DynaThink: Fast or Slow? A Dynamic Decision-Making Framework for Large Language Models

Jiabao Pan
Cambridge University
Zhejiang University
jp2100@cam.ac.uk

Yan Zhang
National University
of Singapore
eleyanz@nus.edu.sg

Chen Zhang
National University
of Singapore
chen_zhang@u.nus.edu

Zuozhu Liu
Zhejiang University
zuozhuliu@intl.zju.edu.cn

Hongwei Wang
Zhejiang University
hongweiwang@intl.zju.edu.cn

Haizhou Li
The Chinese University of Hong Kong
haizhou.li@nus.edu.sg

Abstract

Large language models (LLMs) have demonstrated emergent capabilities across diverse reasoning tasks via popular Chains-of-Thought (COT) prompting. However, such a simple and fast COT approach often encounters limitations in dealing with complicated problems, while a thorough method, which considers multiple reasoning pathways and verifies each step carefully, results in slower inference. This paper addresses the challenge of enabling LLMs to autonomously select between fast and slow inference methods, thereby optimizing both efficiency and effectiveness. We introduce a dynamic decision-making framework that categorizes tasks into two distinct pathways: 'Fast', designated for tasks where the LLM quickly identifies a high-confidence solution, and 'Slow', allocated for tasks that the LLM perceives as complex and for which it has low confidence in immediate solutions as well as requiring more reasoning paths to verify. Experiments on five popular reasoning benchmarks demonstrated the superiority of the DynaThink over baselines.Our code can be found at https://github.com/dian1414/Dynathink/tree/main

1 Introduction

LLMs (OpenAI, 2022; Touvron et al., 2023; OpenAI, 2023) have emerged as prominent foundation models for diverse applications due to their outstanding ability to understand and generate human-like text. One notable attribute of LLMs is their capability of COT reasoning (Wei et al., 2022), where they can perform multi-step reasoning using only a few demonstrations. However, the performance of such simple and fast COT prompting is not satisfactory for complex reasoning problems, which often need more careful thought and analysis (Stanovich and West, 2000). To improve this, Wang et al. (2022) introduced a self-consistency strategy as a replacement for the previous straightforward decoding method used in COT prompting. This self-consistency strategy involves trying out multiple reasoning paths and selecting the most consistent answers, significantly improving LLM performance in different reasoning tasks. Recent advanced works (Yao et al., 2023; Besta et al., 2023; Shinn et al., 2023; Madaan et al., 2023; Miao et al., 2023) expanded on this strategy by considering different reasoning paths and self-evaluation to make holistic decisions. However, it's important to note that these methods require more resources and therefore slow down the decision-making process, especially for high-cost LLMs such as GPT-4 (OpenAI, 2023). This highlights the need for a method that lets LLMs adjust resource use based on the specific needs and complexities of tasks, which is crucial for developing and using LLMs effectively in real-world situations, allowing these models to provide the best value in various applications while managing resources wisely.

To address the challenge, we introduce a dynamic decision-making framework called "DynaThink." This framework helps LLMs make smart choices between quick and thorough ways of solving problems. It optimizes a balance between being efficient and effective in various reasoning tasks. In this framework, we categorize tasks into two groups: "Fast" for tasks where LLMs can quickly find confident answers, and "Slow" for more complex tasks that require thorough exploration of different reasoning paths to confirm answers. Establishing rules to distinguish between fast and slow reasoning is challenging. We adopt two criteria including consistency verification and reasoning complexity verification. The principle of consistency verification is reflective of the human heuristic, wherein the concurrence of multiple distinct thought processes on a singular answer augments the confidence of its correctness (Kahneman, 2011). Different from the straightforward majority voting used in (Wang et al., 2022), which sometimes leads

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 14686–14695

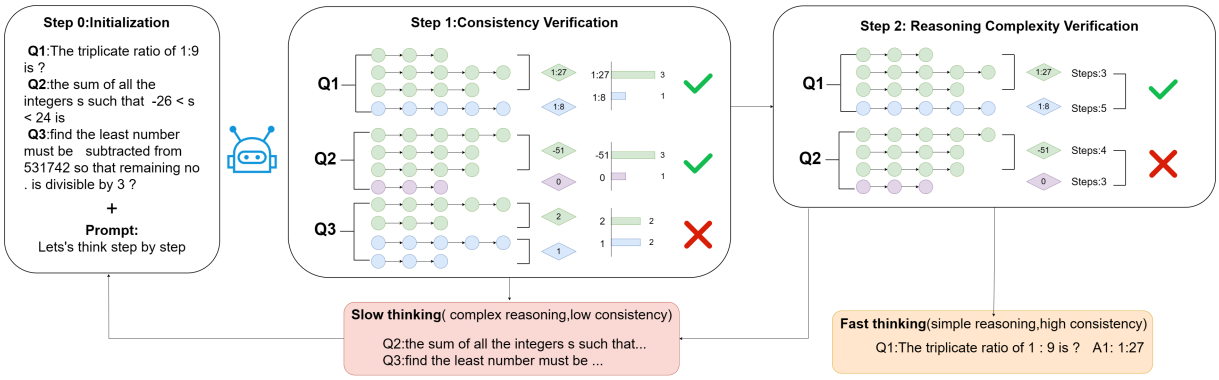


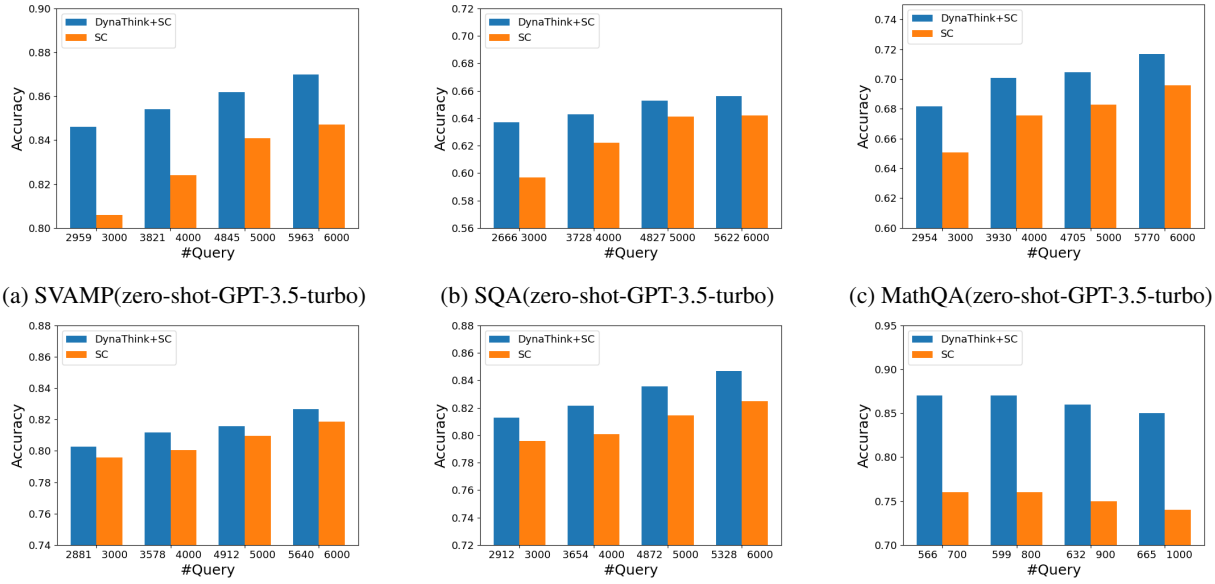
Figure 1: The workflow starts by incorporating the widely used CoT prompt, like 'let's think step by step' and then initially querying LLMs four times for each question (which can be fewer than four to begin with). In the first step, we choose question 1 and question 2 because they have one answer with more than half the votes. However, question 3 has a tie, with two answers getting equal votes, so we consider it a slow-thinking question. Next, we look at questions 1 and 2 to see how many steps are needed for each answer. For question 2, the answer with the most votes takes four steps, while the other answer needs three steps, so question 2 is also categorized as a slow-thinking question. Regarding question 1, we classify it as a fast-thinking question because the answer with the most votes also requires the fewest steps, allowing us to output this answer directly. However, for slow-thinking questions, we require additional iterations to make a selection.

to problems like tie-breakers, we only validate answers when they secure more than half of the total votes, indicating they are high-confidence solutions with the best votes. On the other hand, our reasoning complexity verification criteria are based on reasoning processes with a minimum number of reasoning steps. LLMs can make mistakes during the reasoning process (Madaan et al., 2023; Ling et al., 2023), while each reasoning step of LLMs can be thought of as a decision made under uncertainty. The accumulation of these decisions can lead to a compounding of uncertainty, reducing the overall confidence in the final decision. Therefore, fewer reasoning steps often yield more reliable and confident outcomes due to reduced error propagation.To empirically corroborate the two criteria, we have conducted experiments on multiple reasoning tasks and found a strong correlation between the efficacy of LLMs in problem-solving and both the length of their reasoning paths and the voting of the outputs. Questions that meet these criteria are sorted using simple and fast decision making processes. The rest of the questions are subject to more exploration of different reasoning paths and comprehensive self-evaluations (Yao et al., 2023; Besta et al., 2023; Miao et al., 2023).

2 DynaThink

DynaThink, as shown in Figure 1 [1], starts by setting the initial query times for LLM, which in this case is 2. The LLM is then queried with the problem set P, yielding the response set Q. First, we employ consistency verification to analyze the diversity of answers each question receives. Questions whose answers get over half the votes are placed in Q_1, serving as potential fast-thinking questions. The rest go into Q_2 for more review in the following rounds with added resources. Next, we utilize reasoning complexity verification to examine the step counts needed for each response, picking the one with the fewest steps throughout all generations. We then see if this selected answer from Q_1 indeed has the least steps. If it does, it goes into Q_3 as a fast-thinking question of this round. If not, it returns to Q_2 for more rounds of checking. Lastly, we see if we've picked any fast-thinking questions this round. If Q_3 has questions, we proceed to another iteration; if it's empty, we conclude, leaving the questions in Q_2 as the slow-thinking ones. For the fast-thinking questions, we can easily arrive at an answer (more than half vote + least number of reasoning steps). For the slow questions, we repeat the same CoT + self-consistency procedure with more queries to the LLMs so as to identify the answer with more than half vote and the least number of reasoning steps.

[1]The algorithm presudo code is shown in appendix A



(a) SVAMP(zero-shot-GPT-3.5-turbo) (b) SQA(zero-shot-GPT-3.5-turbo) (c) MathQA(zero-shot-GPT-3.5-turbo)

(d) GSM8K(zero-shot-Gemini) (e) GSM8K(zero-shot-GPT-4) (f) GSM8K(zero-shot-Mixtral-8x7B)

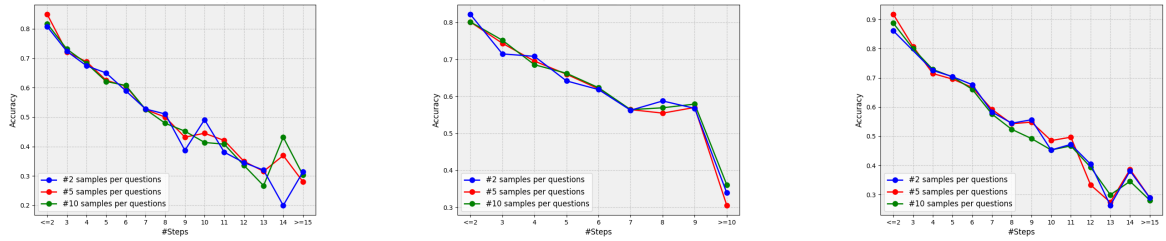
Figure 2: SC: the original self-consistency approach that uses majority voting to identify the most agreed-upon answer (Wang et al., 2022). DynaThink+SC: divides the question sets into fast and slow-thinking categories and applies different selection criteria to the answers from each set. The CoT prompting technique is utilized by both SC and DynaThink+SC, and this is applied in both zero-shot and few-shot settings (Wei et al., 2022; Kojima et al., 2022). To ensure a fair comparison, DynaThink+SC also utilizes the SC strategy for the slow-thinking question set, to determine the final answer. However, there is a slight adjustment in the number of queries for LLM to accommodate the different processing requirements of the fast and slow-thinking question sets. It's important to highlight that we always ensure that the operational cost of using DynaThink+SC is either lower or competitive with the use of the SC strategy. In essence, the goal of DynaThink+SC is to optimize efficiency and cost-effectiveness. SQA means StrategyQA. Due to space limitations, more results are presented in appendix I.

3 Experiments

Setup We evaluate the efficiency and effectiveness of DynaThink on six diverse reasoning datasets, i.e., StrategyQA (Geva et al., 2021), GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), AQUA-RAT (Ling et al., 2017), SVAMP (Patel et al., 2021) and MATHQA (Amini et al., 2019).[2] We use three popular black-box LLMs like GPT-3.5-turbo, GPT-4, Gemini as well as one popular open-source LLM, i.e., Mixtral-8x7B (Jiang et al., 2024) as the backbones. We employ the self-consistency strategy (Wang et al., 2022) within the CoT reasoning as our main baseline following (Ling et al., 2023; Lightman et al., 2023; Miao et al., 2023). Although it's simple, it's still a strong method according to current research. Additionally, for all the open-source models, we set the temperature to 0.7.

Main Results Figure 2 presents the main results. We can observe that DynaThink consistently enhances both the effectiveness and efficiency of SC across a range of reasoning tasks in both zero-shot and few-shot scenarios. For instance, within the zero-shot MATH setting, utilizing GPT-3.5-Turbo, DynaThink attains an accuracy rate of 45% with 2758 LLM queries, outperforming SC, which secures an accuracy of 41.9% using the same number of LLM queries. Likewise, in the few-shot MathQA context, our approach exhibits a 4% improvement in accuracy with 2849 LLM queries, in contrast to SC's performance with 3000 LLM queries.At the same time, the horizontal axis representing 'query', i.e., the total number of model accesses, can also serve as an indicator of the time required for result generation. This implies that we achieved better results than the baseline in a comparable or even shorter time. The integration of consistency checks and reasoning complexity verification as criteria is deemed crucial for this enhanced performance. These criteria provide an advanced method for accurately evaluating confidence levels, minimizing the build-up of uncertainties, and improving resource allocation and efficiency, thereby significantly increasing the accuracy of solutions.

[2]The statistics can be found in the Appendix.



(a) AQuA(zero shot) (b) GSM8K(zero shot) (c) MathQA(zero shot)

Figure 3: Correlation between the distribution of reasoning steps and reasoning performance. We employed the self-consistency strategy within the zero-shot and few-shot CoT prompting techniques, to query the GPT-3.5-Turbo model two, five, and ten times for each question.Due to space limitations, more results are presented in appendix E

In addition, we can find that, GPT-4 demonstrates the best performance among the three evaluated LLMs. Despite the high baseline established by such a top-performing LLM, DynaThink is still able to exhibit superior performance. For example, in the zero-shot MATHQA scenario, DynaThink achieves an accuracy of 73.8% with 2827 LLM queries, surpassing SC, which attains an accuracy of 71.7% with 3000 LLM queries. Similarly, in the zero-shot GSM8K scenario, DynaThink achieves an accuracy of 81.6% with 2912 LLM queries, exceeding SC's performance, which is 79.4% accuracy with 3000 LLM queries. These findings suggest that the DynaThink framework generalizes effectively across various LLMs.

Ablation Study of Reasoning Complexity Verification We study the correlation between the distribution of reasoning steps and reasoning performance. Specifically, we conducted a comprehensive analysis by randomly selecting 200 questions from each of the AQuA, GSM8K, and MathQA datasets to compile a development set. The study employed the self-consistency strategy, within the zero-shot and few-shot CoT prompting techniques to query the GPT-3.5-Turbo model two, five, and ten times per question. The findings, illustrated in Figure 3, indicate a consistent pattern: an increase in the number of reasoning steps correlates with a reduction in the LLM's reasoning performance. This trend is evident across both zero-shot and few-shot learning paradigms.The conclusion also holds true for the same question.(shown in appendix G).

The results from this investigation lend substantial support to our initial hypothesis, which posits that minimizing the number of reasoning steps serves as a practical metric for verifying reasoning complexity. This approach is pivotal in assessing the confidence level of the LLMs' reasoning outcomes, offering valuable insights into achieving an ideal equilibrium between the extent of reasoning steps and the efficacy of model performance. It underscores the significance of maintaining efficiency and succinctness in the reasoning process to procure responses from LLMs with elevated confidence, regardless of the dataset or prompting method utilized.

4 Related Work

The introduction of CoT prompting by (Wei et al., 2022) has been a seminal development in highlighting the multi-step reasoning capabilities of LLMs. This approach has paved the way for various refinements and enhancements (Zhou et al., 2022; Wang et al., 2022; Fu et al., 2022; Zelikman et al., 2022; Yao et al., 2023; Besta et al., 2023), to enhance the effectiveness of LLMs in addressing complex problems. For further improvement of the result of CoT, some methods like Wang et al. (2022) and Yao et al. (2023) concentrate on the optimization of reasoning structures and pathways.While others focused on careful planning and verification of the reasoning process.For example, (Lightman et al., 2023) introduced the PRM800K dataset, which is notable for its inclusion of step-wise correctness labels obtained through crowdsourcing.And (Miao et al., 2023) extracted insights from multiple reasoning steps and conducted sequential verifications to rectify inconsistencies.

While these works have demonstrated significant advancements in handling complex problems, they often require significantly increased resource. Balancing resource expenditure and efficiency is essential. This research aims to streamline budget allocation and resource utilization while maintaining high efficacy and efficiency in LLM reasoning tasks. The goal is to achieve a harmonious balance

between resource deployment and optimal performance, allowing for an efficient and resourceful approach to LLM reasoning.

5 Conclusion

In this paper, we introduced a dynamic decision-making framework called DynaThink, which enables LLMs to make smarter choices between fast and slow problem-solving methods, striking a balance between efficiency and effectiveness. Across various reasoning tasks, DynaThink outperforms existing self-consistency methods, enhancing accuracy while optimizing computational resources.

6 Limitations

DynaThink has limitations. The dichotomy of 'Fast' and 'Slow' thinking may oversimplify the complexity of problems, potentially overlooking tasks of intermediate difficulty. This recognition motivates our future work, focusing on refining task categorization methods to enable a more nuanced and adaptable approach. This will empower LLMs to handle a wider range of problem complexities with greater precision and adaptability. The insights gained from this research will drive advancements in the field, leading to the development of LLMs better equipped to handle diverse reasoning challenges.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *ArXiv*, abs/1905.13319.

Maciej Besta, Nils Blach, Ale Kubek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2023. Graph of thoughts: Solving elaborate problems with large language models. *ArXiv*, abs/2308.09687.

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. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Yao Fu, Hao-Chun Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2022. Complexity-based prompting for multi-step reasoning. *ArXiv*, abs/2210.00720.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Xiaodong Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *ArXiv*, abs/2103.03874.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, L'elio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *ArXiv*, abs/2401.04088.

Daniel Kahneman. 2011. Thinking, fast and slow.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *ArXiv*, abs/2205.11916.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *ArXiv*, abs/2305.20050.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Annual Meeting of the Association for Computational Linguistics*.

Z. Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. Deductive verification of chain-of-thought reasoning. *ArXiv*, abs/2306.03872.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *ArXiv*, abs/2303.17651.

Ning Miao, Yee Whye Teh, and Tom Rainforth. 2023. Selfcheck: Using llms to zero-shot check their own step-by-step reasoning. *ArXiv*, abs/2308.00436.

OpenAI. 2022. Chatgpt. *https://chat.openai.com/*.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Arkil Patel, S. Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *North American Chapter of the Association for Computational Linguistics*.

Noah Shinn, Beck Labash, and Ashwin Gopinath. 2023. Reflexion: an autonomous agent with dynamic memory and self-reflection. *ArXiv*, abs/2303.11366.

Keith E. Stanovich and Richard F. West. 2000. Individual differences in reasoning: Implications for the rationality debate? *Behavioral and Brain Sciences*, 23:645 – 665.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *ArXiv*, abs/2305.10601.

E. Zelikman, Yuhuai Wu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *ArXiv*, abs/2203.14465.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*.

A Algorithm

The algorithm presents how Dynathink chooses the threshold and recursively separates the whole question set into to subsets.

Algorithm 1 DynaThink

Require: Problem set, P
Ensure: Fast thinking question set, Q_f and Slow thinking question set, Q_s

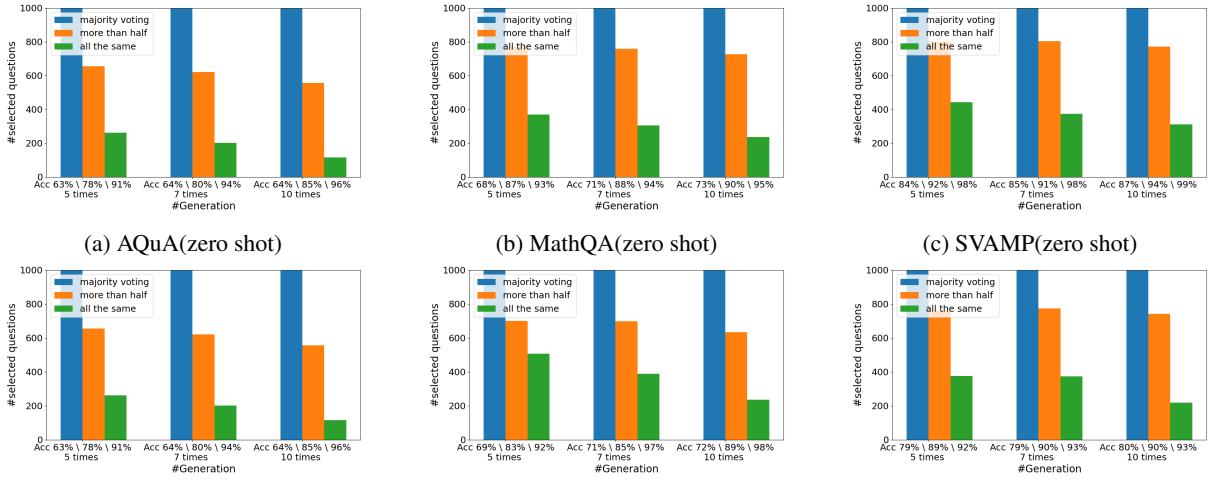
$Q_f \leftarrow \varnothing, Q_s \leftarrow \varnothing$ ▷ Initialize set for fast questions and slow questions
$n \leftarrow 2$ ▷ n can be initialized by any integer less than total generation times.
repeat
    Generate n responses for problem set P by querying the LLM; store as Q.
    Initialize question set Q_1, Q_2 and Q_3 as $\varnothing$.
    Calculate voting distribution F for each question in Q based on consistency.
    for each question i in Q **do**
        Determine the answer with the highest votes, a_j, and its vote count, $max(F(i))$.
        if $max(F(i)) \geq \lfloor \frac{n}{2} \rfloor + 1$ **then**
            Add $Q(i)$ to Q_1 ▷ First selected question set
        else
            Add $Q(i)$ to Q_2 ▷ Set for slow questions
        end if
    end for
    for each question in Q_1 **do**
        Extract the minimum step array from each answer and determine the step distribution.
        for each question i **do**
            Find the minimum steps, $min(Steps(i))$, and the step of the majority-voted answer, a_i.
            if $a_i == min(Steps(i))$ **then**
                Add $Q_1(i)$ to Q_3
            else
                Add $Q_1(i)$ to Q_2
            end if
        end for
    end for
    if $Q_3 \neq \varnothing$ **then**
        Update Q_f with $Q_f \cup Q_3$
        Update P with the questions in Q_2
        increase n ▷ n can increase by any integer based on the budget limit
    else
        $Q_s = Q_2$
    end if
until $Q_3 = \varnothing$ ▷ Continue until Q_3 is $\varnothing$
return Q_f, Q_s

B Prompt Template

To let the large language models solve problems in steps, we use the following prompt as shown in Table 1.

C Ablation Study of Consistency Verification

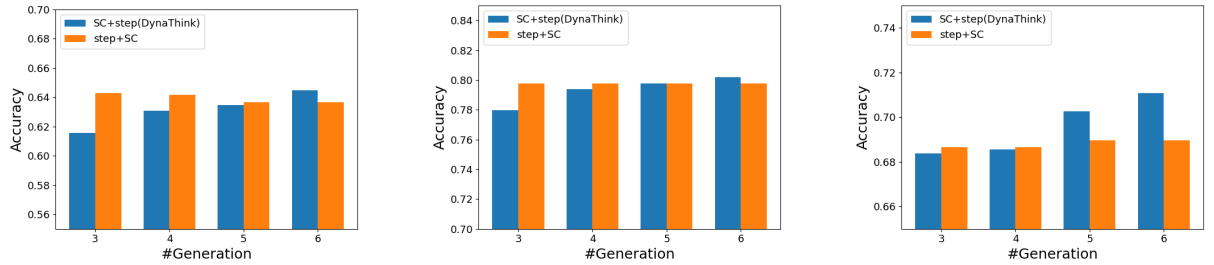
In the development of our DynaThink framework, we have chosen to adopt the "over half votes" cri-



(a) AQuA(zero shot) (b) MathQA(zero shot) (c) SVAMP(zero shot)

(d) AQuA(few shot) (e) mathqa(few shot) (f) GSM8K(few shot)

Figure 4: Ablation Study of Consistency Verification. We employed the zero-shot and few-shot COT prompting techniques, to query the GPT-3.5-Turbo five, seven and ten times for each question. Three thresholds for consistency verification in DynaThink are considered, i.e., majority voting, more than half and all the same.



(a) AQuA(zero shot) (b) GSM8K(zero shot) (c) MathQA(zero shot)

Figure 5: Ablation Study of Verification Order. SC + Step (DynaThink): initially deployed with consistency verification, followed by reasoning complexity verification. Step + SC: initially deployed with reasoning complexity verification, followed by consistency verification. The zero-shot COT prompting technique is utilized to query the GPT-3.5-Turbo six times.

========== PROMPT EXAMPLE ==========

Solve the following problem step by step. Please start each step with "Step :"

[The problem description]

[Answer]
Step : .....
Step : .....
...
...
...

Table 1: Instruction template to prompt ChatGPT to solve the problems in steps.

terion as our threshold within the Consistency Verification process. This decision is elucidated by examining the outcomes associated with three distinct voting thresholds: "majority voting," "more than half," and "all the same." These thresholds were evaluated across various datasets, including MathQA, AQuA, and SVAMP, utilizing the COT prompting technique to query the GPT-3.5-Turbo model five, seven, and ten times for each question.

The empirical results, as depicted in Figure 4, reveal that the "all the same" voting threshold does indeed enhance the accuracy for the questions selected under this criterion. Nevertheless, this method results in the exclusion of a significant portion of potential questions. On the other hand, the "more than half" voting threshold allows for the inclusion of approximately 80% of the questions, with only a 4%-6% decrease in accuracy com-

pared to the "all the same" voting threshold, which accommodates about 30% of all questions. This trend holds true across both the AQuA and SVAMP datasets. These findings suggest that while uniformity in responses can increase the accuracy of the chosen question set, it simultaneously limits the overall number of questions that can be considered. By implementing the "over half votes" criterion, it is possible to secure a subset of questions with relatively high accuracy while also including a larger portion of the questions, thus achieving an optimal balance between quality and quantity.

D Ablation Study of Verification Order

This section conducts a thorough analysis of the viability of reversing the sequence of these verifications by inverting the order in which they are conducted. The study is assessed using MathQA, AQuA, and GSM2K test set. The zero-shot COT prompting techniques have been utilized to query the GPT-3.5-Turbo for each question multiple times. The implications of prioritizing step selection before assessing consistency are illustrated in Figure 5. It can be observed that initiating with reasoning complexity verification may yield higher initial results, yet the overall accuracy tends to stabilize, and in some cases it even experiences a decrease. Conversely, the employment of DynaThink fosters a stable enhancement in results.

E The rest data of ablation study of reasoning complexity verification

All the results of few-shot and part of the results of zero-shot are shown in Figure 6.

F Number of test data

Dataset	Total of data
MATH	700
MathQA	1000
SVAMP	1000
GSM8K	1000
AQuA	1000
StrategyQA	1000

Table 2: Number of test data

G Supplementary experiments for ablation study of reasoning complexity verification

To demonstrate that the complexity feature remains effective for the same question, we compared two conditions: selecting only the longest step response and selecting only the shortest step response of the same question as the result. The following table shows that accuracy is higher when choosing the shortest step responses than the longest step responses, which means that for the same question, a shorter response can be better than a longer one.

MathQA		
	Few-shot	Zero-shot
Longest	0.580	0.623
Shortest	0.659	0.632

AQuA		
	Few-shot	Zero-shot
Longest	0.460	0.479
Shortest	0.556	0.576

GSM8K		
	Few-shot	Zero-shot
Longest	0.624	0.612
Shortest	0.685	0.682

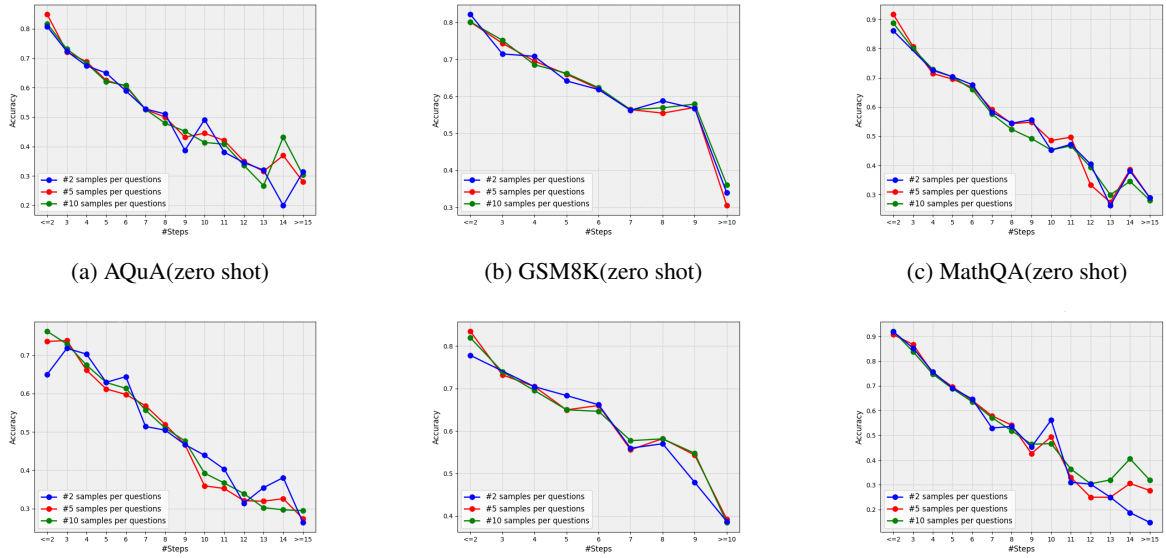
Table 3: Performance comparison across MathQA, AQuA, and GSM8K datasets (Few-shot vs Zero-shot).

H Generalization to SelfCheck

We have demonstrated the superiority of the DynaThink framework when compared to the self-consistency baseline. In this section, we integrate DynaThink with SelfCheck[3] (Miao et al., 2023), a recent method that presents a more complex strategy beyond the linearly structured CoT approach, by incorporating careful self-evaluations. We evaluate this enhanced method, which we have termed DynaThink+SelfCheck, using the MathQA dataset. The experimental setting was designed in alignment with the zero-shot setting, as detailed in the work of (Miao et al., 2023)

Our observations from this experiment reveal that DynaThink+SelfCheck not only maintains a commendable accuracy level comparable to that of the original SelfCheck method—preserving a

[3] https://github.com/NingMiao/SelfCheck



(a) AQuA(zero shot) (b) GSM8K(zero shot) (c) MathQA(zero shot)

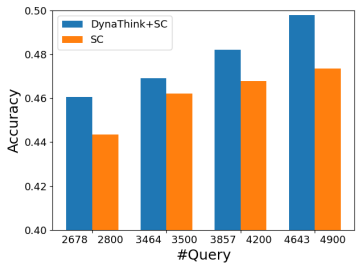
(d) AQuA(few shot) (e) GSM8K(few shot) (f) MathQA(few shot)

Figure 6: The rest of data in ablation study of reasoning complexity verification.

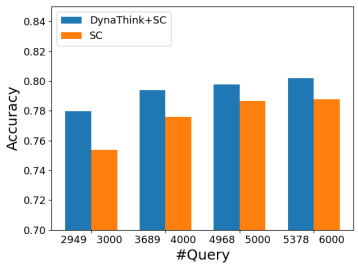
72% accuracy rate—but it also achieves a significant efficiency gain by reducing the number of GPT-3.5-Turbo queries to merely one-quarter of those utilized by the original SelfCheck approach. This finding underscores the effectiveness of DynaThink+SelfCheck in enhancing the operational efficiency of solving complex problems while retaining high accuracy,showcasing its potential as a formidable tool in the domain of advanced reasoning and problem-solving.

I The rest data of main results

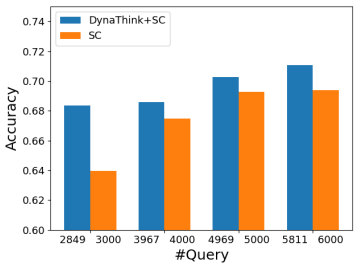
All the results of few-shot and part of the results of zero-shot are shown in Figure 7.



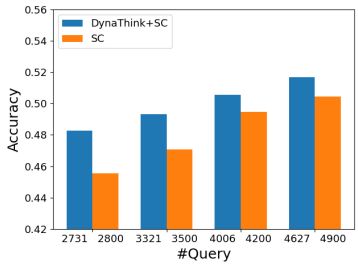
(a) MATH(few shot-GPT-3.5-turbo)



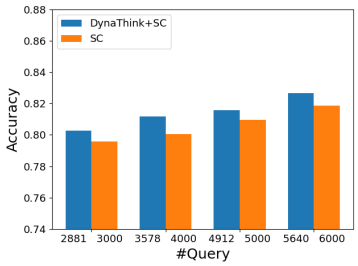
(b) GSM8K(few shot-GPT-3.5-turbo)



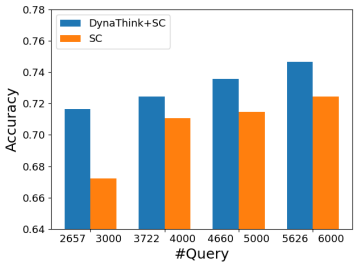
(c) MathQA(few shot-GPT-3.5-turbo)



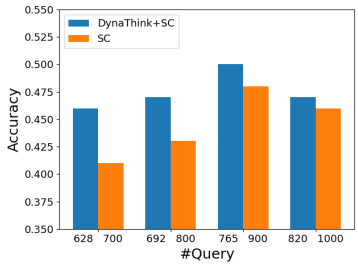
(d) MATH(zero-shot-Gemini)



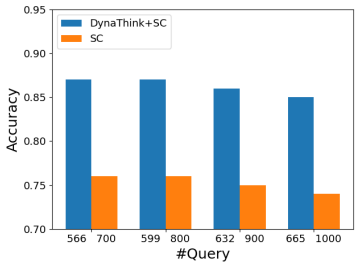
(e) GSM8K(zero-shot-Gemini)



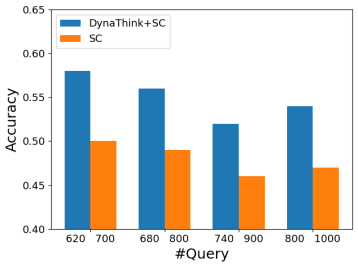
(f) MathQA(zero-shot-Gemini)



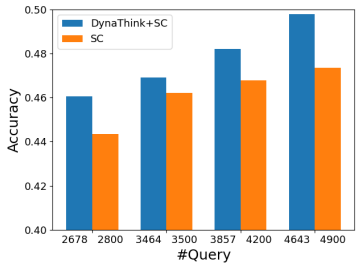
(g) AQuA(Mixtral-8x7B-v0.1)



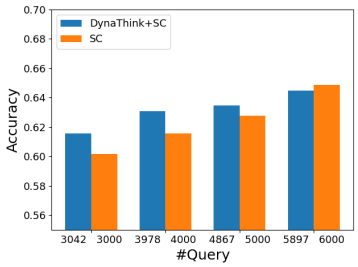
(h) GSM8K(Mixtral-8x7B-v0.1)



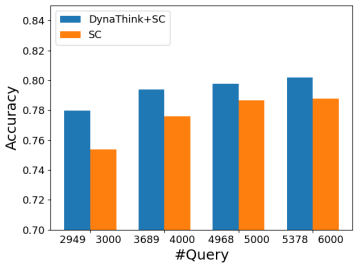
(i) MathQA(Mixtral-8x7B-v0.1)



(j) MATH(few shot-GPT-3.5-turbo)



(k) AQuA(few shot-GPT-3.5-turbo)



(l) GSM8K(few shot-GPT-3.5-turbo)

Figure 7: The rest of data in main results.