DDPrompt: Differential Diversity Prompting in Large Language Models

Lin Mu
Anhui University
Hefei, China
mulin@ahu.edu.cn

Wenhao Zhang
Anhui University
Hefei, China
zhangwenhao@stu.ahu.edu.cn

Yiwen Zhang*
Anhui University
Hefei, China
zhangyiwen@ahu.edu.cn

Peiquan Jin
University of Science and Technology of China
Hefei, China
jpq@ustc.edu.cn

Abstract

Large Language Models (LLMs) have shown that their reasoning ability could be enhanced through approaches like Chain-of-Thought (CoT) prompting. However, these methods use single prompts for different types of questions and do not design appropriate prompts for questions with different characteristics. In this paper, we aim to explore a methodology that generates differentially diverse reasoning paths for different types of questions. To achieve this, we propose a novel prompting strategy called Differential Diversity Prompting (**DDPrompt**). Firstly, we generate the optimal prompts collection based on question characteristics. Then, we use this optimal prompt collection to generate multiple answers for a question and choose the final answer by voting. We evaluated DDPrompt on twelve reasoning benchmarks and significant improvement in the performance of LLMs on complex reasoning tasks (e.g., GSM8K 75% $\rightarrow$ 84%, Tracking Shuffled Objects (68.8% $\rightarrow$ 83.9%)).

1 Introduction

Large Language Models (LLMs) (Brown et al., 2020; Chowdhery et al., 2023) have shown remarkable abilities by learning from demonstrations while keeping their parameters frozen, which is called prompting. The design of prompting is crucial as it can significantly impact the performance of the LLMs on complex reasoning tasks (Chu et al., 2023), such as arithmetic reasoning (Cobbe et al., 2021; Patel et al., 2021), commonsense reasoning (Geva et al., 2021; Talmor et al., 2019), symbolic reasoning (Wei et al., 2022; Srivastava et al., 2022).

Recent studies (Chu et al., 2023) have explored various prompting strategies. For instance, Chain-of-Thought (CoT) prompting (Wei et al., 2022) provided step-by-step reasoning examples to facilitate LLMs decomposing complex reasoning tasks into intermediate steps. However, this method required careful manual design of demonstrations, which is time-consuming and labor-intensive. Zero-Shot-CoT (Kojima et al., 2022) discovered that by adding a single trigger sentence, such as *"Let's think step by step"*, after the question to induce the LLMs in generating the reasoning paths, they could achieve competitive performance to standard CoT. Some research (Wang et al., 2023; Naik et al., 2023) have found that utilizing diverse prompts could effectively improve the reasoning ability of LLMs. For example, (Wang et al., 2023) introduced a self-consistency technique involving generating multiple reasoning paths using a decoding strategy different from standard CoT. (Naik et al., 2023) leveraged LLMs to automatically generate diverse prompts, which were then ensemble across multiple inference calls for each question.

In this paper, we aim to improve the performance of the LLMs by designing a prompting strategy that decreases manual labor and increases the diversity of prompts. One method we were considering is to utilize diversity trigger sentences, such as *"Let's think step by step", "Let's think about this logically"* mentioned in Zero-Shot-CoT, to facilitate LLMs generate diversity reasoning paths for each question. However, this naive approach is inefficient. As per human experience, choosing the appropriate methods for a question based on its characteristics is crucial. Inspired by this mind, we assume that different trigger sentences have varying effects on different types of questions. We choose appropriate trigger sentences to generate reasoning paths for a question. As shown in Figure 1, we conducted a preliminary experiment on the GSM8K (Cobbe et al., 2021). We noticed that the accuracy of different trigger sentences varied across different clusters. Therefore, we explore an approach to generate differentially diverse reasoning paths for different

*Corresponding author

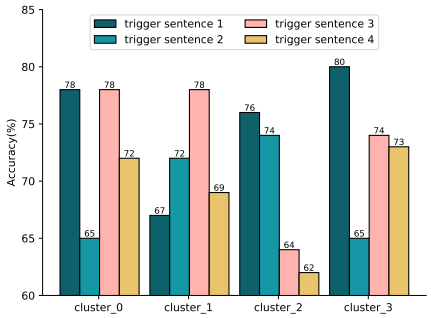

Figure 1: Accuracy(%) of different trigger sentences for different clusters. We partition questions in GSM8K (Cobbe et al., 2021) into several clusters based on their similarity and use the method proposed in (Kojima et al., 2022) to verify the accuracy of four trigger sentences on questions in different clusters.

types of questions. To achieve this, we propose a novel prompting strategy called the **D**ifferential **D**iversity **Prompt**ing **(DDPrompt)**. This approach involves two stages. In the first stage, we generate an optimal trigger sentence set for each type of question. In the second stage, we utilize the optimal trigger sentence set to obtain the final answer for a question. By using this approach, we can provide differentially diverse reasoning paths for different types of questions and ensure an analysis from various perspectives.

We evaluate DDPrompt on twelve reasoning benchmarks from four categories of reasoning tasks, including arithmetic, commonsense, symbolic, and logical reasoning tasks. The result shows that DDPrompt could significantly improve the performance of LLMs compared to Zero-Shot-CoT. For instance, GSM8K (75% $\rightarrow$ 84%), AQUA-RAT (50% $\rightarrow$ 63%), Last Latter (64% $\rightarrow$ 89.8%), Tracking Shuffled Objects (68.8% $\rightarrow$ 83.9%).

2 Method

In this section, we provide a detailed explanation of the techniques used in DDPrompt. This method is distinct from the Zero-Shot-Cot (Kojima et al., 2022), which uses a uniform trigger for different questions, e.g., *Let's think step by step*. Figure 2 shows the difference between Zero-Shot-Cot and DDPrompt. DDPrompt involves two stages: **G**enerating **O**ptimal **T**rigger **S**entence **S**et(**GOTSS**) and **Inference**.

2.1 GOTSS

In this section, we introduce how to generate the optimal trigger sentence set for different types of questions, which consist of two parts: (1) **Question clustering**, We partition the questions into a small number of clusters based on their similarity; (2) **Generating Optimal collection**. An optimal trigger sentence set is generated for each cluster by verifying the validity of different trigger sentences.

2.1.1 Question clustering

To classify the questions into different types, we first cluster the questions based on their similarity. Give a question collection $\boldsymbol{Q}$. We obtain an embedding for each question $q \in \boldsymbol{Q}$ using Sentence-BERT (Reimers and Gurevych, 2019). Then, the question embeddings are fed into the K-Means clustering. Finally, we get a collection of clusters $\boldsymbol{C} = \{c_1, c_2, ..., c_m\}$, where each cluster $c_i \in \boldsymbol{C}$ contain several questions of the same type and m is the number of the cluster in $\boldsymbol{C}$.

2.1.2 Generating Optimal Collection

Since different trigger sentences may perform differently depending on the type of question. For each cluster, we select a few best-performing trigger sentences to form the optimal trigger sentence set. Specifically, followed (Kojima et al., 2022), we first manually constructed a set of different trigger sentences $\boldsymbol{T} = \{t_1, t_2, ..., t_n\}$, where n is the number of trigger sentence in $\boldsymbol{T}$. Second, we verify the effectiveness of n trigger sentences separately for each cluster $c_i \in \boldsymbol{C}$ obtained in the previous parts. For each $t \in \boldsymbol{T}$ and each $q \in c_i$, we input $[q, t]$ into (Kojima et al., 2022) to obtain an answer a. We then compare a to the ground truth to determine the accuracy of t for c_i. After that, we get the accuracy of n trigger sentences for c_i. We then choose the highest accuracy k trigger sentences to form the optimal trigger sentence set for c_i, where $k < n$. We perform the above operation for each $c \in \boldsymbol{C}$, and finally get a collection of optimal trigger sentence set $\boldsymbol{D} = \{d_1, d_2, ..., d_m\}$, where d_i is the optimal trigger sentence set for c_i.

During the GOTSS phase, we cluster the training dataset and then randomly select a subset of samples to generate the optimal trigger sentence set. In the case where only the test dataset is available, we randomly partition the test dataset into a training dataset and a test dataset, and then apply the same procedure to the specified training dataset.

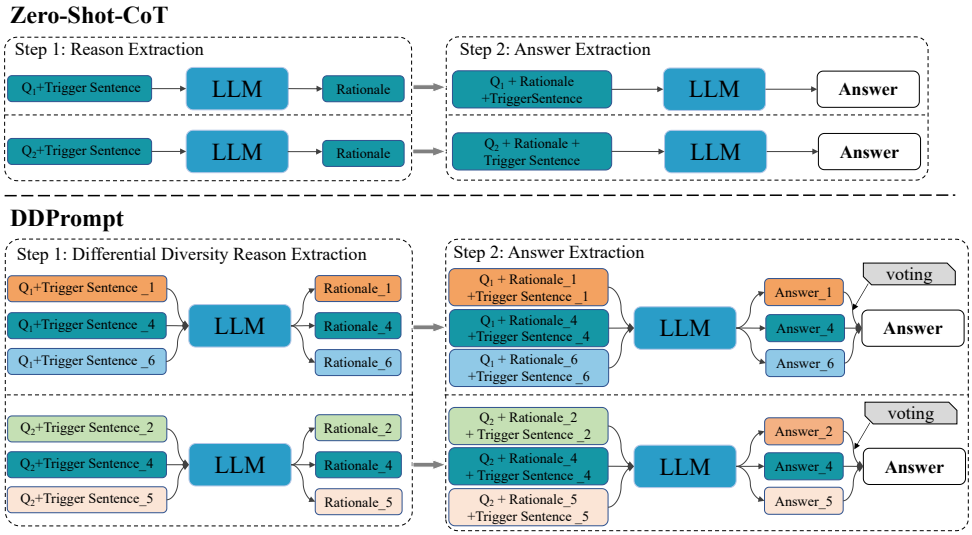

Figure 2: Comparison of Zero-Shot-CoT and DDPrompt. Notice that both have two different types of questions: Q_1 and Q_2. Zero-Shot-CoT uses a single trigger, e.g., *Let's think step by step*. However, DDPrompt uses an optimal trigger sentence set depending on the type of question.

2.2 Inference

In the previous stage, we generated an optimal trigger sentence set for each cluster. In this stage, we leverage these optimal trigger sentence sets to infer the answer to the question. As shown in Figure 2. First, give a question q, we obtain embedding of q using Sentence-BERT (Reimers and Gurevych, 2019). Then, we identify the cluster that is most similar to q by computing the cosine similarity between q and each cluster $c_i \in \boldsymbol{C}$. Subsequently, we select c_i that is most similar to q and retrieve the optimal trigger sentence set $d_i = \{t_1, t_2, ..., t_k\}$ for c_i. For each $t \in d_i$, we input $[q, t]$ into (Kojima et al., 2022) to obtain an answer a. Finally, we get k answers for q and the final answer is determined by utilizing majority voting.

3 Experiments

3.1 Tasks and Datasets

In the experiment, we evaluate DDPrompt on twelve benchmarks from four categories of reasoning tasks: (1) Arithmetic (SingleEq (Koncel-Kedziorski et al., 2015), AddSub (Hosseini et al., 2014), MultiArith (Roy and Roth, 2015), AQUA-RAT (Ling et al., 2017), GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021)); (2) Commonsense (CSQA (Talmor et al., 2019), StrategyQA (Geva et al., 2021)); (3) Symbolic (Last Letter Concatenation, Coin Flip) (Wei et al., 2022); (4) Logical (Date Understanding, Tracking Shuffled Objects) (Srivastava et al., 2022).

3.2 Baselines

We compare DDPrompt to four baselines: Zero-Shot (Kojima et al., 2022), Zero-Shot-CoT (Kojima et al., 2022), Few-Shot (Wei et al., 2022), and Few-Shot-CoT (Wei et al., 2022). Zero-Shot and Zero-Shot-CoT utilize the same trigger sentence as stated in (Kojima et al., 2022). Few-Shot and Few-Shot-CoT use the same demonstration examples as stated in (Wei et al., 2022)

In the experiment, we use the GPT3.5-turbo from OpenAI[1] as LLM. We manually constructed $n = 14$ different trigger sentences and set $k = 5$.

3.3 Result

The accuracy of DDPrompt is compared with different baseline methods for twelve reasoning datasets in Table 1. DDPrompt shows significant improvement in performing reasoning tasks as compared to Zero-Shot-CoT. For instance, GSM8K (75% $\rightarrow$ 84%), AQUA-RAT (50% $\rightarrow$ 63%), Last Latter (64% $\rightarrow$ 89.8%), Tracking Shuffled Objects (68.8% $\rightarrow$ 83.9%). DDPrompt outperforms eight out of twelve reasoning tasks (SingleEq, MultiArith, AQUA-RAT, GSM8K, SVAMP, CSQA, Last Letter Concatenation, Tracking Shuffled Objects) compared to Few-Shot-CoT that has manual design rationales. Additionally, for the arithmetic reasoning tasks, AQUA-RAT, GSM8K, and SVAMP datasets involve multi-step reasoning, which is more complex than other arithmetic datasets (Chu

[1]https://openai.com/

Method	Arithmetic					
	SingleEq	**AddSub**	**MultiArith**	**AQUA-RAT**	**GSM8K**	**SVAMP**
Zero-Shot	80.1	78.7	59.7	25.6	13.0	61.4
Zero-Shot-CoT	90.6	79.2	96.2	50.0	75	78.1
Few-ShoT	86.8	86.1	81.5	44.9	42.5	76.6
Few-Shot-CoT	90.2	**87.1**	97.0	53.9	72.3	79.6
DDPrompt	**92.5**$_{(+1.9)}$	86.9$_{(+7.7)}$	**98.7**$_{(+2.5)}$	**63**$_{(+13)}$	**84.0**$_{(+9)}$	**83.6**$_{(+5.5)}$
Method	**Commonsense**		**Symbolic**		**Logical**	
	CSQA	Strategy	Coin Flip	Last Letter	Date	Tracking
Zero-Shot	71.6	63.6	51.0	1.4	41.7	34.5
Zero-Shot-CoT	68.5	62.4	92.2	71.6	64.0	68.8
Few-Shot	70.4	43.1	50.2	6.6	52.3	30.9
Few-Shot-CoT	58.1	**65.6**	**99.8**	71.6	**72.6**	75.0
DDPrompt	**74.5**$_{(+6)}$	64.6$_{(+2.2)}$	95.2$_{(+3)}$	**89.8**$_{(+18.2)}$	72.1$_{(+8.1)}$	**83.9**$_{(+15.1)}$

Table 1: Accuracy(%) of twelve reasoning tasks. (*) indicate the improvement of DDPrompt compared to Zero-Shot-CoT.

et al., 2023). DDPrompt has proved to be more effective in improving performance on these complex datasets. It indicates that DDPrompt is better suited for solving intricate and challenging problems.

In the datasets used in this paper, the first three arithmetic datasets, i.e. SingleEq, AddSub, and MultiArith, contain relatively simple problem(Chu et al., 2023). Consequently, commendable results can be attained without necessitating multi-perspective analysis, as shown in Table1. For these datasets, using Zero-Shot-CoT and Few-Shot-CoT produces satisfactory results, reducing the distinctiveness of our approach's advantage. However, for the last three arithmetic datasets, especially AQUA-RAT and GSM8K, which contain more intricate problems(Chu et al., 2023), addressing these intricate problems requires generating multiple reasoning paths for solving the problem from diverse perspectives(Wang et al., 2023). This significantly improves the performance of our method on more complex arithmetic problems.

3.4 Ablation study

To evaluate the effectiveness of two design components of DDPrompt, which are called **Random-K** and **Single**: In the Random-K variation, K trigger sentences are randomly selected and compared with the K trigger sentences that have the highest accuracy to evaluate the effectiveness of Top-K. In the Single variation, only the Top 1 trigger sentence is selected as a contrast experiment to evaluate the effectiveness of diversity. We conduct an ablation study by removing each component one at a time. Figure 3 shows an ablation study results with two variations of DDPrompt. We can conclude that both design components are effective and essential.

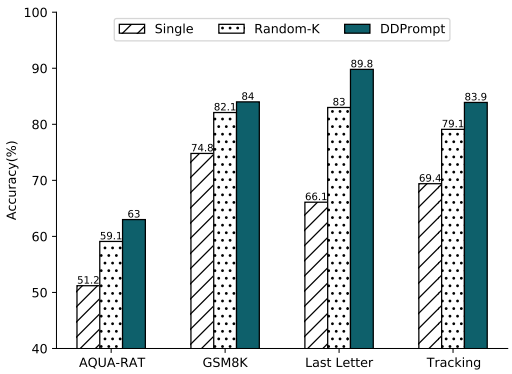

Figure 3: Ablation Studies of Design Components.

4 Related Works

Chain-of-Thought (CoT) (Wei et al., 2022) generated intermediate thought steps for problem-solving and significantly improved the reasoning ability of LLMs. Different from the CoT approach, least-to-most (Zhou et al., 2023) suggested solving complex problems by decomposing them into a series of simpler subproblems. These methods are tedious to manually construct the appropriate rationales for the different questions in the demonstration. Zero-Shot-CoT (Kojima et al., 2022) in-

volved adding a simple trigger sentence like "*Let's think step by step*" after the question to facilitate LLMs generating a step-by-step reasoning path. Auto-CoT (Zhang et al., 2022) proposed selecting demonstrations from different cluster methods and exploiting the benefits of diversity in demonstrations. (Wang et al., 2023) proposed a self-consistency method that replaced the greedy decoding method used in CoT with a temperature sample to obtain a set of diverse reasoning paths. Li et al. (Li et al., 2023) proposed sampling from varying prompts and then employed a verifier to verify the quality of each reasoning path.

5 Conclusion

In this paper, we introduce DDPrompt, which is designed to generate differentially diverse reasoning paths for different types of questions. DDPrompt consists of two stages: the GOTSS stage, which generates the optimal trigger sentence set for each type of question; the Inference stage, which uses this optimal trigger sentence set to generate multiple answers for a question and choose the final answer by majority voting. We evaluated DDPrompt's performance on twelve reasoning benchmarks and observed a significant improvement in the performance of LLMs.

6 Limitations

Our proposed DDPrompt is capable of generating differentially diverse reasoning paths for different types of questions. The inference stage requires multiple trigger sentences and questions to be fed into the LLM to generate multiple answers. As a result, our method is much slower in reasoning than other prompting methods. Additionally, another limitation is that we tested DDPrompt using only GPT3.5-turbo and have not yet evaluated it on other LLMs. Therefore, we will evaluate DDPrompt on other LLMs in the future.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No.62206004, No.62272001, No.62072419, No.62106004) and Hefei Key Common Technology Project (GJ2022GX15).

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. 2023. A survey of chain of thought reasoning: Advances, frontiers and future. *arXiv preprint arXiv:2309.15402*.

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

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.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, Doha, Qatar. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*

(Volume 1: Long Papers), pages 5315–5333, Toronto, Canada. Association for Computational Linguistics.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

Ranjita Naik, Varun Chandrasekaran, Mert Yuksekgonul, Hamid Palangi, and Besmira Nushi. 2023. Diversity of thought improves reasoning abilities of large language models. *arXiv preprint arXiv:2310.07088*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. 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 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*, volume 35, pages 24824–24837. Curran Associates, Inc.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.

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

A Appendix

A.1 Compared to other methods

DDPrompt automatically generates diverse reasoning paths for different types of problems, and clustering techniques constitute a component of our methodology. It is similar to self-consistency(SC)(Wang et al., 2023) and Auto-CoT(Zhang et al., 2022). Furthermore, we conducted complementary experiments on SC and Auto-CoT employing GPT3.5-turbo on the AQUA-RAT and GSM8K datasets. Notably, SC is a few-shot method that requires manually constructing reasoning paths. For a fair comparison, we compare DDPrompt with zero-shot setting SC. The results are presented in Table 2, and it reveals that DDPrompt exhibits superior performance compared to SC and Auto-CoT across both the AQUA-RAT and GSM8K datasets.

Method	**GSM8K**	**AQUA-RAT**
Auto-CoT	76.7	55.9
SC	82.1	61
DDPrompt	**84.0**	**63.0**

Table 2: DDPrompt compared to other methods.

A.2 All trigger sentences

Table3 shows all trigger sentences used in this paper.

No.	Trigger Sentences
1	Let's think step by step.
2	We should think about this step by step.
3	First,
4	Before we dive into the answer,
5	Proof followed by the answer.
6	Let's think step by step in a realistic way.
7	Let's think step by step using common sense and knowledge.
8	Let's think like a detective step by step.
9	Let's think about this logically.
10	Let's think step by step. First,
11	Let's think
12	Let's solve this problem by splitting it into steps.
13	The answer is after the proof.
14	Let's be realistic and think step by step.

Table 3: All trigger sentences used in this paper.