Thinking Before You Speak: A Proactive Test-time Scaling Approach

Cong Liu, Wenchang Chai[†], Hejun Wu, Yan Pan, Pengxu Wei, Liang Lin
Sun Yat-sen University, [†]Hong Kong Polytechnic University
liucong3@mail.sysu.edu.cn, wenchang.chai@connect.polyu.hk,
wuhejun.sysu.edu.cn, panyan5@mail.sysu.edu.cn,
weipx3@mail.sysu.edu.cn, linliang@ieee.org

Abstract

Large Language Models (LLMs) often exhibit deficiencies with complex reasoning tasks, such as maths, which we attribute to the discrepancy between human reasoning patterns and those presented in the LLMs' training data. When dealing with complex problems, humans tend to think carefully before expressing solutions. However, they often do not articulate their inner thoughts, including their intentions and chosen methodologies. Consequently, critical insights essential for bridging reasoning steps may be absent in training data collected from human sources. To bridge this gap, we proposes inserting *insight*s between consecutive reasoning steps, which review the status and initiate the next reasoning steps. Unlike prior prompting strategies that rely on a single or a workflow of static prompts to facilitate reasoning, *insight*s are *proactively* generated to guide reasoning processes. We implement our idea as a reasoning framework, named *Thinking Before You Speak* (TBYS), and design a pipeline for automatically collecting and filtering in-context examples for the generation of *insight*s, which alleviates human labeling efforts and fine-tuning overheads. Experiments on challenging mathematical datasets verify the effectiveness of TBYS. Project website: https://gitee.com/jswrt/TBYS

1 Introduction

OpenAI's O1 (OpenAI, 2024) demonstrates the potential of leveraging long chains of thought (CoT) (Wei et al., 2022) to enhance the reasoning capabilities of large language models (LLMs). Through its generated reasoning, O1 exhibits advanced cognitive skills, such as problem decomposition, error identification, and correction – processes that continuously guide thinking toward accurate solutions. Inspired by this, various test-time scaling (Snell et al., 2024; Zhang et al., 2025) approaches were proposed, such as using prompts

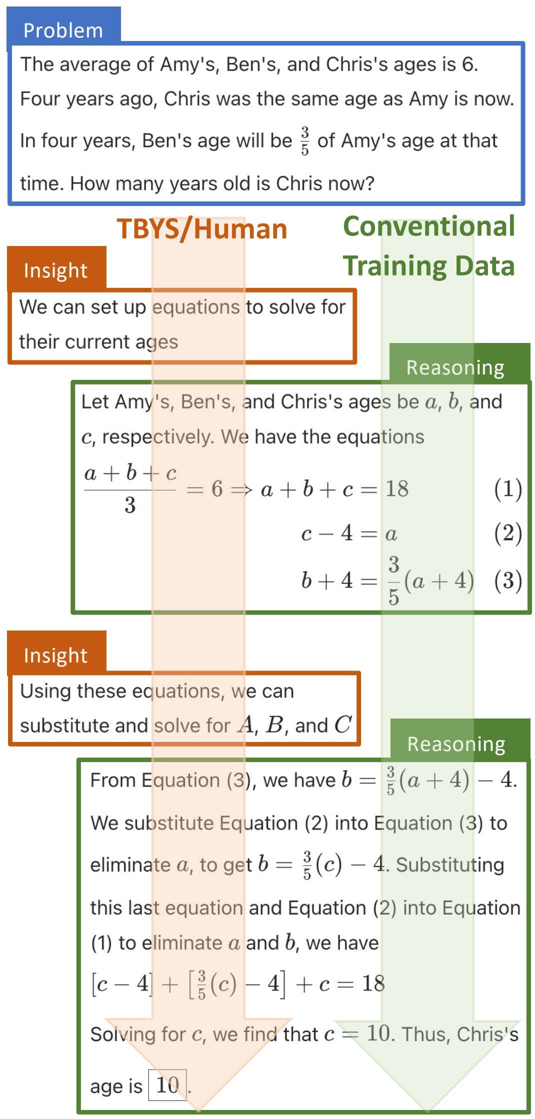

Figure 1: A simplified example to compare the reasoning trace of human and TBYS against one from conventional training data. Humans and TBYS excel with a flow of insight-driven reasoning that is more comprehensible. On the other hand, the training set example adds to the difficulty of learning, as it is not always straightforward to re-engineer the connection between consecutive steps behind the succinct reasoning logic. TBYS proactively fill reasoning gaps with *insight*s representing intention, explanation, or justification, etc.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 24643–24650
November 4-9, 2025 ©2025 Association for Computational Linguistics

like "Wait," (Muennighoff et al., 2025) to stimulate self-correction, "Wait, using Python" to encourage coding (Li et al., 2025a), or fixed workflows of prompts to structure inferences (Hong et al., 2024). However, these methods suffer from task and LLM sensitivity. For instance, certain agentic workflows (e.g., MetaGPT (Hong et al., 2024)) may improve coding tasks but not Q&A performance. Similarly, LLMs exhibit sensitivity to prompt design, including style and example ordering (Zhuo et al., 2024). As a result, they are most effective when paired with reinforcement learning techniques (e.g., rejection sampling) to filter suboptimal cases, but are ill-suited for direct application to scale reasoning at test time.

This paper introduces a novel prompting paradigm called **proactive prompting**, where an LLM proactively generates prompts to steer its own reasoning steps, rather than passively reacting to predefined prompting patterns. This approach demonstrates particular advantages in complex reasoning tasks, such as advanced mathematics problems, where the proactive generation of "inner thoughts" (critical for guiding reasoning) is often absent from final reasoning outputs in conventional training data.

To validate this paradigm, we develop a reasoning framework named *Thinking Before You Speak* (**TBYS**), which iteratively inserts a proactive prompt – termed the *insight* – before each reasoning step to explicitly define the status and the goal of that step. Figure 1 contrasts a TBYS reasoning process with that in conventional training data (with which LLMs are trained). TBYS mirrors human inner-thinking patterns, producing more explainable reasoning traces that facilitate LLM learning and offering greater educational values for human readers.

In the remainder of this paper, we detail the TBYS reasoning framework in Section 2. Since TBYS relies on iteratively generating insights to guide reasoning, the quality of these generated insights is critical to its accuracy. To address this, we employ in-context learning with examples retrieved from a library of insight exemplars. Section 3 describes our pipeline for automatically collecting, filtering, and selecting example *insights* for this library. Section 4 briefly reviews prior related work. Finally, Section 5 evaluates TBYS against strong baselines on challenging datasets, demonstrating significant performance improvements and better accuracy-overhead trade-offs. We further conduct ablation studies to validate the contributions of key components.

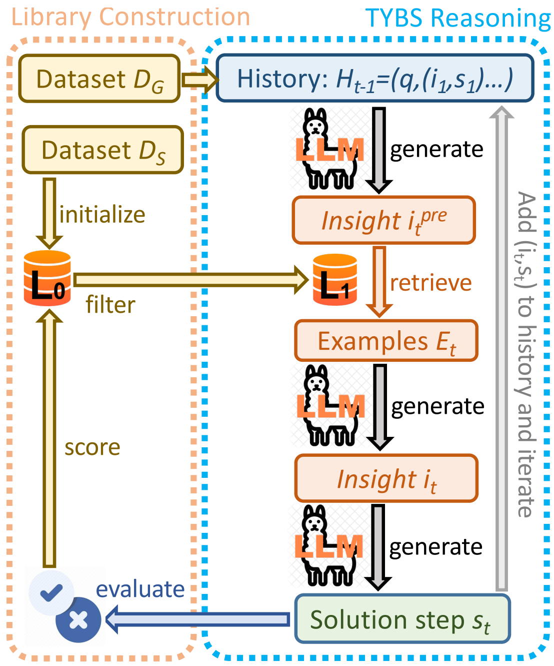

Figure 2: The TBYS reasoning framework (Section 2) and *insight* library construction (Section 3).

2 The TBYS Reasoning Framework

TBYS utilizes a library L of high-quality *insight*s. The automatic construction of this library is detailed in Section 3. During inference, examples are retrieved from L using some off-the-shelf embedding model for in-context learning. We also manually define three seed examples S, each containing a question and the complete reasoning steps for the question with the associated *insight*s.

As shown in Figure 2, TBYS employs a multi-round reasoning approach. Each round t consists of three steps: (1) *Insight Generation*: A preliminary *insight* i_t^{pre} is generated based on the current reasoning history $H_{t-1} = (q, (i_1, s_1), (i_2, s_2), \cdots, (i_{t-1}, s_{t-1}))$, where q is the question, and i_j, s_j denote the *insight* and solution step in round j, respectively. (2) *Example Retrieval*: Each *insight* is defined by its two components: *situation* (summarizing the current reasoning status) and *goal* (stating the intention for solution step s_t). The situation of i_t^{pre} is used to retrieve $k_E = 8$ examples E_t from library L. Using these k_E high-quality *insight*s as in-context examples, a refined *insight* i_t is generated. (3) *Solution Step Generation*: The solution step s_t is generated using H_{t-1} and i_t, then appended to H_{t-1} to form

H_t. To signal the end of reasoning, s_t includes a field indicating whether a confident answer to q has been reached.

3 Construction of the *Insight* Library

As shown in Figure 2, we build the library of *insights* in two stages: initialization and filtering.

Initialization: We use manually curated seed examples S and a dataset D_S containing questions and their chain-of-thought solutions. First, an LLM is prompted to split each solution in D_S into 1–3 steps. The LLM is then prompted again to generate an *insight* i_t for each solution step s_t, consisting of a *situation*, which should represents the reasoning status up to that step, and a *goal*, which should offers a purpose and a guideline to stimulate the LLM to reproduce solution step s_t. All *insights* and divided solution steps are collected into an initial library L_0.

Filtering: To identify high-quality *insights*, we use a dataset D_G (containing questions and ground-truth answers) and a scoring mechanism: (1) For each *insight* $i_i \in L_0$, maintain counters r_i (correct uses) and w_i (wrong uses). (2) Evaluate L_0 by running *TBYS* on each question $q \in D_G$. For each reasoning step for q, retrieve $k_F = 25$ examples from L_0 and randomly select one as a 1-shot example. If the reasoning yields a correct answer, increment r_i for each i_i used; otherwise, increment w_i. (3) Rank *insights* in L_0 by the score $\frac{r_i}{r_i+w_i}\log(r_i + w_i)$, which balances accuracy and usage coverage. Select the top-k_L examples to form L_1. The insight library can be progressively improve through multiple iterations. In each iteration, the initial library L_0 is updated to include the filtered library L_1 and the newly generated insights from dataset D_G, which is produced during the filtering of L_1.

In our experiments, the MATH-500 dataset (Lightman et al., 2023) serves as D_S and the test set, e.g., MATH-500 or AIME (Zhang et al., 2023a), serves as D_G in a test-time adaptation (Jang et al., 2023) manner, with k_L as a variable parameter.

4 Related Work

Extensive research has investigated prompt designs to improve LLM reasoning, including *Chain-of-Thought* (Wei et al., 2022), *Least-to-Most* (Zhou et al., 2023), *Self-Consistency* (Wang et al., 2023b), and *Tree-of-Thoughts* (Cao et al., 2023). Methods to enhance task-specific performance include question rephrasing, subtask decomposition, verification, and symbolic grounding (Lyu et al., 2023; Xu et al., 2024; Wang et al., 2023a; Zelikman et al., 2022; Wang et al., 2024); factuality and faithfulness checking for reasoning chains (Wang et al., 2024); and separating knowledge retrieval from reasoning (Jin et al., 2024).

Iterative prompting techniques rely on predefined, hardcoded actions to guide reasoning, such as *Self-Refine* (Madaan et al., 2023), *IRCoT* (Trivedi et al., 2023), *iCAP* (Wang et al., 2022), *MetaGPT* (Hong et al., 2024), and *Chain of Ideas* (Anonymous, 2024b).

Memory-based methods include *Buffer of Thoughts* (Yang et al., 2024c), which distills high-level guidelines from previously solved tasks and stores them in a buffer for future reuse. Skill-based CoT (Didolkar et al., 2024) predicts skill-based labels for the questions. (Zhang et al., 2023b) identifies key concepts in questions and uses inductive prompting templates to extract related concepts.

rStar (Qi et al., 2024) employs a self-play mutual reasoning approach, augmented by Monte Carlo Tree Search (MCTS) with a set of five reasoning-inducing prompts, to enhance reasoning.

Finetuning-based methods, such as *STaR* (Zelikman et al., 2022), *ReST-MCTS* (Zhang et al., 2024),

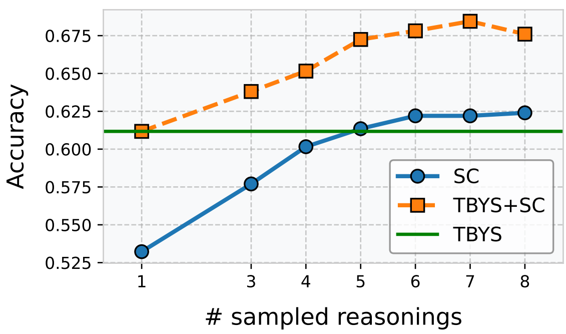

Figure 3: Performance comparison on MATH-500

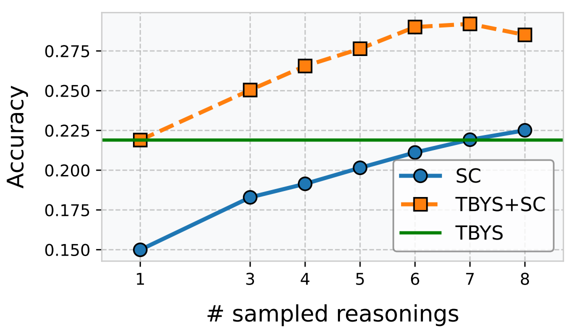

Figure 4: Performance comparison on AIME

and *AFlow* (Anonymous, 2024a), demonstrate that iterative training on reasoning histories and task-specific workflows of correct answers enables models to tackle increasingly complex problems.

5 Experiments

5.1 Experiment settings

We conducted experiments on two challenging mathematical datasets, *AIME* (Zhang et al., 2023a) and MATH-500 (Lightman et al., 2023). We compare TBYS against a simple yet very strong baseline: 8-shot *In-context Learning* (Lu et al., 2022) with *Self-Consistency* (Wang et al., 2023b).

For the experiments, use utilize the LLM *Qwen2.5-7B-Instruct* (Yang et al., 2024a) via the LLM API provided by Siliconflow (sil), with the following configurations: max_tokens=1024, temperature=0.2, top_k=40, top_p=0.7, and n=1. The *bge-large-en-v1.5* embedding model is employed for *insight* retrieval. Results are reported as the average across 8 experimental runs.

Since coding benefits mathematical problems (Chen et al., 2023), when Python code blocks are detected in the LLMs' responses, we invoke a customized sandboxed Python interpreter and append the output to the code block.

5.2 Comparison

When compared with *Self-Consistency* (SC), TBYS demonstrates comparable performance to SC using 5 reasoning samples (SC@5) on MATH-500 (Figure 3) and SC@7 on AIME (Figure 4). The results further indicate that TBYS integrates effectively with SC: TBYS+SC yields over 5% absolute gains in accuracy on MATH-500 and 7.5% on AIME.

5.3 Overhead Analysis

Table 1: Cost comparison to SC under similar accuracy.

MATH-500	Acc.	Time	Prompt	Completion
TBYS	0.61	52.82	18163.80	999.57
SC@5	0.61	102.56	13334.62	2217.30
AIME	**Acc.**	**Time**	**Prompt**	**Completion**
TBYS	0.22	78.15	20686.23	1559.60
SC@7	0.22	322.79	25,242.54	7,102.49

We compare the overhead of TBYS with SC@5 on MATH-500 and with SC@7 on AIME, where the methods achieve comparable accuracies. The metrics analyzed include wall-time, number of prompt tokens, and completion tokens. As shown in Table 1, under similar accuracies, TBYS reduces wall-time and the number of completion tokens by approximately half on MATH-500 and one-third on AIME. While TBYS uses 46% more prompt tokens on MATH-500, these can be cached and are typically much cheaper and faster to predict than completion tokens. Since completion token counts typically dominates runtime, our results show that completion token counts are consistently proportional to our runtime measurements across methods.

5.4 Ablation Study

Table 2: Ablation Study

	MATH-500	AIME
TBYS	61.17%	21.90%
- Library Construction	58.90%	19.51%
- Coding	57.00%	18.11%
8-shot	53.23%	14.99%

We conducted ablation experiments by using the raw insight library L_0 as L_1 (without filtering, as described in Section 3). Accuracy declines were observed in both datasets. Notably, we only performed one round of insight filtering (i.e., using $L = L_1$), and additional filtering rounds are expected to further improve accuracy. Table 2 also demonstrates that coding contributes half of the accuracy gain compared to simple 8-shot prompting.

5.5 Impact of Library Size

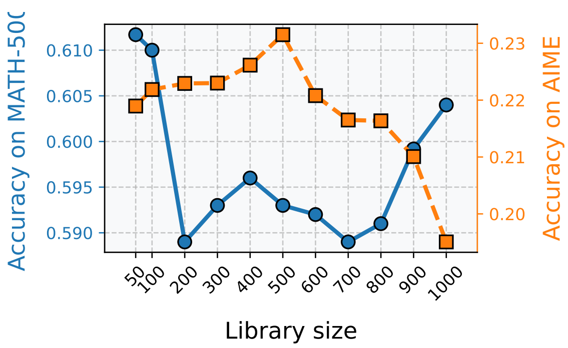

Figure 5: Impact of *insight* library size

In Section 3, we sorted the *insight* library L_0 and selected the top-k_L insights to form L_1. Figure 5 shows that on MATH-500, TBYS achieves peak accuracy with an insight library size of 50. On AIME, the optimal size is 500. Here, performance initially improves as library size decreases due to the filtering of lower-quality insights. However, as library size continue to decreases, excessively small libraries size reduces diversity in problem types and harms performance.

6 Additional Comparison Experiments

We compare with Skill-based CoT (Didolkar et al., 2024), a prompt-guided interaction procedure that enables LLMs to assign skill labels to math questions and perform ICL with label-specific exemplars. We also conducted experiments using a k-wait approach, where we append "Wait, " after the model completion and let the LLM to continue its generation for k times. Below are the comparison results.

Table 3: Comparison to Skill-based CoT and k-wait.

Method	Acc.
k-shot CoT	54.30%
Skill-based CoT	60.52%
k-wait (k=1)	55.00%
k-wait (k=2)	56.60%
k-wait (k=3)	54.20%
TBYS (Ours)	61.99%

Results in Figure 3 shown that TBYS is slight better than Skill-based CoT, which is task-specific, and is much better than k-wait.

7 Qualitative Analysis of Insight Quality

We provide qualitative analysis of the insights using two selected examples. These problems are relatively simple, but where k-shot reasoning fails. We use these examples to illustrate how TBYS's insights effectively steer its multi-step reasoning processes.

The first example in Figure 6 asks to convert $\frac{21}{2^2 \cdot 5^7}$ to a terminating decimal. TBYS solves the problem in two steps, with the goals of the insights being "ensure the denominator is a power of 10" and "Simplify the numerator and express the fraction as a terminating decimal".

The second example in Figure 7 asks to solve the question: $\sqrt{x+\sqrt{3x+6}}+\sqrt{x-\sqrt{3x+6}}=6$. TBYS solves the problem in two steps, with the, with the goals of the insights being "simplify the equation by squaring both sides to remove the square roots" and "find the value of x by substitution".

Both examples demonstrate that TBYS generates suitable insights for their respective problems.

8 Conclusion and Future Work

This paper introduces a novel proactive prompting paradigm, instantiates it with the simple TBYS reasoning framework, and verifies the effectiveness of TBYS on challenging advanced mathematics reasoning tasks.

Promising directions for future improvement include: Automated search for optimal *insight*s (Yang et al., 2024b); integration of long-term memory mechanisms (Tang et al.; Anonymous, 2025); enhancement of programming capabilities (Chen et al., 2023); enforcement of structured inference processes (Li et al., 2025b; Cao et al., 2023).

Limitations

Our method incurs higher computational overhead compared to direct prompting, a common drawback among advanced prompting techniques that involve scaling test-time inference.

Due to time and financial constraints (our current experiments take about 50 days with single-threaded API calls), we only evaluated the proposed method on two math-domain datasets using a single LLM.

Ethical Statement

This work fully adheres to the ACL Ethics Policy. To the best of our knowledge, no ethical issues are associated with this research.

References

https://siliconflow.cn/.

Anonymous. 2024a. AFlow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Anonymous. 2024b. Chain of ideas: Revolutionizing research in idea development with LLM agents. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Anonymous. 2025. Inference scaling for long-context retrieval augmented generation. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Shulin Cao, Jiajie Zhang, Jiaxin Shi, Xin Lv, Zijun Yao, Qi Tian, Lei Hou, and Juanzi Li. 2023. Probabilistic tree-of-thought reasoning for answering knowledge-intensive complex questions. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12541–12560, Singapore. Association for Computational Linguistics.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*.

Aniket Rajiv Didolkar, Anirudh Goyal, Nan Rosemary Ke, Siyuan Guo, Michal Valko, Timothy P Lillicrap, Danilo Jimenez Rezende, Yoshua Bengio, Michael Curtis Mozer, and Sanjeev Arora. 2024. Metacognitive capabilities of LLMs: An exploration in mathematical problem solving. In *AI for Math Workshop @ ICML 2024*.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Minguk Jang, Sae-Young Chung, and Hye Won Chung. 2023. Test-time adaptation via self-training with nearest neighbor information. *ICLR 2024*.

Mingyu Jin, Weidi Luo, Sitao Cheng, Xinyi Wang, Wenyue Hua, Ruixiang Tang, William Yang Wang, and Yongfeng Zhang. 2024. Disentangling memory and reasoning ability in large language models. *Preprint*, arXiv:2411.13504.

Chengpeng Li, Mingfeng Xue, Zhenru Zhang, Jiaxi Yang, Beichen Zhang, Xiang Wang, Bowen Yu, Binyuan Hui, Junyang Lin, and Dayiheng Liu. 2025a. Start: Self-taught reasoner with tools. *Preprint*, arXiv:2503.04625.

Zhuoqun Li, Xuanang Chen, Haiyang Yu, Hongyu Lin, Yaojie Lu, Qiaoyu Tang, Fei Huang, Xianpei Han, Le Sun, and Yongbin Li. 2025b. Structrag: Boosting knowledge intensive reasoning of llms via inference-time hybrid information structurization. In *International Conference on Learning Representations (ICLR)*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *ICLR 2024*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 305–329, Nusa Dua, Bali. Association for Computational Linguistics.

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

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *Preprint*, arXiv:2501.19393.

OpenAI. 2024. Learning to Reason with LLMs.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2024. Mutual reasoning makes smaller llms stronger problem-solvers. In *Arxiv*.

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

Xiangru Tang, Tianyu Hu, Muyang Ye, Yanjun Shao, Xunjian Yin, Siru Ouyang, Wangchunshu Zhou, Pan Lu, Zhuosheng Zhang, Yilun Zhao, et al. Chemagent: Self-updating library in large language models improves chemical reasoning. In *The Twelfth International Conference on Learning Representations*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Boshi Wang, Xiang Deng, and Huan Sun. 2022. Iteratively prompt pre-trained language models for chain of thought. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2714–2730, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jianing Wang, Qiushi Sun, Xiang Li, and Ming Gao. 2024. Boosting language models reasoning with chain-of-knowledge prompting. In *The 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4958–4981, Bangkok, Thailand. Association for Computational Linguistics.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim.

2023a. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

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*.

Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. 2024. Faithful logical reasoning via symbolic chain-of-thought. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13326–13365, Bangkok, Thailand. Association for Computational Linguistics.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2024b. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. 2024c. Buffer of thoughts: Thought-augmented reasoning with large language models. *arXiv preprint arXiv:2406.04271*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, volume 35, pages 15476–15488. Curran Associates, Inc.

Dan Zhang, Sining Zhoubian, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024. Rest-mcts*: Llm self-training via process reward guided tree search. Thirty-eighth Conference on Neural Information Processing Systems (NeurIPS).

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, Irwin King, Xue Liu, and Chen Ma. 2025. A survey on test-time scaling in large language models: What, how, where, and how well? *Preprint*, arXiv:2503.24235.

Xingyuan Zhang, Philip Becker-Ehmck, Patrick van der Smagt, and Maximilian Karl. 2023a. Action inference by maximising evidence: Zero-shot imitation from observation with world models. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*.

Zhebin Zhang, Xinyu Zhang, Yuanhang Ren, Saijiang Shi, Meng Han, Yongkang Wu, Ruofei Lai, and Zhao Cao. 2023b. IAG: Induction-augmented generation framework for answering reasoning questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1–14, Singapore. Association for Computational Linguistics.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

Jingming Zhuo, Songyang Zhang, Xinyu Fang, Haodong Duan, Dahua Lin, and Kai Chen. 2024. ProSA: Assessing and understanding the prompt sensitivity of LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1950–1976, Miami, Florida, USA. Association for Computational Linguistics.

A Figures for Qualitative Analysis of Insight Quality

- PROBLEM: Convert $\frac{21}{2^2 \cdot 5^7}$ to a terminating decimal.
- Insight For Reasoning Step 1:
  - Situation: We need to convert the fraction $\frac{21}{2^2 \cdot 5^7}$ to a terminating decimal.
  - Goal: To convert the fraction to a terminating decimal, we need to ensure the denominator is a power of 10. This can be achieved by adjusting the powers of 2 and 5 in the denominator to be the same.
- Reasoning Step 1: To convert the fraction $\frac{21}{2^2 \cdot 5^7}$ to a terminating decimal, we need to ensure the denominator is a power of 10. The denominator is currently $2^2 \cdot 5^7 = 4 \cdot 78125 = 312500$. To make it a power of 10, we need to multiply both the numerator and the denominator by $2^5 = 32$ because $2^2 \cdot 5^7 \cdot 2^5 = 2^7 \cdot 5^7 = 10^7$. This will make the denominator $10^7 = 10000000$.
- Insight For Reasoning Step 2:
  - Situation: After multiplying both the numerator and the denominator by 32, the fraction becomes $\frac{21 \cdot 32}{10^7}$.
  - Goal: Simplify the numerator and express the fraction as a terminating decimal.
- Reasoning Step 2: After multiplying both the numerator and the denominator by 32, the fraction becomes $\frac{21 \cdot 32}{10^7} = \frac{672}{10000000}$. Converting this to a decimal gives 0.0000672.

Figure 6: Example 1 for Qualitative Analysis of Insight Quality

- PROBLEM: Solve $\sqrt{x + \sqrt{3x+6}} + \sqrt{x - \sqrt{3x+6}} = 6$.
- Insight For Reasoning Step 1:
  - Situation: We are given the equation $\sqrt{x + \sqrt{3x+6}} + \sqrt{x - \sqrt{3x+6}} = 6$, which involves nested square roots and a sum of square roots. The presence of nested square roots complicates the equation and makes it difficult to isolate x directly.
  - Goal: To simplify the equation, we need to find a way to eliminate the square roots. One approach is to square both sides of the equation to remove the outer square roots, but we must be cautious of potential extraneous solutions that may arise from this process.
- Reasoning Step 1: To solve the equation $\sqrt{x + \sqrt{3x+6}} + \sqrt{x - \sqrt{3x+6}} = 6$, let's denote $a = \sqrt{x + \sqrt{3x+6}}$ and $b = \sqrt{x - \sqrt{3x+6}}$. We know that $a + b = 6$. Squaring both sides of the equation $a + b = 6$ gives us $a^2 + b^2 + 2ab = 36$. Also, since $a^2 = x + \sqrt{3x+6}$ and $b^2 = x - \sqrt{3x+6}$, we have $a^2 + b^2 = 2x$. Therefore, $2x + 2ab = 36$, which simplifies to $x + ab = 18$. Now, we need to find ab. Notice that $ab = \sqrt{(x + \sqrt{3x+6})(x - \sqrt{3x+6})} = \sqrt{x^2 - (3x+6)} = \sqrt{x^2 - 3x - 6}$. Thus, we have $x + \sqrt{x^2 - 3x - 6} = 18$. Let's denote $y = \sqrt{x^2 - 3x - 6}$, so the equation becomes $x + y = 18$ and $y^2 = x^2 - 3x - 6$. We can substitute $y = 18 - x$ into the second equation to find x.
- Insight For Reasoning Step 2:
  - Situation: We have the equations $x + y = 18$ and $y^2 = x^2 - 3x - 6$, where $y = \sqrt{x^2 - 3x - 6}$.
  - Goal: Substitute $y = 18 - x$ into the equation $y^2 = x^2 - 3x - 6$ to find the value of x.
- Reasoning Step 2: To find x, we substitute $y = 18 - x$ into the equation $y^2 = x^2 - 3x - 6$:

$$(18 - x)^2 = x^2 - 3x - 6.$$

Expanding the left side, we get:

$$324 - 36x + x^2 = x^2 - 3x - 6.$$

Simplifying by canceling x^2 from both sides, we get:

$$324 - 36x = -3x - 6.$$

Rearranging terms, we get:

$$324 + 6 = 36x - 3x,$$
$$330 = 33x,$$
$$x = 10.$$

We need to verify if $x = 10$ satisfies the original equation. Substituting $x = 10$ into the original equation:

$$\sqrt{10 + \sqrt{3 \cdot 10 + 6}} + \sqrt{10 - \sqrt{3 \cdot 10 + 6}} = \sqrt{10 + \sqrt{36}} + \sqrt{10 - \sqrt{36}} = \sqrt{10 + 6} + \sqrt{10 - 6} = \sqrt{16} + \sqrt{4} = 4 + 2 = 6.$$

Since the left side equals the right side, $x = 10$ is a valid solution.

Figure 7: Example 2 for Qualitative Analysis of Insight Quality