Pitfalls in Evaluating Inference-time Methods for Improving LLM Reliability

Anonymous authors
Paper under double-blind review

Abstract

Though Large Language Models (LLMs) have demonstrated remarkable capabilities, they are still prone to outputting falsehoods using seemingly persuasive language. Many recent works attempt to address this problem by using LLMs in a framework where a single seed prompt results in a series of interactions involving augmented prompts with an otherwise unchanged LLM, and the results are aggregated with a goal of producing a more reliable output. We consider the replicability and generalizability of evaluations of inference-time methods intended to improve the reliability of responses from a base LLMs. We survey how methods have been evaluated in the literature and find a great variety of benchmarks and models in use. Motivated by this, we conduct our own evaluation to evaluate the effectiveness of a few methods across a range of benchmarks and models. Our evaluation reveals that while these techniques show promise in improving reliability, there is still significant variability in performance across different domains and tasks, and methods that show substantial improvements on weaker base models often do not improve reliability for better base models.

1 Introduction

Large language models (LLMs) have made remarkable progress, but are still unable to reliably provide factual responses. Research on LLM reliability has increased with the widespread excitement about these models and recognition of their current limitations. Methods aiming to improve reliability have been proposed across various stages of the LLM lifecycle, including training, deployment, and inference. Training methods include various methods for fine-tuning and knowledge distillation. Reliability can also be improved by incorporating methods in how an LLM is deployed, such as retrieval-augmented generation and integration with external tools and knowledge sources. All of these types of methods can contribute to improving LLM reliability and are worth investigating. In this study we limit our focus to methods that are intended to improve the reliability of an underlying LLM by changing the way it is used at inference time. Although there is often value in considering the impact of changes in a full system with a particular LLM and on a specific task, many works claim to provide general methods for improving reliability at inference time, with expectations that performance improvements in tested settings generalize to other tasks and different base models. The goal of our work is to understand the robustness and generalizability of inference-time LLM methods.

We focus on approaches that are implemented within the generalized pipeline of a user interacting with an LLM—a user submits a seed prompt that is intended to capture what the user wants from the LLM system, the system performs some automated computations including interactions with an underlying LLM to generate a response to that seed prompt, and then outputs a response to the user. We explore system-level inference-time methods intended to improve reliability, which is a limited, but important, part of the solution space being explored to improve LLM reliability. Many of these methods involve redundancy and variation, combining multiple interactions with the underlying LLM with the goal of deriving a better response than would be obtained through a single, straightforward submission of the prompt. These methods are in general complementary with methods to improve underlying models, but differ in that they can readily be applied to any LLM. They do not rely on any additional training or tuning, or on external knowledge sources or tools.

Contributions. We survey the evaluation of inference-time methods to improve LLM reliability. We analyze how research in this area evaluates the effectiveness of proposed methods (Section 3). We find a broad range of approaches, and a lack of agreement on the benchmarks (Section 3.2) and models (Section 3.3) that are used in these evaluations. Motivated by these findings, we conduct an experiment to comprehensively evaluate a few methods with a group of representative benchmarks and a diverse set of models (Section 4). Our main findings are that claims about the effectiveness of inference-time LLM methods are fragile to both the models and benchmarks used. In particular, methods that result in large improvements for weaker models and on poorly-chosen benchmarks often have disappointing performance when evaluated more comprehensively.

2 Related Work

Many works have conducted evaluations on the reliability of LLMs. Notable examples include Chang et al. (2023) and the HELM project (Liang et al., 2023). Other works have focused on evaluating the reliability of a single LLM. For example, Shen et al. (2023) studied ChatGPT. Yang et al. (2024) focused on analyzing the performance of LLMs on specific downstream tasks. Wang et al. (2024) evaluate particular characteristics of specific models and examined various aspects of trustworthiness, including toxicity, bias, robustness, privacy, ethics, and fairness on GPT-3.5-turbo and GPT-4. Kadavath et al. (2022) studied reliability of LLM responses and whether it is possible to predict whether a model's response is reliable. These works (and many others) share with ours the broader goal of understanding the performance of LLMs, but all of them focus on the unaided model, whereas our focus is on inference-time methods that are intended to improve the reliability of an underlying model.

Prompt engineering, defined by Reynolds & McDonell (2021) as methods whereby humans iteratively modify prompts to elicit desired behaviors from LLMs, is also a common technique used to improve LLM outputs, and there is extensive work in this area. Schulhoff et al. (2024) provide a comprehensive taxonomy of prompt engineering techniques, many of which can be used in conjunction with the inference-time methods that we study. Similarly, answer engineering involves crafting or selecting algorithms to extract precise answers from LLM outputs, often requiring human involvement (Schulhoff et al., 2024, p. 17). Although some notions of prompt engineering are broad enough to include many of the inference-time methods we consider here, prompt engineering typically involves manual human effort (at either the task or individual query level) and post-hoc refinement in selecting seed prompts, which is outside the scope of the general-purpose and automated inference-time methods we consider here.

Mialon et al. (2023) surveys LLMs that are augmented with reasoning capabilities and external tools and considers how to evaluate these augmented LLMs. They do consider some inference-time methods like the ones we focus on in this paper, but the main emphasis is on the use of external tools like web search engines and symbolic reasoning modules. A recent literature review by Welleck et al. (2024) reviews different types of inference-time algorithms used with LLMs. Although the authors define differently than us, there is an overlap between some of the algorithms that they review and the methods that we evaluate. Our evaluation of these methods across various benchmarks and models, including state-of-the-art LLMs, provides novel insights and recommendations for future reliability studies.

The work closest to ours is a recent review by Sprague et al. (2024), focused on the popular "Chain-of-Thought" (CoT) method (Section 4.1). Their approach of evaluating methods with common goals across a standard set of datasets and benchmarks is similar to what we do in Section 4, and several of our conclusions are consistent with their results. In particular, the authors found that advantages of using the CoT method are largely limited to improving symbolic execution, although it does not perform as well as external symbolic solvers. This work was limited to studying just the CoT method, unlike our study which reviews a range of inference-time methods (including CoT) for LLM reliability.

3 Evaluations in the Literature

To assess the efficacy and generalizability of the inference-time reliability methods, we conducted a comprehensive automated analysis of the evaluation approaches employed in the relevant research literature (Section 3.1). For each paper, we catalog the evaluation benchmarks and models used by the authors to

test their proposed methods. Our analysis reveals a large variety of different benchmarks (Section 3.2) and models (Section 3.3) in use; no single model or benchmark is used in more than a third of the papers. This motivates our experiments in Section 4 to understand how robust evaluations are to choices of underlying models and benchmarks.

3.1 Literature selection

We performed our analysis using a collection of papers on inference-time methods for improving LLM reliability. To identify relevant papers, we implemented an automated system using the Semantic Scholar API.[1] This system systematically collected all papers that cited the "Chain-of-Thought" paper (Wei et al., 2022), which amounted to 6 318 papers in the Semantic Scholar index. We selected this paper as a starting point because it is widely considered a seminal early work in the field, so we expect it to be cited by most later research papers on inference-time techniques to improve LLM reliability.

To enable automated analysis and avoid licensing issues, we then filtered this set to only papers available on arXiv, which reduced the set to 4 895 papers. Of these, our automated script successfully retrieved 4 886 papers; the nine unsuccessful retrievals were all papers that had been withdrawn from arXiv. Our dataset spans from January 2022 (when the Wei et al. (2022) ("Chain-of-Thought") paper was posted on arXiv) through the end of 2024 (our collection search was run on 7 January 2025). For each identified paper, we retrieved the full text using the arXiv API, implementing appropriate rate limiting and error handling to ensure reliable data collection. The papers were stored with standardized naming conventions based on their titles, facilitating systematic organization and analysis.

To validate the automated analysis, we manually analyzed a set of 50 papers to extract model and benchmark data. For these 50 papers, we compared the results from the manual analysis to those from the automated GPT-4o analysis, finding a Jaccard similarity index of 0.887 for benchmark categorization and 0.874 for model categorization. A detailed examination of the discrepancies revealed three main types of cases. First, there were four instances where the automated analysis was more precise than human annotation, such as when the human analysts incorrectly classified certain models as foundational in papers like "ReAct" Yao et al. (2023c). We found five cases where terminology ambiguity in the original papers led to different but equally justifiable interpretations between human and automated analysis, as seen in papers discussing variations of benchmark names. There were also occasional oversights by the automated system, such as missing the text-curie-001 model in the "Boosted Prompt Ensembles" paper Pitis et al. (2023). The automated analysis showed consistency in identifying both common and rare model/benchmark combinations, suggesting reliable performance across the full spectrum of papers. These findings suggest that the high similarity index and the nature of the discrepancies support that the data resulting from our automated analysis is high enough quality to use for our purposes. Additional notes and inconsistencies between the categorizations can be found in the literature_analysis subdirectory in our repository.

Following this assurance mechanism, we used our automated system to systematically extract and categorize information from our corpus. This automated extraction was designed with specific criteria, though we encountered technical limitations when processing certain papers due to tokenization conflicts with special tokens like '<|endoftext|>' in the source text. Thus, we were unable to catalog requisite information from 12 papers, although we plan to address this issue in the future. For benchmarks, we included both standard evaluation datasets and custom evaluation sets, while excluding datasets used solely for training. For models, we captured all baseline comparisons, proposed models, and variants tested in ablation studies, but excluded models that were only referenced without experimental evaluation. We also normalize benchmark and model names following the categorizations, as the same model or benchmark could have been classified in different ways. For example, LLaMA-2-7B could have been referred to as LLaMA-2 7B or LLaMA 2 (7 billion), among other variants. We normalized these different ways of referring to the same model to a canonical name.

[1] This, and all of the data and code to reproduce our work, is available under a permissive open source license in a public GitHub repository which would normally be linked here. To support anonymous review, we do not include the URL in this submission, but provide an anonymized version at https://anonymous.4open.science/r/LLM-Evaluation-Framework-E0F0.

Table 1: Twenty most frequently used benchmarks and models in evaluations (Jan 2022–Dec 2024).

Benchmarks		**Models**	
Name	**#Papers**	**Name**	**#Papers**
GSM8K	425	GPT-4	1789
MATH	175	ChatGPT	1739
MMLU	171	GPT-3	709
SVAMP	139	PaLM-2	415
StrategyQA	126	GPT-4o	373
HotPotQA	120	LLaMA-2-7B	372
HumanEval	119	BERT	266
TruthfulQA	81	LlaMA-3-8B	264
CommonsenseQA	78	Mistral-7B	243
HellaSwag	71	LLaMA-1	230
TriviaQA	64	LLaMA-7B	201
AQUA	64	LLaMA-2-13B	198
MBPP	63	GPT-4V	195
MultiArith	61	LlaMA-3-70B	175
Winogrande	57	LLaMA-2-70B	172
BoolQ	56	T5	171
PIQA	53	RoBERTa	159
OpenBookQA	50	LLaMA-2	116
ASDiv	46	GPT-2	110
SQuAD	45	LLaMA-2-7B-Chat	108

3.2 Benchmarks

We catalog the evaluation methods used in the reviewed research works, focusing on the specific benchmarks, data types, and metrics employed. There were a total of 7 635 different benchmarks used across the 4 874 papers. The total number of benchmark mentions across all papers was 14 970, so on average each benchmark is used in fewer than two papers. The average number of benchmarks used per paper is 3.07.

Table 1 lists the most popular benchmarks used for evaluation across the set of paper. A few popular benchmarks are used in hundreds of papers, but no benchmark is common to more than 10% of the papers. Among the benchmarks, GSM8K (Grade School Math 8K) (Cobbe et al., 2021) is the most widely used, with 425 uses across the papers considered. GSM8K is a dataset of multiple-choice word problems, where the accuracy of the predicted answers is used as the evaluation metric. Several other popular benchmarks primarily test mathematical reasoning including SVAMP (139 uses), HumanEval (119), MBPP (63), MultiArith (61), and ASDiv (Academia Sinica Diverse MWP Dataset) Miao et al. (2020) with 46 mentions.

The most popular benchmarks that do not focus just on mathematical reasoning are MMLU (171 mentions) and StrategyQA (126). MMLU (Multi-task Language Understanding) Hendrycks et al. (2021) is a large-scale multi-subject benchmark that covers a wide range of academic subjects. StrategyQA (Geva et al., 2021) consists of multiple-choice questions that require reasoning over both a question and a given context to arrive at the correct answer. Other notable benchmarks include HotpotQA (120 mentions), TruthfulQA (81 mentions), and CommonsenseQA (78 mentions). HotPotQA is a dataset of multi-hop question-answering problems that require reasoning over multiple paragraphs. TruthfulQA is a benchmark for evaluating the truthfulness of generated answers, where models are assessed based on their ability to generate truthful and informative responses. SVAMP (Patel et al., 2021) is another multiple choice dataset of verb argument

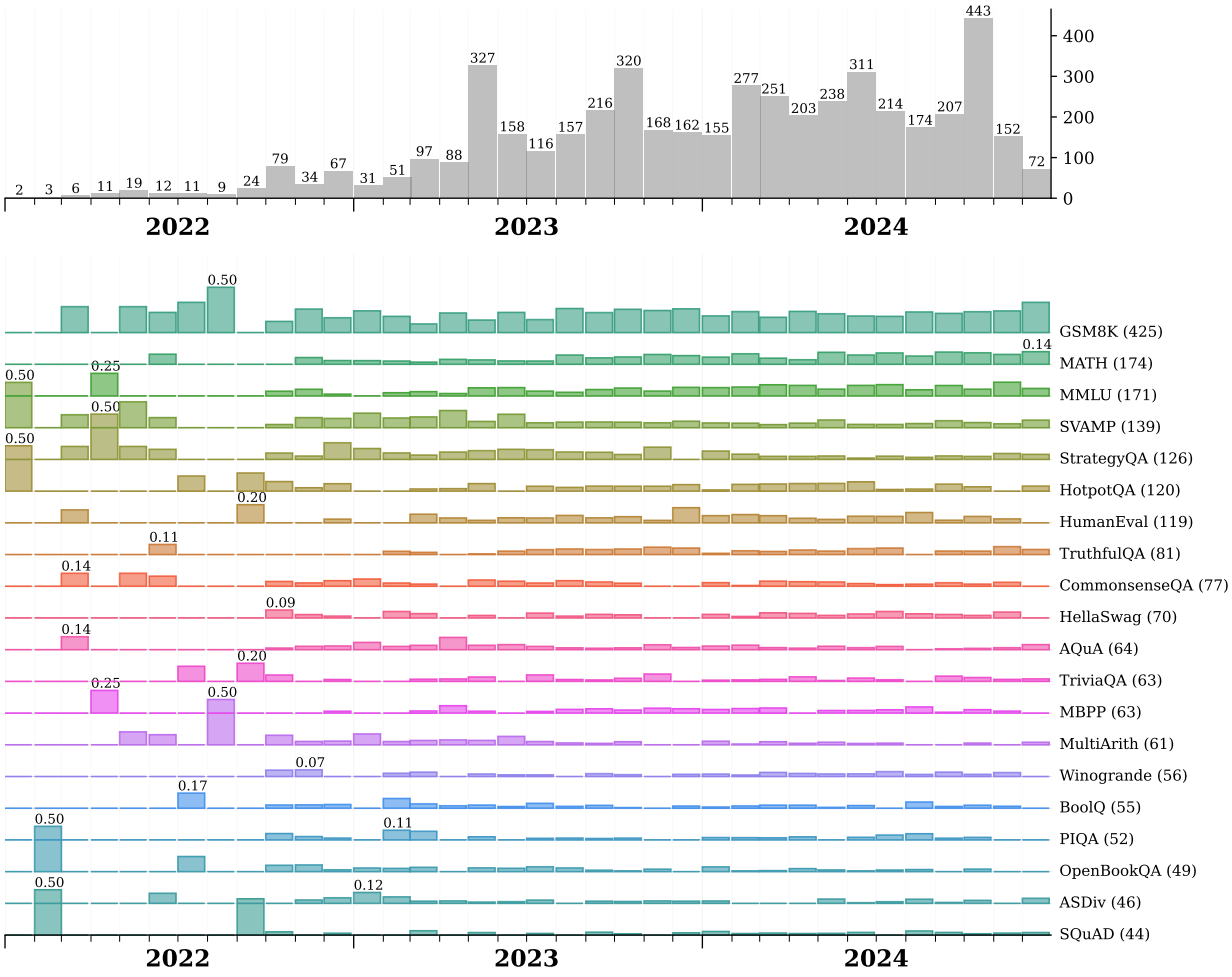

Figure 1: Proportions of papers posted in each month using each of the Top-20 LLM benchmarks (1 Jan 2022–31 Dec 2024). The top plot shows the total number of papers published per month. The bottom plot shows the proportion of the top-20 benchmarks. Each bar represents the proportion of papers in that month that used the given benchmark, calculated as (number of papers using benchmark) / (total benchmark mentions in papers that month). Total uses of each benchmark across the dataset are shown in parentheses.

structure alternations, and the accuracy of predicted verb forms serves as the evaluation method. Figure 1 shows these distribution changes, as different benchmarks become more popular in the research space.

Our analysis reveals a lack of consensus on what benchmarks should be used to evaluate methods for improving LLM reliability. For our experiments in Section 4, we select a representative set of benchmarks to evaluate methods across a diverse set of tasks and domains.

3.3 Models

Although there is somewhat more consensus on the models to use for evaluations than there is on the benchmarks, there is still a large variation in the models used with a total of 4 809 distinct models identified across 16 647 total model mentions in the analyzed papers. The large number of models may be partly due to variations in how the same model is named or very minor variations of a common model, although we attempted to canonicalize model names, at least for the commonly used models, in our analysis. The average number of models used per paper is 3.42.

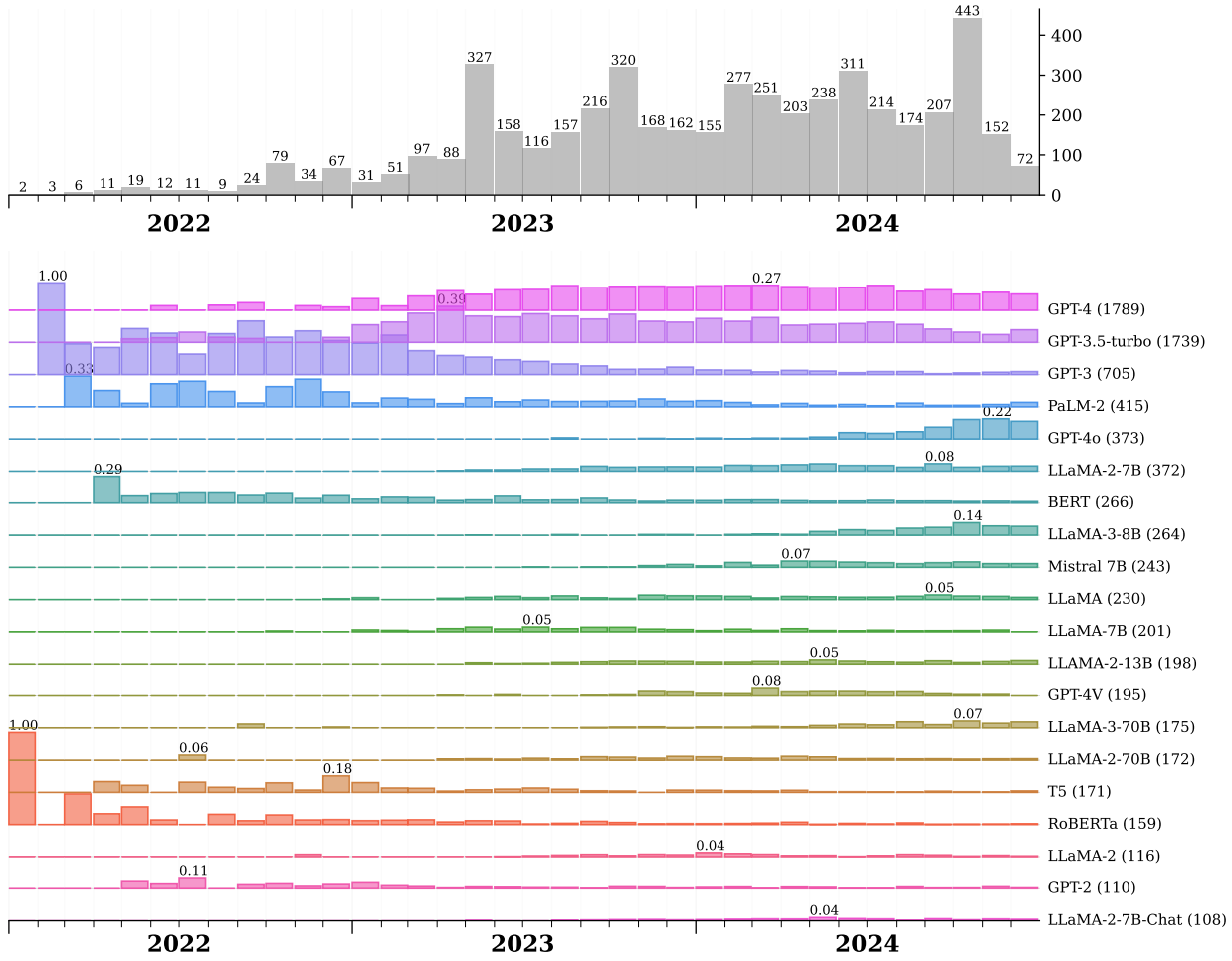

Figure 2: Monthly proportions of papers using each of the Top-20 LLMs (Jan 2022 - Dec 2024). The top plot shows the total number of papers published per month. Each bar in the bottom plot shows the fraction of papers published in that month that used the given model, computed as (papers mentioning model) / (total models that month). Total usage counts across the entire period are shown in parentheses.

The right side of Table 1 summarizes the models most commonly used across the considered papers. GPT-4 is the most frequently mentioned model, appearing in 1 789 of the 4 874 papers, followed by ChatGPT with 1 739 mentions and GPT-3 with 709 mentions. The diversity in model selection even among the top twenty spans from smaller models like T5 (171 mentions) to large open models like LLaMA-2-70B (172 mentions), reflecting the research community's interest in understanding performance across different model scales and architectures.

Open-weights models also feature prominently in the evaluations, with Mistral-7B (243 mentions), LLaMA-2-7B (372 mentions), and various LLaMA variants collectively used in a significant portion of the evaluations. Of important note is that there are 116 mentions of LLaMA-2 without specific distinction of the size of the model used. The frequent use of open-weights models enables reproducibility and transparency in a way that is not possible with models only available through an API. Open models also have important cost advantages over models that can only be access through pay-per-use APIs.

Model selection also changes over time, as new models become available. Figure 2 show how the distribution of models has changed over time. While newer models like GPT-4 dominate evaluations through the end of 2024, established models like BERT (266 mentions) and RoBERTa (159 mentions) continue to serve as

important baselines. Notably, the emergence of multi-modal models is evident with GPT-4V receiving 195 mentions, highlighting the growing interest in models that can handle both text and visual inputs.

4 Experiments

Our review of the research literature revealed variation in the benchmarks and underlying models used to evaluate inference-time methods for improving LLM reliability, motivating us to conduct experiments to assess the robustness and generalizability of proposed methods. Our primary goal is to determine whether results obtained from specific benchmarks and models in previous evaluations hold up when tested more comprehensively across a diverse range of state-of-the-art language models and previously unseen test data.

In our experiments, we evaluate the performance of the selected methods on different LLMs including more recent and powerful language models than may have been used in the original evaluations. We also test proposed methods on a broader set of benchmarks, including both common and rarely used benchmarks. Our experiments also assess the consistency of inference-time method performance across different model architectures and types, comparing both proprietary and open-weights models. With these experiments, we aim to provide a more nuanced understanding of the effectiveness of current techniques, identifying both strengths and limitations that may not have been apparent in original evaluations.

4.1 Methods Evaluated

Given our limited resources, we were not able to include all methods in our experiment. We selected methods to evaluate from papers in the literature review based on availability of standard implementations with a goal of having a representative set of methods to test. We conduct experiments using the following methods, ordered by the date that the methods were posted on arXiv: *Chain of Thought* (CoT) (Wei et al., 2022), *Self-Consistency* Wang et al. (2023), *ReAct* (Yao et al., 2023c), *Tree of Thoughts* (ToT) (Yao et al., 2024), *Graph of Thoughts* (GoT) (Besta et al., 2024), and *LLM Multi-Agent Debate* (Du et al., 2024). We provide descriptions of each of these methods and how we configured them for our experiments below.

The selected methods range from the basic prompt augmentation and output aggregation in CoT to complex multi-model interactions and sophisticated aggregation in LLM Multi-Agent Debate. Although CoT involves manually designing prompts to guide the model through step-by-step reasoning, we include the Chain-of-Thought method as a baseline to compare its effectiveness against the other automated inference-time techniques. We emphasize that our goal is not to identify the "best" inference-time method, hence this limited but representative selection of methods, but rather to understand what is necessary to perform a robust evaluation of an inference-time method in general.

For each method, we used the default settings provided by their respective repositories to ensure reproducibility and maintain consistency with the original implementations. It's important to note that this approach means the same language models may have different hyper-parameters when used across different methods. While this could potentially introduce confounding factors in performance trends, we prioritize fidelity to the original implementations as reported in their respective papers and repositories. This allows for a more direct comparison with previously published results. Full prompt structures for each method can be found in Appendix A.

In evaluating some of the methods, reproducing and running results proved challenging due to resource limitations, compatibility issues with current AI models, and the use of outdated or unsupported models in original evaluations. Some methods required excessive processing time or computing power, while others used packages incompatible with state-of-the-art models like those from Anthropic and newer OpenAI versions. Some methods include Pitis et al. (2023), Arora et al. (2022), and Si et al. (2023). We take these into consideration when comparing methods, models, and benchmarks and providing recommendations.

Chain-of-Thought (CoT) (Wei et al., 2022). CoT is a method designed to enhance the reasoning abilities of large language models. Each exemplar in few-shot prompting is augmented with a series of intermediate natural language reasoning steps—that leads to the final answer. The method samples from the output using greedy decoding. The original experiments used multiple arithmetic reasoning benchmarks (GSM8K,

SVAMP, ASDiv, MAWPS, and AQuA), and several models reflecting the state-of-the-art at the time: GPT-3 (350M–175B parameters), LaMDA (422M–137B), PaLM (8B–540B), UL2 (20B), and Codex.

For evaluating models on multiple-choice benchmarks, we used a CoT implementation based on the approach outlined in the referenced repository (Yao et al., 2023a). For more generative benchmarks, we used the method outlined in Besta et al. (2024). Each of the models were configured with a temperature of 0.7 and a maximum token limit of 1024 to allow for more elaborate reasoning chains. The prompt included multiple stages, with the model first analyzing the problem, laying out intermediate thought processes, and then computing or inferring the final result. We use "Let's think step by step" as a leading instruction guided the model in decomposing tasks into manageable chunks. More information on the prompt structure is found in Appendix A.

Self-Consistency (Wang et al., 2023). Self-consistency is an enhancement to CoT prompting that aims to improve language models' performance on complex reasoning tasks. While standard CoT uses greedy decoding to generate a single reasoning path, self-consistency samples multiple diverse reasoning paths from the model. It then extracts the final answer from each path and determines the most consistent answer through majority voting. The authors evaluated self-consistency on a variety of arithmetic and commonsense reasoning benchmarks (GSM8K, SVAMP, AQuA, StrategyQA, and ARC-challenge) with four underlying LLMs—PaLM (540B), GPT-3 (175B), LaMDA (137B), and UL2 (20B).

The original authors did not provide a public implementation, so we produced our own implementation following the description in the paper to include this in our experiments. The method was applied by generating 3, 5, and 10 diverse reasoning paths for each task. These reasoning paths were produced by prompting the model multiple times, with a focus on encouraging variation in the intermediate steps taken toward the solution. The prompt augmentation phase utilizes CoT prompt methods. Each path was evaluated independently, and the final answer was determined by aggregating the results to select the most consistent outcome across all generated paths. The process allowed the model to explore a range of potential solutions, increasing the likelihood of arriving at a good answer through collective reasoning.

ReAct (Yao et al., 2023c). The ReAct method combines reasoning and acting by augmenting a language model's action space to include both external actions and language-based thoughts. For each seed prompt, the model generates a series of thoughts that update the context without affecting the environment. The model's responses are based on few-shot in-context examples, each containing a human-generated trajectory of actions, thoughts, and observations for a specific task instance. The original study tested the method using PaLM-540B and GPT-3 on the HotpotQA, Fever, Alfworld, and WebShop benchmarks. Subsequent ablation studies also included testing GPT-3.5-turbo on the GSM8k dataset (Face, 2023).

For our experiments, ReAct was implemented using a custom LangChain agent based on the original reference repository Chase (2022). Format errors were corrected during execution to ensure the output followed the required structure. The model was configured with a temperature of 0.5, a maximum token limit of 512, and up to two retries in case of errors. It was instructed to stop generating text at specific markers such as "\nHuman:" or "Final Answer:".

Tree of Thoughts (ToT) (Yao et al., 2024). While CoT generates a single sequence of thoughts and lacks exploration of alternative reasoning paths and Self-Consistency improves upon this by sampling multiple independent chains, Tree of Thoughts frames problem-solving as a search over a tree of thoughts, allowing for both local and global exploration of the problem space. This paradigm incorporates planning, look-ahead, and backtracking, enabling the evaluation and pruning of intermediate states. The ToT framework consists of four key components: thought decomposition, thought generation, state evaluation, and search algorithms. Thought decomposition breaks down the intermediate process into discrete steps, adapting to different problem types. The thought generator produces k candidates for the next thought given a tree state. It employs two strategies: sampling, which uses a CoT prompt to generate thoughts, and proposing, which uses a "propose prompt" to sequentially generate thoughts. The state evaluator then assesses the progress of different states towards solving the problem, serving as a heuristic for the search algorithm. It can either value each state independently or vote across states, depending on the problem's characteristics.

The search algorithm navigates the tree structure to find the solution, using either a breadth-first search (BFS) or depth-first search (DFS) strategy. BFS maintains a set of the most promising states per step and is used for problems with limited tree depth, while DFS explores the most promising state first until a final output is reached or deemed impossible to solve.

For benchmarks based on multiple-choice question answering solutions, we used a ToT implementation based on the approach outlined in the referenced repository (Yao et al., 2023b). For more generative benchmarks, we used the method outlined in Besta et al. (2024). To run the experiments, we used the standard prompt format to generate sequences of thoughts through sequential proposals, where each intermediate thought was sampled based on previous reasoning steps, creating a more structured exploration. For this, the model used a propose strategy for thought generation and a value approach for evaluating the intermediate states. During evaluation, each path was sampled multiple times, with the system generating thoughts multiple times to improve diversity. The default parameters prompted the model to generate sequences 10 times and evaluate each generated state five times, while the BFS algorithm was configured to retain the top three states at each step to explore further.

Graph of Thoughts (GoT) (Besta et al., 2024). GoT introduces a graph-based structure to model reasoning paths, allowing for more flexible exploration of possible solutions by dynamically connecting paths. GoT models the LLM's reasoning process as an arbitrary graph, where thoughts are represented as vertices and dependencies between thoughts as edges. This graph-based approach allows for more complex transformations than previous methods like Chain-of-Thought or Tree of Thoughts. The framework is modeled as a tuple containing the reasoning process graph, "thought" transformations, an evaluator function, and a ranking function. GoT enables several graph-enabled transformations, including thought aggregation, where multiple thoughts can be combined into new ones, thought refinement through iterative improvement, and parallel thought generation. These transformations are managed through a scoring and ranking system that evaluates thoughts and selects the most promising forward paths. For our experiments, we used the open-source repository (Blach et al., 2023).

LLM Multi-Agent Debate (Du et al., 2024). In an LLM Multi-Agent Debate, multiple agents are prompted to evaluate a problem from different perspectives, and through a series of interactions, they converge on a solution through collective reasoning. For our experiments we used the repositories (Du et al., 2023) for proprietary models and (Gauss5930, 2023) for open-source models. Furthermore, we utilized three agents, all based on the same model, and conducted two rounds of debate per task. During each round, the agents would present their arguments, and subsequent rounds allowed them to refine or rebut each other's points. The debate was structured to ensure that the agents were working both collaboratively and competitively to arrive at the most accurate solution. The response was determined based on the consensus or the strongest argument presented by the agents at the end of the debate. While the same model was used for all agents in this study, future research could explore the potential benefits of using a diverse range of models as agents, allowing for even greater variation in reasoning and argumentation.

4.2 Setup

For our experiments, we selected a set of representative methods to evaluate and a common set of models and benchmarks to use in the evaluation, informed by our literature analysis from Section 3.

Models. As reported in Table 1, many different models have been used in evaluations, and most works only evaluate on three different models, averaged across papers. For our experiments, we selected a mix of models to enable us to measure both the performance of each method on state-of-the-art models and how that performance varies with smaller models. For the state-of-the-art models, we selected GPT-4o and Claude 3.5-Sonnet. As of May 2024 (when we started our experiments) the two models performed well in the LMSYS Chatbot Arena Leaderboard Chiang et al. (2024). Although more recent models have now surpassed these models in most rankings, both GPT-4o and Claude 3.5-Sonnet are still reasonably highly-ranked in LMSYS and other rankings such as the SCALE AI rankings (Scale AI). We include GPT-3.5-turbo to compare our results with most of the reported results across methods. We include two open-weights models, Llama-3.1-8B-Instruct and Mixtral 8x22B, both for comparison and to enable reproducibility.

Benchmarks. For our analysis, we choose five of the most commonly used benchmarks in the literature—GSM8K, MMLU, AQUA, SVAMP, TruthfulQA. These benchmarks are described in Section 3.2, and include three commonly used mathematical reasoning benchmarks (GSM8K, AQUA, and SVAMP) and two broad language understanding benchmarks (MMLU and TruthfulQA).

We also include GSM-Symbolic (Mirzadeh et al., 2024), a relatively new benchmark that was not used by any of the papers in our analysis. GSM-Symbolic was constructed by converting GSM8K questions into symbolic templates that allow for controlled variation of parameters like names, numbers, and problem complexity. Using 100 templates from GSM8K, it generates 50 samples per template, resulting in 5 000 total examples for each benchmark variant. The dataset includes different difficulty levels, from simpler versions with clauses removed (GSM-Symbolic-M1) to more complex versions with additional clauses (GSM-Symbolic-P1, P2), and a special variant (GSM-NoOp) that tests models' ability to identify relevant information. The dataset enables evaluation of models' robustness to parameter changes, handling of increasing complexity, and true understanding of mathematical concepts versus pattern matching. Our evaluation follows the methodology established in Mirzadeh et al. (2024).

We include two additional benchmarks that were not commonly used, but were chosen to further evaluate generalization. Sorting 032 (Besta et al., 2024) evaluates a model's ability to sort a sequence of numbers in ascending order. It consists of array sequences of 32 numbers in the range 0–9, and measures the model's performance as the percentage of correctly sorted sequences. Document Merging (Besta et al., 2024) was introduced in the Graph of Thoughts paper. It assesses a model's ability to merge multiple documents into a single coherent document. To evaluate an output, we use the same evaluation criteria as in Besta et al. (2024). We use the underlying LLM in each experiment to assess two key metrics, each queried three times to obtain an average. The first metric measures conciseness, with 0 suggesting at least 50% redundant information and 10 indicating no redundancy. The second metric gauges information preservation, where 0 indicated total information loss and 10 signifies complete retention. The final score is the harmonic mean of these two values and the average rating across multiple document sets is used as the evaluation metric.

5 Results

A key goal of our experiments is to understand how well results from reported benchmarks and models predict the performance of a method on other benchmarks and with better models. Section 5.1 reports on the overall performance of the inference-time methods. Section 5.2 considers how well results in previous evaluations are reproduced in our experiments, and Section 5.3 evaluates the how the execution costs vary.

5.1 Performance of Inference-Time Methods

Figure 3 compares the results from different methods using different models on the selected benchmarks and Table 2 summarizes the average performance improvement for each method across the benchmarks across the five evaluation models.

Due to the high cost of running some of the methods (which we discuss in Section 5.3), for each of the methods we randomly sample 150 data points from each of benchmarks. We then normalize the data by employing a min-max normalization to standardize data across different methods, scaling all values to a 0–1 range to achieve consistent comparisons across benchmarks. The *baseline* refers to the accuracy achieved by each model when no inference-time method is employed. Positive deviations, colored green in the heat map, indicate improvements over the unaided model baseline, whereas negative deviations, shown in red, indicate a decline in performance.

We observe that significant variations in performance across different tasks, models, methods, and benchmarks, which remains a problem in evaluating large language models and respective ensemble methods. Note in particular that each method has a negative impact on at least one of the benchmarks. Two of the benchmarks (GSM8K, Document Merging) have positive improvements on average for all of them methods, but for every other benchmarks at least one of the methods results in worse performance.

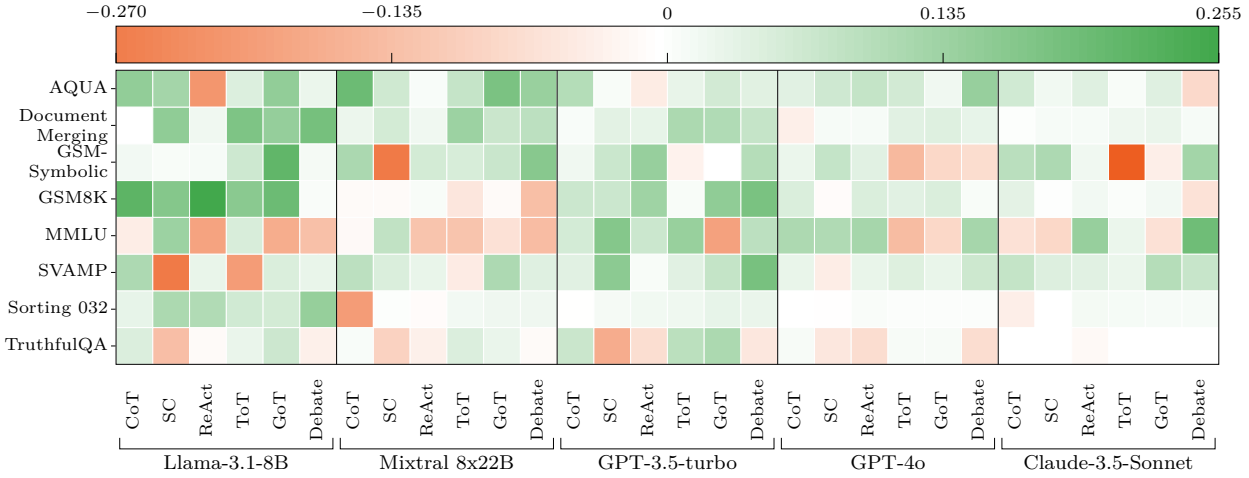

Figure 3: Heatmap displaying the deviation from baseline accuracy for various inference-time methods applied across different models and benchmarks. Positive deviations (in green) indicate improvements over the unaided model (baseline), while negative deviations (in red) indicate performance decline.

Table 2: Average performance change across all models and methods. Positive values indicate improved performance relative to a baseline. Results show varying effectiveness of each technique, highlighting the challenge of developing methods that work across different problem types. Cost is measured in average number of API calls per problem for GPT-3.5-turbo.

	CoT	SC	ReAct	ToT	GoT	Debate
Relative Cost	3.5	10.0	5.2	53.8	37.2	6.0
GSM8K	0.073 ± 0.078	0.044 ± 0.067	0.092 ± 0.092	0.033 ± 0.069	0.081 ± 0.080	0.002 ± 0.103
GSM-Symbolic	0.051 ± 0.044	-0.002 ± 0.146	0.052 ± 0.054	-0.075 ± 1.000	0.034 ± 1.000	0.063 ± 0.092
MMLU	-0.024 ± 0.120	0.099 ± 0.054	-0.031 ± 0.114	-0.002 ± 0.099	-0.147 ± 0.068	-0.008 ± 0.107
AQUA	0.110 ± 0.057	0.057 ± 0.040	-0.024 ± 0.102	0.045 ± 0.024	0.090 ± 0.062	0.054 ± 0.081
SVAMP	0.070 ± 0.030	-0.011 ± 0.143	0.028 ± 0.010	-0.026 ± 0.092	0.074 ± 0.030	0.080 ± 0.054
Sorting	-0.041 ± 0.082	0.026 ± 0.044	0.028 ± 0.040	0.024 ± 0.020	0.026 ± 0.019	0.042 ± 0.051
Document Merg	0.001 ± 0.019	0.054 ± 0.051	0.019 ± 0.008	0.096 ± 0.057	0.079 ± 0.042	0.079 ± 0.060
TruthfulQA	0.026 ± 0.031	-0.102 ± 0.042	-0.038 ± 0.026	0.036 ± 0.033	0.042 ± 0.044	-0.050 ± 0.029

For the Llama-3.1-8B-Instruct model, we observe significant improvements (visible in Figure 3) across most benchmarks when using ensemble methods. The Graph of Thoughts method shows particularly strong performance on the Document Merging task. Chain of Thought and Tree of Thoughts also demonstrate consistent improvements across various benchmarks for this model. Mixtral 8x22B shows a more varied performance profile. While it benefits from ensemble methods in tasks like AQUA and GSM8k, it shows some negative deviations in benchmarks such as MMLU and TruthfulQA. The Multi-Agent Debate approach appears particularly effective for this model on the AQUA benchmark. GPT-3.5-turbo demonstrates more modest improvements from ensemble methods compared to the previous two models. However, it still shows positive deviations in several benchmarks, particularly when using the Self-Consistency method on GSM8K and the ReAct method on AQUA.

For the more advanced models, GPT-4o and Claude-3.5-Sonnet, the impact of the tested methods is less pronounced, as visible in the lighter color shades in the heat map. This suggests that these models already perform well on many tasks without additional ensemble techniques and obtaining further improvements through inference-time methods is more challenging. We do observe some improvements, particularly in the Document Merging task for both models when using the Graph of Thoughts method (which introduced this benchmark). The SVAMP benchmark shows interesting variations across models and methods. While some

Table 3: Comparison of reported and reproduced results across different methods and benchmarks. The Chain of Thought reported results are from the Chain-of-Thought-Hub repository Fu et al. (2023). The Tree of Thoughts Yao et al. (2024) GSM8K reported results are from the original GPT-4 experiments. The Self-Consistency reported results are from experiments with GPT-3 Wang et al. (2023). All reproduced results are from our own experiments using GPT-3.5-turbo.

	GPT-3.5-turbo			
	Unaided		Using Method	
Methods and Benchmarks	Reported	Reproduced	Reported	Reproduced
Chain of Thought				
GSM8K	–	0.63	0.75 (+0.12)	0.70 (+0.07)
MMLU	–	0.63	0.67 (+0.04)	0.69 (+0.06)
AQuA	–	0.64	–	0.74 (+0.10)
SVAMP	–	0.72	–	0.76 (+0.04)
Sorting 032	0.86	0.96 (+0.10)	0.79 (-0.17)	0.96 (+0.00)
Document Merging	0.64	0.70 (+0.06)	0.66 (-0.04)	0.71 (+0.01)
Self-Consistency				
GSM8K	–	0.63	–	0.70 (+0.07)
MMLU	–	0.63	–	0.79 (+0.16)
AQuA	–	0.63	–	
SVAMP	–	0.72	–	0.87 (+0.15)
ReAct				
GSM8K	–	0.63	–	0.76 (+0.13)
Tree of Thoughts				
GSM8K	–	0.63	–	0.64 (+0.01)
Sorting 032	0.86	0.96 (+0.10)	0.95 (-0.01)	0.98 (+0.02)
Document Merging	0.64	0.70 (+0.06)	0.78 (+0.08)	0.81 (+0.11)
Multi-Agent Debate				
GSM8K	0.77	0.63 (-0.14)	0.85 (+0.22)	0.81 (+0.18)
MMLU	0.64	0.63 (-0.01)	0.71 (+0.08)	0.72 (+0.09)

ensemble methods yield improvements for Llama-3.1-8B and Mixtral 8x22B, the more advanced models show minimal changes or even slight negative deviations when applying these methods to SVAMP.

Overall, our experiments reveal that the effectiveness of tested methods varies not only across different benchmarks but also across different model capacities. Less powerful models like Llama-3.1-8B-Instruct tend to benefit more consistently from the inference-time methods, while more advanced models like GPT-4o and Claude-3.5-Sonnet show limited improvements.

5.2 Reproducibility

In addition to understanding how well methods generalize to different models and benchmarks, we also wanted to study now reliably results reports in papers could be reproduced.

We compare GPT-3.5-turbo's performance both without any inference-time method (*Unaided*) to the tested method (*Using Method*) in Table 3, showing both the results reported in the original papers and our reproductions and generalizations. The table reveals discrepancies between the reported and reproduced results for GPT-3.5-turbo, underscoring reproducibility challenges in evaluating large language models.

Detailed view of Document Merging benchmark. Figure 4 shows a more detailed view of the impact of the methods on the Document Merging benchmark. This task serves as a useful benchmark to compare

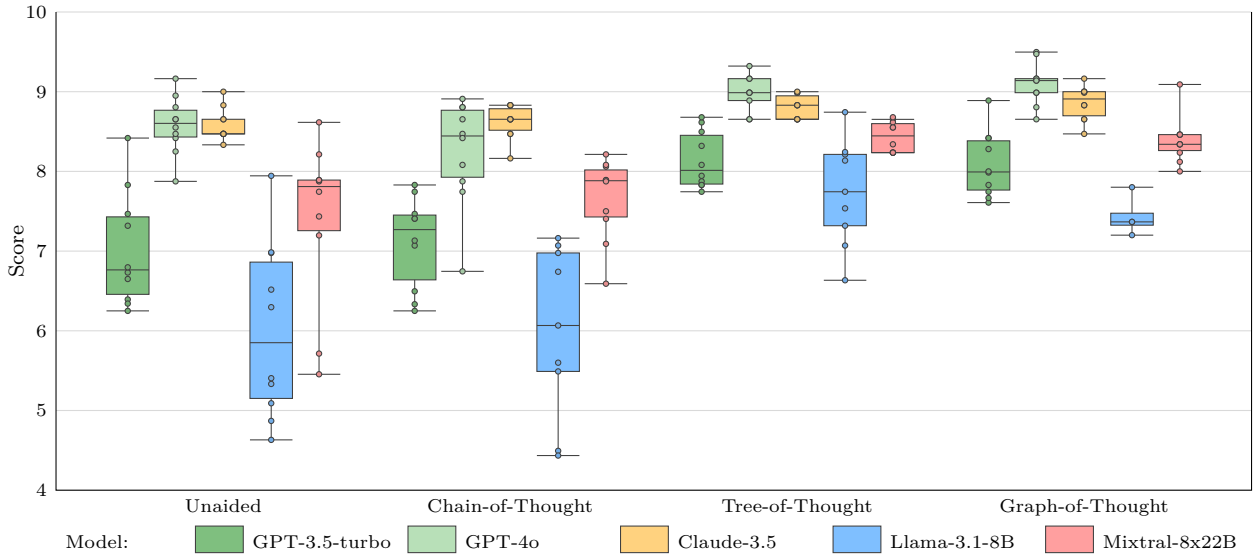

Figure 4: Comparison of document merging task across different models and approaches Besta et al. (2024). The plot illustrates the performance of different approaches (*Unaided*, *Chain-of-Thought*, *Tree-of-Thoughts*, and *Graph-of-Thoughts*), using box plots to summarize score distributions and swarm plots to show scores of individual iterations.

the performance of different language models, as it represents a common use case across various LLM applications. While this benchmark was not originally used in the papers introducing Chain-of-Thought and Tree-of-Thoughts, the authors of the Graph-of-Thoughts paper instantiated their own replications of these methods for comparison purposes. This explains the discrepancy between the benchmark data presented in Table 3. The Document Merging benchmark thus offers a unique perspective on how these different approaches perform when implemented under consistent conditions by the same research team.

We first replicate the results reported by Besta et al. (2024) for this task using GPT-3.5-turbo. We then extend our evaluation to use our selected models. Our analysis reveals interesting patterns in model performance. While state-of-the-art models like GPT-4o and Claude 3.5-Sonnet show negligible improvements over the baseline, the magnitude of these gains varies. These results not only demonstrate the varying capabilities of different models in handling complex tasks like document merging but also highlight the importance of model selection for specific applications. The findings suggest that while more powerful models generally perform better, the degree of improvement can vary based on the task complexity and the specific strengths of each model.

5.3 Cost

Although our analysis focused on reliability, the cost of executing an inference-time method can be substantial. We measure costs by the number of API calls used to solve a given problem in a benchmark and include these results in Table 2.

It is clear that the cost of all of the methods are high, requiring an average of from 3.5 (CoT) to over 50 (ToT) instantiations of the base LLM for each seed prompt. We use API calls to measure cost due to the changing nature of token costs for black box models. We report the number of calls when the base model is GPT-3.5-turbo, and the multiple may vary slightly based on the base model used.

Recent work has explored optimizing these costs: Snell et al. (2024) studies how to optimally scale test-time computation for LLMs on math reasoning tasks, while Chen et al. (2024) examines the cost-benefit tradeoffs of multiple API calls. These findings align with our observations about the importance of balancing performance gains against computational costs.

As mentioned earlier, these high costs limited our experiments, but they are a more important factor in any considered deployment. One measure of the practical value of these methods would be if the total cost of obtaining the same performance is lower using the inference-time method with a less expensive model than the cost of obtaining similar performance from a state-of-the-art high cost model.

6 Discussion

Improving LLM reliability is a critical goal, and there is an active research community exploring myriad approaches, including much focus on the inference-time methods we study here. To make progress in this area, it is critical that evaluations are done in a way that can robustly determine if a proposed method is a meaningful improvement on other methods. Our analysis of the evaluation approaches used in the considered literature shows a large range of different evaluation methods, with hundreds of different benchmarks used and more than half of the papers conducting evaluations with just one benchmark and no benchmark used by more than a quarter of the evaluations (although these are primarily the result of surveys in the citation chain). There is somewhat more consensus on the models to use, and the available state-of-the-art proprietary and open weights models will continue to change over time.

As demonstrated in our experiments, and captured in Figure 3, the impact of an inference-time method on reliability varies substantially across both underlying models and selected benchmarks. Methods that produce large improvements with weaker models often produce little improvement (or even make things worse) for stronger models. As further emphasized by Table 2, methods that produce significant improvements for certain mathematical reasoning benchmarks, may not result in improvements for other benchmarks.

Our results highlight the importance of evaluating methods with a range of underlying models, but especially with models representative of the state-of-the-art, at least if the goal is to develop methods that are useful in making overall improvements in reliability. The choice of benchmarks is also important. Unfortunately, the resources required to run extensive tests on all available benchmarks are not available to researchers outside of the largest industry groups, so it is important to select a suite of benchmarks that are sufficient to understand the impact of a method across a range of settings that cover the intended use cases. None of the current benchmarks, at least of the ones considered in our evaluation, are representative enough to be used as a single benchmark that would allow researchers to draw general conclusions.

Much work is needed in understanding the effectiveness of different benchmarks and underlying models to predict the performance of a method in other settings. Developing new benchmarks is part of this, but it is important that the predictive value of any new benchmark is evaluated by also using more established benchmarks. Research will accelerate in this area as the community makes progress to a set of standardized benchmarks and better understanding of how performance impacts translate across models and tasks.

Availability. Code for reproducing our experiments is available anonymously for reviewers (and will be made public) at *https://anonymous.4open.science/r/LLM-Evaluation-Framework-E0F0*.

References

Simran Arora, Avanika Narayan, Mayee F. Chen, Laurel Orr, Neel Guha, Kush Bhatia, Ines Chami, Frederic Sala, and Christopher Ré. Ask me anything: A simple strategy for prompting language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. *AAAI Conference on Artificial Intelligence*, 2024.

Nils Blach, Robert Gerstenberger, and Ales Kubicek. Graph of thoughts. https://github.com/spcl/graph-of-thoughts, 2023.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. A survey on evaluation of large language models, 2023. URL https://arxiv.org/abs/2307.03109.

Harrison Chase. Langchain, October 2022. URL https://github.com/langchain-ai/langchain. Version 1.2.0.

Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. Are more llm calls all you need? towards scaling laws of compound inference systems, 2024. URL https://arxiv.org/abs/2403.02419.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating llms by human preference. In *International Conference on Machine Learning (ICML)*, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. LLM multi-agent debate. https://github.com/composable-models/llm_multiagent_debate, 2023.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *International Conference on Machine Learning (ICML)*, 2024.

Hugging Face. Open-source llms as agents: How they work and why they matter. https://huggingface.co/blog/open-source-llms-as-agents, 2023.

Yao Fu, Litu Ou, Mingyu Chen, Yuhao Wan, Hao Peng, and Tushar Khot. Chain-of-thought hub: A continuous effort to measure large language models' reasoning performance. *ICML Workshop Deployable-GenerativeAI homepage*, 2023.

Gauss5930. Llm-agora, September 2023. URL https://github.com/gauss5930/LLM-Agora. Version 1.0.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies. *Transactions of the Association for Computational Linguistics (TACL)*, 2021.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *International Conference on Learning Representations (ICLR)*, 2021.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2023.

Grégoire Mialon, Roberto Dessi, Maria Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, Roberta Raileanu, Baptiste Roziere, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. Augmented language models: A survey. *Transactions on Machine Learning Research*, 2023.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models, 2024. URL https://arxiv.org/abs/2410.05229.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021.

Silviu Pitis, Michael R. Zhang, Andrew Wang, and Jimmy Ba. Boosted prompt ensembles for large language models. *arXiv preprint arXiv:2304.05970*, 2023.

Laria Reynolds and Kyle McDonell. Prompt programming for large language models: Beyond the few-shot paradigm, 2021. URL https://arxiv.org/abs/2102.07350.

Scale AI. Leaderboards. https://scale.com/leaderboard.

Sander Schulhoff, Michael Ilie, Nishant Balepur, Konstantine Kahadze, Amanda Liu, Chenglei Si, Yinheng Li, Aayush Gupta, HyoJung Han, Sevien Schulhoff, Pranav Sandeep Dulepet, Saurav Vidyadhara, Dayeon Ki, Sweta Agrawal, Chau Pham, Gerson Kroiz, Feileen Li, Hudson Tao, Ashay Srivastava, Hevander Da Costa, Saloni Gupta, Megan L. Rogers, Inna Goncearenco, Giuseppe Sarli, Igor Galynker, Denis Peskoff, Marine Carpuat, Jules White, Shyamal Anadkat, Alexander Hoyle, and Philip Resnik. The prompt report: A systematic survey of prompting techniques. *arXiv preprint arXiv:2406.06608*, 2024.

Xinyue Shen, Zeyuan Chen, Michael Backes, and Yang Zhang. In ChatGPT we trust? measuring and characterizing the reliability of ChatGPT. *arXiv preprint arXiv:2304.08979*, 2023.

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Boyd-Graber, and Lijuan Wang. Prompting gpt-3 to be reliable. In *International Conference on Learning Representations (ICLR)*, 2023.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. *arXiv preprint arXiv:2409.12183*, 2024.

Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, Sang T. Truong, Simran Arora, Mantas Mazeika, Dan Hendrycks, Zinan Lin, Yu Cheng, Sanmi Koyejo, Dawn Song, and Bo Li. Decodingtrust: A comprehensive assessment of trustworthiness in gpt models, 2024. URL https://arxiv.org/abs/2306.11698.

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. In *International Conference on Learning Representations (ICLR)*, 2023.

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 *Advances in Neural Information Processing Systems*, 2022.

Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. From decoding to meta-generation: Inference-time algorithms for large language models, 2024. URL https://arxiv.org/abs/2406.16838.

Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Bing Yin, and Xia Hu. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *ACM Transactions on Knowledge Discovery from Data*, 2024.

Francis Yao, Litu Ou, Mingyu Chen, Yuhao Wan, Hao Peng, Tushar Khot, and Wenhu Chen. Chain-of-thought hub. https://github.com/FranxYao/chain-of-thought-hub, 2023a. Accessed: 2024-09-14.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts. https://github.com/princeton-nlp/tree-of-thought-llm, 2023b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023c.

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. In *Advances in Neural Information Processing Systems*, 2024.

A Experimental Prompts

ReAct Prompt Example

You are an AI assistant designed to answer multiple-choice questions. Analyze the problem and select the best answer from the provided options (A, B, C, D).
Question:
Options: A: B: C: D:
Previous steps:
Your task is to provide a final answer.
Use the following format: Thought: [Your reasoning] Final Answer: [Letter choice (A, B, C, or D)] [Brief explanation]
If you cannot determine the answer with certainty, make your best guess based on the information available.
Your response:

Chain-of-Thought Prompt Example

Question: Angelo and Melanie want to plan how many hours over the next week they should study together for their test next week. They have 2 chapters of their textbook to study and 4 worksheets to memorize. They figure out that they should dedicate 3 hours to each chapter of their textbook and 1.5 hours for each worksheet. If they plan to study no more than 4 hours each day, how many days should they plan to study total over the next week if they take a 10-minute break every hour, include 3 10-minute snack breaks each day, and 30 minutes for lunch each day? Let's think step by step Angelo and Melanie think they should dedicate 3 hours to each of the 2 chapters, 3 hours x 2 chapters = 6 hours total. For the worksheets they plan to dedicate 1.5 hours for each worksheet, 1.5 hours x 4 worksheets = 6 hours total. Angelo and Melanie need to start with planning 12 hours to study, at 4 hours a day, 12 / 4 = 3 days. However, they need to include time for breaks and lunch. Every hour they want to include a 10-minute break, so 12 total hours x 10 minutes = 120 extra minutes for breaks. They also want to include 3 10-minute snack breaks, 3 x 10 minutes = 30 minutes. And they want to include 30 minutes for lunch each day, so 120 minutes for breaks + 30 minutes for snack breaks + 30 minutes for lunch = 180 minutes, or 180 / 60 minutes per hour = 3 extra hours. So Angelo and Melanie want to plan 12 hours to study + 3 hours of breaks = 15 hours total. They want to study no more than 4 hours each day, 15 hours / 4 hours each day = 3.75 They will need to plan to study 4 days to allow for all the time they need. The answer is 4
Question: Mark's basketball team scores 25 2 pointers, 8 3 pointers and 10 free throws. Their opponents score double the 2 pointers but half the 3 pointers and free throws. What's the total number of points scored by both teams added together? Let's think step by step Mark's team scores 25 2 pointers, meaning they scored 25*2= 50 points in 2 pointers. His team also scores 6 3 pointers, meaning they scored 8*3= 24 points in 3 pointers They scored 10 free throws, and free throws count as one point so they scored 10*1=10 points in free throws. All together his team scored 50+24+10= 84 points Mark's opponents scored double his team's number of 2 pointers, meaning they scored 50*2=100 points in 2 pointers. His opponents scored half his team's number of 3 pointers, meaning they scored 24/2= 12 points in 3 pointers. They also scored half Mark's team's points in free throws, meaning they scored 10/2=5 points in free throws. All together Mark's opponents scored 100+12+5=117 points The total score for the game is both team's scores added together, so it is 84+117=201 points The answer is 201 ... Therefore, 1000 - 480 = 520 do not like to play basketball. The percentage of the school that do not like to play basketball is 520/1000 * 100 = 52 The answer is 52

Tree-of-Thought Prompt Example

Here's a math word problem: [input]
Current solution steps:[partial solution]
What should be the next step in solving this problem?
Here's a math word problem: [input]
Partial solution: [partial solution]
How likely is this partial solution to lead to the correct answer? (impossible/unlikely/likely/very likely/certain)
Here's a math word problem: [input]
Proposed final answer: [answer]
How likely is this answer to be correct? (impossible/unlikely/likely/very likely/certain)

Graph-of-Thought Prompt Example

<Instruction> Merge the following 2 sorted lists of length length1 each, into one sorted list of length length2 using a merge sort style approach. Only output the final merged list without any additional text or thoughts!:</Instruction>
<Approach> To merge the two lists in a merge-sort style approach, follow these steps: 1. Compare the first element of both lists. 2. Append the smaller element to the merged list and move to the next element in the list from which the smaller element came. 3. Repeat steps 1 and 2 until one of the lists is empty. 4. Append the remaining elements of the non-empty list to the merged list. </Approach>
Merge the following two lists into one sorted list: 1: input1 2: input2
Merged list:

Self-Consistency Prompt Example

You will be provided with the answer to a question. The question and options are delimited by triple backticks, and the answer is delimited by triple hashtags. Extract the final answer from the provided solution. Return only the letter corresponding to the chosen option (A, B, C, D, or E), prefixed by 'Final answer:'

LLM Multi-Agent Debate Prompt Example

Using the solutions from other agents as additional information, can you provide your answer to the math problem? The original math problem is []. Your final answer should be a single numerical number, in the form [answer], at the end of your response.

B Experimental Results

Table 4: Comparison of Benchmarks across Methods and Models.

Method	Benchmarks	Results				
		GPT-3.5-turbo	**GPT-4o**	**Claude-3.5-Sonnet**	**Mixtral 8x22B**	**Llama-3.1-8B**
Unaided	GSM8K	0.767	0.910	0.963	0.707	0.593
	GSM-Symbolic	0.733	0.893	0.847	0.740	**0.580**
	MMLU	0.630	0.760	0.840	0.760	0.667
	AQUA	0.639	0.820	0.857	0.662	0.613
	SVAMP	0.720	0.890	0.840	0.790	0.770
	Sorting 032	0.961	0.992	0.985	0.968	0.818
	Document Merging	0.702	0.858	0.858	0.740	0.600
	TruthfulQA	0.848	0.949	0.993	0.931	0.911
Chain of Thought	GSM8K	0.700	0.960	1.000	0.900	0.745
	GSM-Symbolic	0.753	0.920	0.940	0.853	0.764
	MMLU	0.689	0.871	0.600	0.748	**0.631**
	AQUA	0.740	0.860	0.920	0.860	0.760
	SVAMP	0.760	0.920	0.920	0.880	0.880
	Sorting 032	0.960	0.992	0.951	0.768	0.850
	Document Merging	0.711	0.825	0.862	0.766	0.600
	TruthfulQA	0.920	0.960	0.980	0.940	0.960
Self-Consistency	GSM8K	0.700	0.902	0.966	0.900	0.767
	GSM-Symbolic	0.804	0.972	0.957	**0.463**	**0.520**
	MMLU	0.795	0.867	0.845	0.844	0.800
	AQUA	0.649	0.886	0.876	0.727	0.736
	SVAMP	0.875	0.854	0.886	0.840	**0.500**
	Sorting 032	0.975	0.990	0.985	0.972	0.930
	Document Merging	0.740	0.870	0.870	0.800	0.750
	TruthfulQA	0.680	0.900	0.920	0.840	0.780
ReAct	GSM8K	0.760	0.960	0.980	0.920	0.857
	GSM-Symbolic	0.873	0.933	0.867	0.800	**0.630**
	MMLU	0.700	0.880	0.800	0.640	**0.480**
	AQUA	**0.600**	0.900	0.900	0.671	**0.400**
	SVAMP	0.730	0.920	0.880	0.820	0.800
	Sorting 032	0.980	1.000	1.000	0.960	0.923
	Document Merging	0.735	0.870	0.870	0.760	0.620
	TruthfulQA	0.780	0.880	0.980	0.900	0.900
Tree of Thoughts	GSM8K	0.640	0.950	0.970	0.860	0.760
	GSM-Symbolic	0.707	0.751	**0.521**	0.794	0.647
	MMLU	0.769	**0.653**	0.867	**0.640**	0.720
	AQUA	0.670	0.881	0.867	0.740	0.660
	SVAMP	0.760	0.935	0.867	0.750	0.570
	Sorting 032	0.982	0.998	0.998	0.985	0.881
	Document Merging	0.814	0.898	0.881	0.873	0.774
	TruthfulQA	0.940	0.960	0.990	0.980	0.940
Graph of Thoughts	GSM8K	0.780	0.960	0.980	0.900	0.800
	GSM-Symbolic	0.733	**0.813**	0.813	0.813	0.793
	MMLU	**0.440**	**0.680**	**0.600**	**0.700**	**0.500**
	AQUA	0.700	0.840	0.900	0.840	0.760
	SVAMP	0.800	0.920	0.940	0.900	0.820
	Sorting 032	0.993	0.998	0.997	0.990	0.878
	Document Merging	0.808	0.902	0.886	0.812	0.743
	TruthfulQA	0.960	0.960	0.980	0.960	0.980
LLM Multi-Agent Debate	GSM8K	0.810	0.920	0.904	0.780	0.610
	GSM-Symbolic	0.833	0.823	0.971	0.901	**0.650**
	MMLU	0.720	0.880	0.854	0.625	0.540
	AQUA	0.680	0.960	0.780	0.800	0.640
	SVAMP	0.902	0.958	0.916	0.833	0.800
	Sorting 032	0.989	0.998	0.997	0.990	0.960
	Document Merging	0.780	0.890	0.870	0.830	0.785
	TruthfulQA	0.800	0.880	0.900	0.920	0.880