速化されたアルゴリズムである。

参考文献

[1] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023.

[2] Junpei Komiyama, Daisuke Oba, and Masafumi Oyamada. Best-of-∞ – asymptotic performance of test-time compute, 2025.

[3] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024.

[4] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. Webgpt: Browser-assisted question-answering with human feedback, 2022.

[5] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

[6] Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. Large language models are better reasoners with self-verification, 2023.

[7] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2024.

[8] Yusuke Yamauchi, Taro Yano, and Masafumi Oyamada. An empirical study of llm-as-a-judge: How design choices impact evaluation reliability, 2025.

[9] Chi-Min Chan, Chunpu Xu, Jiaming Ji, Zhen Ye, Pengcheng Wen, Chunyang Jiang, Yaodong Yang, Wei Xue, Sirui Han, and Yike Guo. J1: Exploring simple test-time scaling for llm-as-a-judge, 2025.

[10] Yutong Wang, Pengliang Ji, Chaoqun Yang, Kaixin Li, Ming Hu, Jiaoyang Li, and Guillaume Sartoretti. Mcts-judge: Test-time scaling in llm-as-a-judge for code correctness evaluation, 2025.

[11] Nishad Singhi, Hritik Bansal, Arian Hosseini, Aditya Grover, Kai-Wei Chang, Marcus Rohrbach, and Anna Rohrbach. When to solve, when to verify: Compute-optimal problem solving and generative verification for llm reasoning, 2025.

[12] Jack Lu, Ryan Teehan, Jinran Jin, and Mengye Ren. When does verification pay off? a closer look at llms as solution verifiers, 2025.

[13] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023.

[14] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023.

[15] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025.

[16] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023.

[17] Yuichi Inoue, Kou Misaki, Yuki Imajuku, So Kuroki, Taishi Nakamura, and Takuya Akiba. Wider or deeper? scaling llm inference-time compute with adaptive branching tree search, 2025.

[18] Siddarth Venkatraman, Vineet Jain, Sarthak Mittal, Vedant Shah, Johan Obando-Ceron, Yoshua Bengio, Brian R. Bartoldson, Bhavya Kailkhura, Guillaume Lajoie, Glen Berseth, Nikolay Malkin, and Moksh Jain. Recursive self-aggregation unlocks deep thinking in large language models, 2025.

[19] MohammadHossein Bateni, Vincent Cohen-Addad, Yuzhou Gu, Silvio Lattanzi, Simon Meierhans, and Christopher Mohri. Algorithmic thinking theory, 2025.

[20] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, **Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics**, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[21] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In **Text Summarization Branches Out**, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[22] Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation, 2015.

[23] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2020.

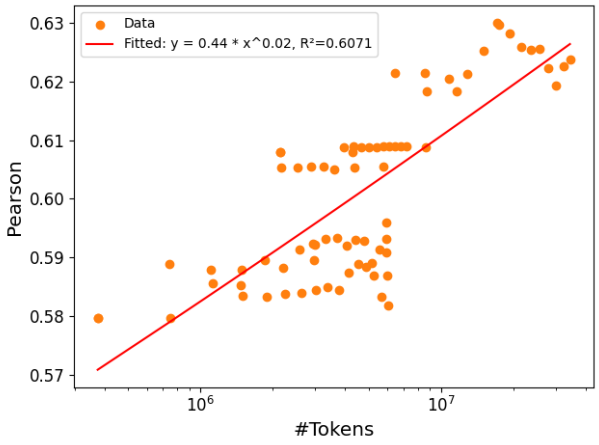
[24] Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. COMET: A neural framework for MT evaluation. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, **Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)**, pp. 2685–2702, Online, November 2020. Association for Computational Linguistics.

[25] Seungone Kim, Juyoung Suk, Ji Yong Cho, Shayne Longpre, Chaeeun Kim, Dongkeun Yoon, Guijin Son, Yejin Cho, Sheikh Shafayat, Jinheon Baek, Sue Hyun Park, Hyeonbin Hwang, Jinkyung Jo, Hyowon Cho, Haebin Shin, Seongyun Lee, Hanseok Oh, Noah Lee, Namgyu Ho, Se June Joo, Miyoung Ko, Yoonjoo Lee, Hyungjoo Chae, Jamin Shin, Joel Jang, Seonghyeon Ye, Bill Yuchen Lin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. The BiGGen bench: A principled benchmark for fine-grained evaluation of language models with language models. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, **Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)**, pp. 5877–5919, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics.

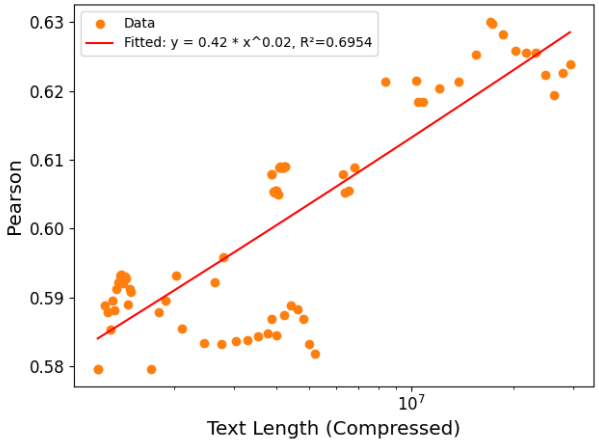
A　BigGenBench のタスクカテゴリ

- Instruction Following（指示遂行）
- Grounding（事実や文脈把握）
- Planning（計画能力）
- Reasoning（推論力）
- Refinement（回答の改善能力）
- Safety（安全性）
- Theory of Mind（他者視点理解）
- Tool Usage（外部ツール利用）
- Multilingual（多言語対応）

B　他パラメータでの解析結果

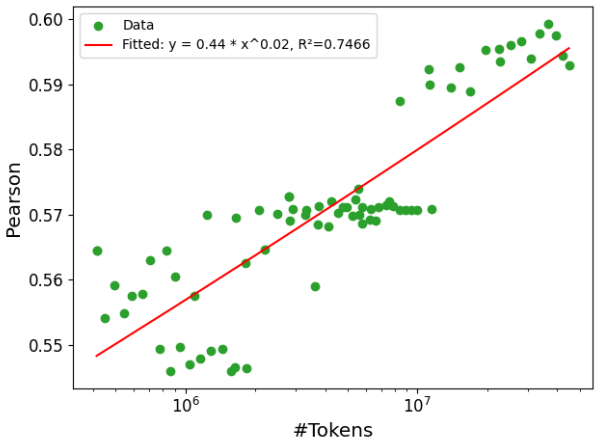


(a) トークン数との関係

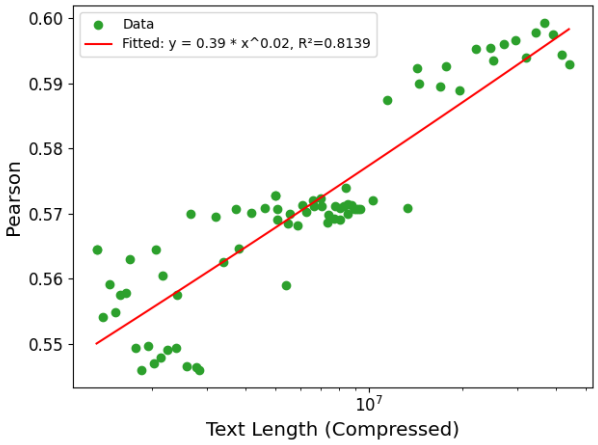


(b) 圧縮後のテキスト長との関係

図 3: Qwen3-14B におけるべき乗則フィッティングの結果。



(a) トークン数との関係



(b) 圧縮後のテキスト長との関係

図 4: Qwen3-32B におけるべき乗則フィッティングの結果。