RankPrompt: Step-by-Step Comparisons Make Language Models Better Reasoners

Chi Hu[1,2*], **Yuan Ge**[1], **Xiangnan Ma**[1], **Hang Cao**[1],
Qiang Li[2], **Yonghua Yang**[2], **Tong Xiao**[1,3], **Jingbo Zhu**[1,3†]
[1]School of Computer Science and Engineering,
Northeastern University, Shenyang, China
[2]Alibaba Group
[3]NiuTrans Research, Shenyang, China
{huchinlp, geyuanqaq}@gmail.com
{xiaotong, zhujingbo}@mail.neu.edu.cn
{lq178896, huazai.yyh}@taobao.com

Abstract

Large Language Models (LLMs) have achieved impressive performance across various reasoning tasks. However, even state-of-the-art LLMs such as ChatGPT are prone to logical errors during their reasoning processes. Existing solutions, such as deploying task-specific verifiers or voting over multiple reasoning paths, either require extensive human annotations or fail in scenarios with inconsistent responses. To address these challenges, we introduce RankPrompt, a new prompting method that enables LLMs to self-rank their responses without additional resources. RankPrompt breaks down the ranking problem into a series of comparisons among diverse responses, leveraging the inherent capabilities of LLMs to generate chains of comparison as contextual exemplars. Our experiments across 11 arithmetic and commonsense reasoning tasks show that RankPrompt significantly enhances the reasoning performance of ChatGPT and GPT-4, with improvements of up to 13%. Moreover, RankPrompt excels in LLM-based automatic evaluations for open-ended tasks, aligning with human judgments 74% of the time in the AlpacaEval dataset. It also exhibits robustness to variations in response order and consistency. Collectively, our results validate RankPrompt as an effective method for eliciting high-quality feedback from language models.

Keywords: Language Modeling, Reasoning, Model Feedback

1. Introduction

Reasoning ability is a fundamental aspect of human intelligence, crucial for tasks such as mathematical problem-solving (Koncel-Kedziorski et al., 2016; Ling et al., 2017) and questions-answering (Talmor et al., 2019; Geva et al., 2021). Recent advancements show that Large Language Models (LLMs) (Brown et al., 2020; Thoppilan et al., 2022; Chowdhery et al., 2023; Ouyang et al., 2022) can demonstrate remarkable reasoning abilities when guided by Chain-of-Thought (CoT) prompting (Wei et al., 2022b; Kojima et al., 2022). This technique provides LLMs with prompts, such as "*Let's think step by step*", to facilitate the generation of a sequence of intermediate steps before arriving at the final result. CoT prompting has yielded impressive performance across a variety of tasks, including arithmetic, commonsense, and symbolic reasoning (Wei et al., 2022a; Zhang et al., 2023; Suzgun et al., 2023; Zhou et al., 2023a).

Despite their success, LLMs often make logical mistakes during the reasoning process (Kojima et al., 2022; Turpin et al., 2023; Lightman et al., 2023). As shown in Table 1, when solving algebra problems, a language model may provide wrong inferences or omit pivotal steps, leading to incorrect final results. One potential solution is to use task-specific verifiers to validate each step (Cobbe et al., 2021; Li et al., 2023; Lightman et al., 2023).

Question: Bob spends $27000 distributed between 3 schools to buy books. He can buy 100 books for $500. How many books can he buy per school?
Answer 1: Bob spends a total of $27000 on books. Each book costs $500. Therefore, he can buy $27000 / $500 = 54 books in total. Since he is distributing the books between 3 schools, he can buy 54 / 3 = 18 books per school. ✘
Answer 2: Bob can buy 27000 / 500 = 54 sets of 100 books. So, Bob can buy 54 x 100 = 5400 books. Therefore, he can buy 5400 / 3 = 1800 books per school. Answer: 1800. ✔
Answer 3: Bob can buy 100 books for $500, so the cost of one book is $500 / 100 = $5. Bob spends $27000, so he can buy 27000 / 5 = 5400 books. Therefore, Bob can buy 5400 books per school. ✘

Table 1: An example from GSM8K (Cobbe et al., 2021). Answer 2 is correct, while others make invalid inferences or miss steps in their reasoning process (marked in red). In this case, there is no major answer among all candidates.

*Work done while the author was interning at Alibaba.
†Corresponding author.

LREC-COLING 2024, pages 13524–13536
20-25 May, 2024. © 2024 ELRA Language Resource Association: CC BY-NC 4.0

However, it requires substantial labeled data for training, which is costly and time-consuming. An alternative is to sample a variety of reasoning paths and aggregate the results via majority voting (Wang et al., 2023d; Fu et al., 2023b). This method can alleviate the impact of individual errors and lead to more accurate predictions (Huang and Chang, 2023; Huang et al., 2022). Nevertheless, this aggregate voting strategy ignores intermediate steps, lacks interpretability, and struggles with inconsistent answers, as illustrated in Table 1. Therefore, it is crucial to develop a robust, interpretable technique that can effectively distinguish among multiple reasoning paths, thereby augmenting the reasoning capabilities of LLMs.

In response to these challenges, we introduce RankPrompt, a novel prompting method for LLM-based reasoning. Unlike previous methods, RankPrompt generates diverse reasoning paths and instructs LLMs to select the optimal one. As illustrated in Figure 1, RankPrompt diverges from the well-established Direct Scoring method (Zheng et al., 2023), which assesses candidates individually. Instead, our approach directs LLMs to perform a comparative evaluation of candidates through two essential components: *step-aware comparison instructions* and *comparison exemplars*. The former decomposes the ranking problem into a series of comparisons, using instructions such as "Let's compare the answers step by step". The latter component, comparison exemplars, leverages the few-shot learning capabilities of LLMs to improve ranking performance further. In contrast to previous methods requiring manual design of exemplars (Wei et al., 2022b; Wang et al., 2023d), our approach tasks LLMs with generating multiple chains of comparisons and selecting the chains yielding correct ranking results as exemplars. These exemplars guide LLMs to systematically compare different paths, thereby reducing the requirement for labeled data and minimizing human intervention.

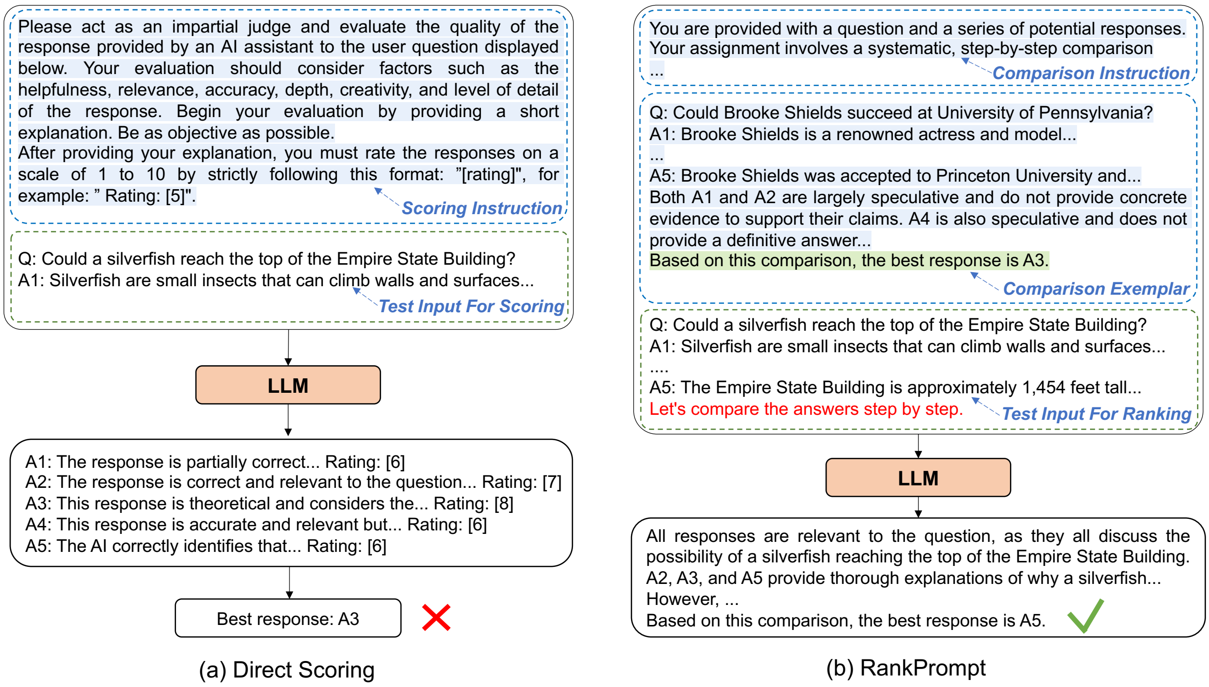
We evaluate RankPrompt across various arithmetic, commonsense, and symbolic reasoning benchmarks using ChatGPT. Empirical results demonstrate that RankPrompt consistently outperforms CoT prompting, achieving an improvement of up to 13% on the AQUA-RAT (Ling et al., 2017) data. On more challenging tasks from BIG-Bench-Hard (Suzgun et al., 2023), RankPrompt boosts the performance of GPT-4 (OpenAI, 2023a), with gains ranging from 5.2% to 9.2%. While our primary focus is on reasoning tasks, RankPrompt also excels in assessing open-ended generation. Specifically, it sets a new standard for LLM-based automatic evaluation by achieving a 74% agreement rate with human judgment on the AlpacaEval set. Remarkably, these impressive results can be obtained using a single exemplar, which underscores the efficacy of RankPrompt. Our analysis demonstrates that RankPrompt is robust to the order of candidate answers. Overall, our findings highlight the importance of considering intermediate steps in ranking tasks and establish RankPrompt as a promising approach for improving LLM-based reasoning.

2. Related Work

There is a surge in research interest in the field of LLMs due to their exceptional performance across a wide array of tasks (Brown et al., 2020; Thoppilan et al., 2022; Chowdhery et al., 2023; Hoffmann et al., 2022; OpenAI, 2023b). A key aspect of LLMs is their emergent abilities when provided with appropriate context (Wei et al., 2022a; OpenAI, 2023b; Zhao et al., 2023), leading to their potential use in reasoning and automatic evaluation. Here, we briefly discuss related work in the two fields.

LLMs as Reasoners. Reasoning with Language Models (LLMs) has become a popular research topic. One promising methodology is Chain-of-Thought (CoT) prompting, which encourages LLMs to generate a chain of reasoning steps (called a reasoning path) before delivering a final answer. This approach has been shown to improve the performance of LLMs across various tasks. CoT prompting optimization generally falls into two categories. The first focuses on enhancing the quality of individual reasoning paths through prompt engineering. For example, Kojima et al. (2022) find that specific trigger words can significantly improve the zero-shot reasoning performance of LLMs. Meanwhile, Fu et al. (2023b) demonstrate that incorporating complex exemplars into prompts can notably enhance the few-shot reasoning capabilities of LLMs. However, these methods often necessitate careful design and manipulation of prompts. The second category involves generating multiple reasoning paths and applying specific strategies to select the most effective one. For example, Wang et al. (2023d) use majority voting to select the final results, while Li et al. (2023) and Lightman et al. (2023) train step-aware verifiers to validate reasoning steps. Nonetheless, these methods also face challenges. Majority voting lacks interpretability and is prone to inconsistent final answers, while training verifiers requires a significant amount of labeled data. Our method addresses these limitations while complementing existing strategies for improving the quality of individual reasoning paths.

LLMs as Evaluators. Recent studies have explored the potential of LLMs in evaluating and refining their outputs. For instance, Liu et al. (2023) and Wang et al. (2023b) utilize LLMs to assess the quality of text generation tasks such as summarization



Please act as an impartial judge and evaluate the quality of the response provided by an AI assistant to the user question displayed below. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail of the response. Begin your evaluation by providing a short explanation. Be as objective as possible.
After providing your explanation, you must rate the responses on a scale of 1 to 10 by strictly following this format: "[rating]", for example: " Rating: [5]".

Q: Could a silverfish reach the top of the Empire State Building?
A1: Silverfish are small insects that can climb walls and surfaces...

A1: The response is partially correct... Rating: [6]
A2: The response is correct and relevant to the question... Rating: [7]
A3: This response is theoretical and considers the... Rating: [8]
A4: This response is accurate and relevant but... Rating: [6]
A5: The AI correctly identifies that... Rating: [6]

Best response: A3 ✗

You are provided with a question and a series of potential responses. Your assignment involves a systematic, step-by-step comparison
...

Q: Could Brooke Shields succeed at University of Pennsylvania?
A1: Brooke Shields is a renowned actress and model...
...
A5: Brooke Shields was accepted to Princeton University and...
Both A1 and A2 are largely speculative and do not provide concrete evidence to support their claims. A4 is also speculative and does not provide a definitive answer...
Based on this comparison, the best response is A3.

Q: Could a silverfish reach the top of the Empire State Building?
A1: Silverfish are small insects that can climb walls and surfaces...
....
A5: The Empire State Building is approximately 1,454 feet tall...
Let's compare the answers step by step.

All responses are relevant to the question, as they all discuss the possibility of a silverfish reaching the top of the Empire State Building. A2, A3, and A5 provide thorough explanations of why a silverfish...
However, ...
Based on this comparison, the best response is A5. ✓

(a) Direct Scoring

(b) RankPrompt

Figure 1: An overview of Direct Scoring (Zheng et al., 2023) (left) and RankPrompt (right). Direct Scoring independently assigns scores to each candidate, whereas RankPrompt ranks candidates through a systematic, step-by-step comparative evaluation. We present the detailed instructions for comparison in Table 2 and describe the construction of comparison exemplars in Section 3.2.2.

and machine translation. Similarly, Madaan et al. (2023) use LLMs to iteratively refine outputs for more complex tasks, such as acronym generation and code optimization. Dubois et al. (2023) and Zheng et al. (2023) show that, when equipped with carefully designed prompts, GPT-4 exhibits a high correlation with human preferences in judging the quality of open-ended text generation. It is established that LLM-based evaluators are cost-effective and efficient alternatives to crowd annotators (Fu et al., 2023a; Liu et al., 2023; Dubois et al., 2023; Zheng et al., 2023). However, the challenge lies in designing effective prompts to elicit the ranking ability of LLMs, often requiring significant human effort and extensive interactions with LLMs (Liu et al., 2023; Wang et al., 2023c,b). In this paper, we extend this line of research by developing a method that leverages LLMs to automatically generate exemplars for ranking, significantly reducing the need for human intervention. Our study also contributes to understanding how LLMs can be effectively utilized for reasoning and automatic evaluation tasks.

3. Method

This section introduces RankPrompt, a two-stage prompting framework for reasoning tasks. In the first stage, we generate multiple diverse reasoning paths, each potentially leading to a unique outcome. Our focus primarily lies in the second stage, where we re-rank these reasoning paths by comparing their steps and selecting the optimal one as the final answer.

3.1. Candidate Generation

The generation and aggregation of multiple reasoning paths have been proven to boost the performance of reasoning models (Wang et al., 2023d; Fu et al., 2023b). This process is similar to ensemble learning, a well-established machine learning method that combines the outputs of multiple models to improve overall accuracy and robustness against individual errors (Dietterich, 2000).

Given a question q, we generate n reasoning paths $\mathbf{p} = (p_1, p_2, \ldots, p_n)$, each potentially leading to a different final answer. We use few-shot CoT prompting (Wei et al., 2022b; Wang et al., 2023d) to generate these reasoning paths and apply temperature sampling (Ficler and Goldberg, 2017; Fan et al., 2018) to encourage diversity among the generated paths. Each reasoning path p_i (where $i \in 1, \ldots, n$) corresponds to a set of final answers $\mathbf{r} = (r_1, r_2, \ldots, r_n)$. We refer to the pairs (p_i, r_i), where each reasoning path p_i corresponds to a final answer r_i, as the *candidates* for question q. Hence, the candidate generation process results in a set of candidates $C_q = \{(p_1, r_1), (p_2, r_2), \ldots, (p_n, r_n)\}$ for each question q. We then use the candidate set C_q as the input for the subsequent ranking process.

[Comparison Instruction]
You are provided with a question and a series of potential responses.
Your assignment involves a systematic, step-by-step comparison of the reasoning paths embedded within each response.
This entails a thorough evaluation of each step's correctness and logical consistency.
After completing this all-encompassing assessment, rank the responses in accordance with the soundness of their respective reasoning paths.
Finally, select the best response and present it on a separate line as the optimal solution.
[Comparison Example]
{Comparison Exemplars}
[Question]
{Question}
[Candidate Answers]
{Candidates}
[Comparison]
Let's compare the answers step by step.

Table 2: The ranking template of RankPrompt. It instructs LLMs to compare candidate answers step by step and output in a specific format (marked in red).

3.2. Candidate Ranking

3.2.1. Comparative Evaluation of Reasoning Steps

A common approach to candidate ranking is to evaluate each candidate individually (Zheng et al., 2023; Wang et al., 2023c), a strategy we refer to as **Direct Scoring** (Figure 1(a)). However, such an approach often fails to account for the relative quality of different reasoning paths. For instance, LLMs such as ChatGPT often assign identical scores to candidates with similar reasoning steps, regardless of their differing outcomes (Dubois et al., 2023; Zheng et al., 2023).

To address this limitation, we introduce a comparative evaluation method, which concatenates all candidate reasoning paths with the original question to form the ranking input. This input is then processed by a ranking model, such as ChatGPT, guided by a step-aware comparison instruction. As presented in Table 2, the comparison instruction directs the model to execute a sequential comparison process before giving the conclusion. It also clarifies the required output format.

However, relying solely on comparison instructions, which we refer to as **Zero Ranking**, does not fully leverage the in-context learning capabilities of LLMs (Brown et al., 2020; Wei et al., 2022b). The Zero Ranking method can sometimes lead to irrelevant outputs, failure to adhere to the desired output format, or only a partial consideration of candidates (Sun et al., 2023; Qin et al., 2023b). To address these issues, we enhance the ranking capabilities of LLMs by incorporating comparison exemplars, as shown in Figure 1(b).

Algorithm 1 Creation of Comparison Exemplars

```
Require: Labeled data set $D = \{(q_1, a_1), \dots, (q_k, a_k)\}$,
    where $q_i$ is a question and $a_i$ is the correct answer,
    empty exemplar set $E$
Ensure: Comparison exemplar set $E = (e_1, \dots, e_k)$
procedure CreateExemplars($D$)
    for each data point $(q_j, a_j)$ in $D$ do
        Generate a diverse candidate set $C_{q_j}$ for $q_j$
        Initialize $e_j$ as an empty exemplar
        while $e_j$ has not been created for $q_j$ do
            Generate a comparison chain $c_j$ using Zero Ranking with $(q_j, C_{q_j})$
            if $c_j$ meets selection criteria then
                Append $e_j = (q_j, C_{q_j}, c_j)$ to $E$
                break
    return $E$
```

3.2.2. Construction of Comparison Exemplars

To fully exploit the in-context learning capabilities of Language Model Machines (LLMs), we enhance the instructions with high-quality examples. However, creating such examples can be a challenging and time-consuming task (Lu et al., 2022; Liu et al., 2022; Fu et al., 2023b). To address this issue, we propose an automatic method for generating comparison examples, as shown in Algorithm 1.

Algorithm 1 initiates by iterating through a labeled dataset D, creating a candidate set C_{q_j} for every question q_j. It then continuously produces comparison chains using Zero Ranking until it identifies a chain that meets the selection criteria. Echoing the approach of Zelikman et al. (2022), we select the comparison chain that accurately leads to the answer a_j. This chosen chain, along with the question and its candidate set, forms an exemplar e_j, which is subsequently added to the exemplar collection E. This procedure is repeated for each question until a suitable chain is found. Compared to previous methods, our approach requires only a minimal amount of labeled data for each task. In Section 5, we delve into the effects of exemplar selection on the efficacy of the ranking process.

4. Experiment

4.1. Experimental Setups

Models. We evaluate our method using state-of-the-art LLMs, including `gpt-3.5-turbo` and

Method	**Arithmetic**				**Commonsense**			**Symbolic**	**Avg.**
	AQUA	GSM8K	SVAMP	ASDiv	StrategyQA	CSQA	ARC	LastLetter	
CoT Prompting	58.51	75.89	80.10	87.07	72.88	74.12	85.26	74.40	76.03
Majority Voting	62.60	81.27	83.80	88.36	74.06	77.48	87.31	76.40	78.91
Direct Scoring	63.39	80.14	82.60	88.69	73.14	78.36	87.20	76.60	78.77
Zero Ranking	67.72	79.98	83.30	89.36	74.55	78.21	**87.57**	74.80	79.44
RankPrompt	**71.65**	**82.43**	**84.30**	**90.12**	**76.07**	**79.20**	87.42	**77.60**	**81.10**
Oracle	79.53	91.05	91.40	94.18	85.37	85.26	92.83	86.40	88.25

Table 3: Comparisons of the accuracy on 8 reasoning tasks with `gpt-3.5-turbo`. CoT Prompting uses greedy decoding (temp=0), while other methods sample 5 candidates (temp=0.7). The best performance for each task under the same settings is shown in **bold**.

`gpt-4`, via the OpenAI API[1]. Additionally, we test a variant of ChatGPT, `gpt-3.5-turbo-16k`, which supports an input length of up to 16K, to analyze the impact of varying numbers of exemplars and candidates. Our experimental evaluations were carried out between August 1, 2023, and October 1, 2023.

Tasks and Datasets. We conduct experiments with `gpt-3.5-turbo` across 8 widely-used reasoning tasks, spanning arithmetic, commonsense, and symbolic reasoning. For arithmetic reasoning, we use 4 math word problem datasets: AQUA-RAT (Ling et al., 2017), ASDiv (Miao et al., 2020), GSM8K (Cobbe et al., 2021), and SVAMP (Patel et al., 2021). For commonsense reasoning, which requires multi-step problem-solving, we utilize ARC Challenge (Clark et al., 2018), CommonsenseQA (Talmor et al., 2019), and StrategyQA (Geva et al., 2021). We evaluate symbolic reasoning with the Last Letter Concatenation task (Wei et al., 2022b). Given the high API cost[2], we reserve `gpt-4` for 3 challenging reasoning tasks from BIG-Bench-Hard (Suzgun et al., 2023): Causal Judge, Logical Deduction Seven Objects, and Formal Fallacies. Following Wang et al. (2023d), we report the accuracy on the test set for all tasks except CommonsenseQA, where we use the validation set. Additionally, we test RankPrompt on AlpacaEval (Dubois et al., 2023), a benchmark for measuring LLM-based automatic evaluation of open-ended generation. The benchmark comprises 805 instructions, each with a pair of responses and 4 human preferences. We compare different methods using `gpt-4` and report the level of agreement with human preferences.

Candidate Generation Setups. For a fair comparison, we employ the same prompts created by Wei et al. (2022b) and Suzgun et al. (2023) for candidate generation. We use a temperature of 0.7 to generate 5 reasoning paths as candidates. We restrict our selection to 5 candidates, as increasing this number yields only marginal performance improvements. Additionally, adding more candidates would increase the API costs due to context expansion. In Section 5.2, we thoroughly analyze the impact of candidate numbers on the results.

Ranking Setups. We leverage language models to rank their outputs. For each task, a task-specific comparison exemplar is generated using the same model utilized for candidate generation. These exemplars systematically evaluate 5 unique candidate responses, ultimately guiding models to the correct answer. Following this, we integrate these exemplars into the ranking template, as detailed in Table 2. Despite the diverse nature of tasks, we maintain a uniform application of comparison instructions and task-specific exemplars, introducing minor modifications to the output format depending on the task type. We restrict our use of comparison exemplars to a single one, as our findings suggest that increasing the number of exemplars has a negligible effect on improving performance but significantly extends the input, often exceeding the maximum length limit of OpenAI models. In Section 5, we conduct a comprehensive examination of how various facets of comparison exemplars influence the final performance.

Baselines. We compare our methods with 4 baseline methods: CoT Prompting (Wei et al., 2022b), Majority Voting (Wang et al., 2023d), Direct Scoring (Zheng et al., 2023), and Zero Ranking. Majority Voting selects the answer that appears most frequently. At the same time, Direct Scoring uses the prompt template proposed by Zheng et al. (2023) to evaluate candidates independently, soliciting Large Language Models (LLMs) to rank candidates on a scale from 1 to 10. Zero Ranking,

[1] `https://platform.openai.com/docs/api-reference`

[2] `https://openai.com/pricing`

Method	Logical Deduction	Causal Judge	Formal Fallacies
CoT Prompting	57.60	69.52	76.80
Majority Voting	62.40	72.19	82.40
Direct Scoring	61.20	71.12	81.60
Zero Ranking	63.70	72.51	83.20
RankPrompt	**66.80**	**74.73**	**84.40**
Oracle	90.00	79.14	92.40

Table 4: Test accuracy on 3 challenging BBH tasks using `gpt-4` over 5 candidates.

the final baseline, employs the comparison instruction shown in Table 2, but excludes the comparison exemplars.

4.2. Main Results

Table 3 summarizes the experimental results on 8 reasoning tasks using `gpt-3.5-turbo`. The CoT Prompting method stands out as it employs greedy decoding at a temperature of 0, while other methods sample 5 candidates at a temperature of 0.7. We also report the *oracle* results, which represent the upper bounds of re-ranking, identified by selecting the optimal response from all possible candidates.

The results demonstrate that both the voting and ranking methods considerably outperform CoT Prompting. Majority Voting and Direct Scoring show similar performance (averaging 78.91 and 78.77, respectively), slightly falling behind Zero Ranking (which averages 79.44). Notably, RankPrompt emerges as the best-performing method, achieving the highest scores in all categories except for ARC, where all methods demonstrate comparable performance. We also find that RankPrompt is more effective for challenging tasks such as AQuA-RAT, GSM8K, and CSQA. In particular, it significantly surpasses other methods on the AQuA-RAT dataset, achieving a 13% improvement over CoT Prompting. These findings highlight the importance of incorporating comparison exemplars in the ranking process. Additionally, the Oracle results signal considerable potential for future enhancements in ranking methods.

4.3. Results on More Challenging Tasks

To further probe the performance on complex tasks, we test various methods on 3 challenging BIG-Bench Hard (BBH) tasks using `gpt-4`. We apply the prompt templates created by Suzgun et al. (2023) for the CoT Prompting baseline and generate candidates with identical settings as described in Section 4.2.

Table 4 shows the experimental results. We observe that Majority Voting beats Direct Scoring, yet falls short when compared to Zero Ranking. RankPrompt emerges as superior over all other methods, achieving performance improvements ranging from 5.2% to 9.2% compared to CoT Prompting. These results validate that RankPrompt is highly effective for complex reasoning tasks.

Method	Human Agreement	Price
Inter-Human	65.70	$241.50
Direct Scoring	64.48	**$11.19**
AlpacaFarm	67.22	$12.35
Alpaca Evaluator	70.13	$14.23
Zero Ranking	71.67	$16.74
RankPrompt	**74.33**	$19.18

Table 5: Human agreements and cost on the test set of AlpacaEval using `gpt-4`. Inter-Human denotes the average results of human annotators.

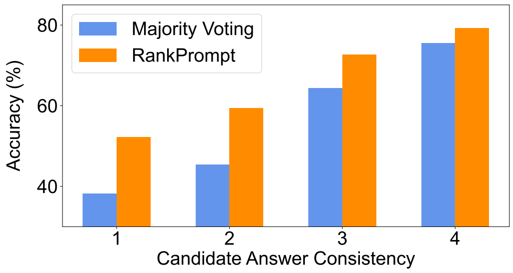


Figure 2: RankPrompt performs much better than majority voting when the candidate answers are inconsistent. The results are obtained on AQuA-RAT over 5 candidates using `gpt-3.5-turbo`.

4.4. Results on Inconsistent Candidates

The results mentioned above show that RankPrompt consistently outperforms Majority Voting across various tasks. We delve deeper into the results of AQUA-RAT by categorizing candidates based on their consistency. We determine consistency by the frequency of major answers among the candidates. Suppose we have n candidates. When all candidates are identical, the consistency reaches n, eliminating the need for re-ranking. Conversely, in the most challenging scenario where all candidates are unique, the number of consistent answers drops to 1. We conduct experiments with `gpt-3.5-turbo` on the AQUA-RAT dataset, maintaining the same settings as in Section 4.2.

Figure 2 illustrates that RankPrompt and Majority Voting exhibit high accuracy when the answer candidates are consistent, especially when there are more than 3 consistent answers. However, the performance dramatically drops when the number of consistent answers is less than 3. Despite this decrease, RankPrompt notably outperforms the voting method. These observations validate our motivation that relying solely on the final answer does not guarantee accurate identification of the optimal candidate.

4.5. Results on Automatic Evaluation

In this section, we delve deeper into the effectiveness of RankPrompt by examining its performance in automatic evaluation tasks. We test RankPrompt on the AlpacaEval benchmark introduced by Dubois et al. (2023). This benchmark comprises a test set of 805 instructions, each accompanied by pairs of responses, designed to assess the instruction-following abilities of language models. Our comparison incorporates Direct Scoring (Zheng et al., 2023), AlpacaFarm, AlpacaEval (Dubois et al., 2023), and Zero Ranking. We assess the performance of each method by calculating the agreement rate with the majority of human preferences, a critical metric for understanding how well each approach aligns with human judgment. Additionally, we present a detailed analysis of the costs associated with each method, including the expenses related to human annotations as reported by Dubois et al. (2023). We experiment with `gpt-4` and present the results in Table 5. RankPrompt outperforms all other methods, achieving a 74.33% agreement rate with human evaluators—Direct Scoring, however, trails by a significant 10% margin. Interestingly, LLM-based evaluators not only yield superior results but also reduce cost by more than 90% compared to crowd-sourced annotators. These findings underscore the critical role of appropriate instructions and exemplars when comparing candidate answers.

5. Analysis

In this section, we thoroughly study the factors that influence ranking performance. Specifically, we examine the effect of exemplars and candidate reasoning paths on ranking outcomes. We also analyze the errors produced by different methods in the complex arithmetic reasoning task. Through this analysis, we aim to deepen the understanding of our proposed method.

5.1. Impact of Comparison Exemplars

Exemplar correctness is the key to the performance of RankPrompt. A fundamental component of RankPrompt is its selection of comparison paths that yield the correct answers. It has been established that, in almost all cases, the intermediate steps generated by LLMs are also correct when the final result of inference is accurate (Wang et al., 2023a). Here, we aim to shed light on how the accuracy of the comparison exemplars influences the overall effectiveness of our method. In the experiments, we condition `gpt-3.5-turbo` with no exemplars, correct exemplars, and incorrect exemplars, respectively. We adhere to the settings specified in Section 4.2 for candidate generation and apply a single exemplar for ranking. Our evaluation comprises 3 tasks: GSM8K, AQUA-RAT, and StrategyQA. As illustrated in Figure 3, the use of incorrect exemplars invariably compromises the performance of the ranking, particularly in more challenging tasks such as AQUA-RAT. On the other hand, the application of correct exemplars consistently enhances the accuracy when contrasted with the use of no exemplars or inconsistent ones. These findings establish that choosing the correct exemplars is essential for RankPrompt.

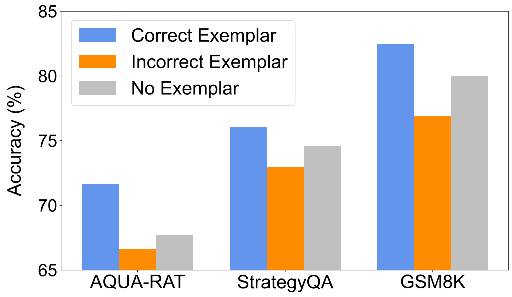


Figure 3: Performance of RankPrompt with a correct example vs. an incorrect example when ranking over 5 candidates. The results are obtained with `gpt-3.5-turbo`.

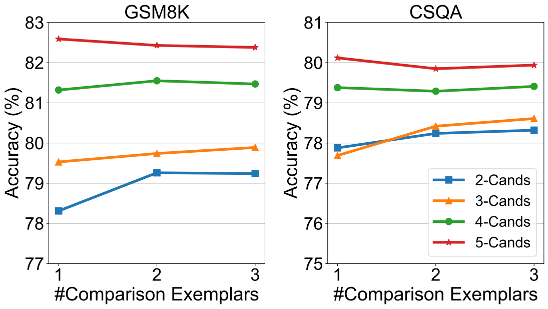


Figure 4: Test accuracy with varying complexity and numbers of comparison exemplars. The results are obtained on GSM8K (left) and CSQA (right) using `gpt-3.5-turbo-16k`.

Exemplar complexity is much more important than quantity. Beyond exemplar correctness, we delve into the influences of complexity and quantity on ranking performance. Intuitively, ranking an expansive and diverse set of candidates inherently possesses greater complexity. This complexity may serve as a reflection of the depth and detail involved in the ranking process. We utilize the count of unique candidates involved in a single comparison exemplar as an indicator of its complexity. We perform ranking over 5 candidates using `gpt-3.5-turbo-16k`, which supports up to 16K tokens. For instance, Figure 4 presents the results from the GSM8K test set. "N-Cands" denotes an exemplar that illustrates the ranking process across N different candidates. The results reveal that the complexity of exemplars is much more important than the quantity. Remarkably, we find that employing a single *complex* exemplar is more effective than using multiple *simple* exemplars.

5.2. Impact of Candidate Answers

We have demonstrated that RankPrompt is robust to the inconsistency in candidate answers in Section 4.4. Here, we further investigate the behaviors of different methods by varying the number and order of candidates.

Using more candidates offers minor benefits. In our main experiments, we opt for 5 candidates, partially due to the input length constraint of LLMs. For instance, `gpt-3.5-turbo` has a 4096-token limit. Here, we explore the impact of increasing the number of candidates using `gpt-3.5-turbo-16k`. We evaluate CoT Prompting, Majority Voting, and RankPrompt on the test sets of GSM8K and CSQA, varying the number of sampled reasoning paths (1, 3, 5, 10, 15). As plotted in Figure 6, both RankPrompt and Majority Voting show improved performance with more candidates, but the gains plateau beyond 5 reasoning paths. While further increasing the number of candidates offers slight improvements, it also significantly raises the cost. Hence, we recommend using 5 candidates to make trade-offs between performance and cost.

RankPrompt is robust to the ordering of candidates. A good evaluator should exhibit robustness against variations in the order of candidate answers. In this section, we investigate the robustness of different ranking methods on the challenging BBH tasks. We employ the identical experimental setup specified in Section 4.3 and run the ranking process 3 times, with candidate orderings being shuffled each time. Instead of reporting the overall accuracy, which would gain from increasing individual reasoning paths, we focus on the prediction consistency across different methods. Specifically, we regard a ranking as *consistent* if it remains unchanged across all 3 iterations. As depicted in Figure 5, RankPrompt exhibits greater robustness compared to Zero Ranking when confronted with variations in candidate orders. Specifically, RankPrompt produces consistent rankings ranging from 75% to 85% of the time. These results demonstrate that RankPrompt is a reliable and robust judge for complex reasoning tasks.

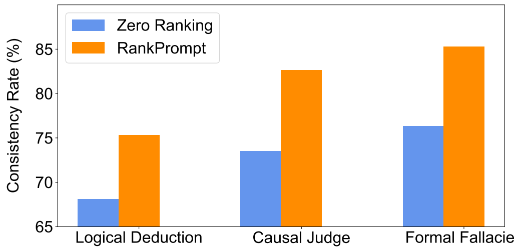


Figure 5: Consistency rates of Zero Ranking and RankPrompt when ranking 5 candidates shuffled 3 times. The results are obtained with `gpt-4`.

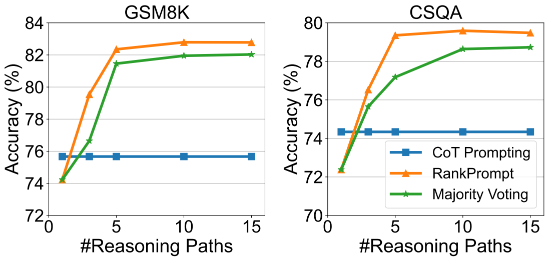


Figure 6: Test accuracy measured against varying numbers of reasoning paths. The results are obtained on GSM8K (left) and CSQA (right) using `gpt-3.5-turbo-16k`. CoT Prompting uses greedy decoding, while others employ sampling (temp=0.7).

5.3. Error Analysis

To gain further insights into how RankPrompt enhances the reasoning performance of language models, we manually analyze the errors made by RankPrompt and CoT Prompting on AQUA-RAT. We utilize the same error categorizations as in (Sawada et al., 2023) for the qualitative analysis of the results in 3. In total, RankPrompt produces 72 errors, while CoT Prompting accumulates 105 errors. We find that RankPrompt mitigates all types of errors identified in CoT Prompting. Interestingly, both CoT Prompting and RankPrompt make a few calculation errors (15 vs. 9). RankPrompt significantly reduces errors caused by wrong approaches (from 42 to 27) but proves less effective in mitigating the impact of misinterpretation (from 17 to 14).

6. Conclusion

We have presented RankPrompt, a novel prompting method for selecting the optimal output from a diverse set of reasoning paths generated by LLMs. This method systematically steers LLMs to compare potential answers, leveraging step-aware comparison instructions and automated exemplars. This approach confers three primary advantages:

Error Type	CoT Prompting	RankPrompt
Calculation Error	15	9
Wrong Approach	42	27
Misinterpretation	17	14
Logical Error	31	22
Total Errors	105	72

Table 6: Error statistics on the AQUA-RAT dataset using `gpt-3.5-turbo`.

(1) it eliminates the need for additional models and human annotations, (2) it achieves strong performance across a broad spectrum of reasoning and automatic evaluation tasks, and (3) it is robust to inconsistent reasoning paths. Our comprehensive evaluation underscores that the precision and complexity of comparison exemplars play a critical role in ranking performance. Collectively, our findings position RankPrompt as an effective strategy to enhance the reasoning capabilities of LLMs.

Acknowledgement

This work was supported in part by the National Science Foundation of China (No.62276056), the Natural Science Foundation of Liaoning Province of China (2022-KF-16-01), the Fundamental Research Funds for the Central Universities (Nos. N2216016 and N2316002), the Yunnan Fundamental Research Projects (No. 202401BC070021), and the Program of Introducing Talents of Discipline to Universities, Plan 111 (No.B16009). The authors would like to thank anonymous reviewers for their insightful comments.

Limitations

Despite the impressive performance of our method, its experiments has been limited to proprietary language models. The lack of publicly accessible training details for these models creates a significant barrier for researchers interested in pursuing enhancements from a modeling standpoint. In the future, we will enhance the ranking capabilities of open-source models like LLaMA (Touvron et al., 2023b,a) and Falcon (Penedo et al., 2023). Learning from the explanations behind GPT-4's ranking decisions offers a promising path for exploration. Additionally, while comparison exemplars in prompts improves performance, they also significantly increases the context size, leading to more expensive API calls. A potential solution is to condense the candidate paths by summarizing their key points.

7. Bibliographical 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. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, John A. Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuan-Fang Li, Scott M. Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *ArXiv preprint*, abs/2303.12712.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457.

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*, abs/2110.14168.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. 2022. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3369–3391, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Tim Dettmers and Luke Zettlemoyer. 2022. The case for 4-bit precision: k-bit inference scaling laws. *ArXiv preprint*, abs/2212.09720.

Thomas G. Dietterich. 2000. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, MCS '00, page 1–15, Berlin, Heidelberg. Springer-Verlag.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori Hashimoto. 2023. Alpacafarm: A simulation framework for methods that learn from human feedback. *ArXiv preprint*, abs/2305.14387.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104, Copenhagen, Denmark. Association for Computational Linguistics.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023a. Gptscore: Evaluate as you desire.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023b. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

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.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and L. Sifre. 2022. Training compute-optimal large language models. *ArXiv preprint*, abs/2203.15556.

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.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *ArXiv preprint*, abs/2210.11610.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065, Toronto, Canada. Association for Computational Linguistics.

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

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California. Association for Computational Linguistics.

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.

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 preprint*, 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 *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.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55:1 – 35.

Yang Liu, Dan Iter, Yichong Xu, Shuo Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment. *ArXiv preprint*, abs/2303.16634.

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.

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 preprint*, abs/2303.17651.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing English math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984, Online. Association for Computational Linguistics.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

OpenAI. 2023a. GPT-4 technical report. *CoRR*, abs/2303.08774.

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

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

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.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2022. Efficiently scaling transformer inference. *ArXiv preprint*, abs/2211.05102.

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2023. Reasoning with language model prompting: A survey. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 5368–5393. Association for Computational Linguistics.

Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023a. Is chatgpt a general-purpose natural language processing task solver? *ArXiv preprint*, abs/2302.06476.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2023b. Large language models are effective text rankers with pairwise ranking prompting. *ArXiv*, abs/2306.17563.

Tomohiro Sawada, Daniel Paleka, Alexander Havrilla, Pranav Tadepalli, Paula Vidas, Alexander Kranias, John J. Nay, Kshitij Gupta, and Aran Komatsuzaki. 2023. Arb: Advanced reasoning benchmark for large language models.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Pengjie Ren, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agent. *ArXiv*, abs/2304.09542.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13003–13051. Association for Computational Linguistics.

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.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam M. Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, Yaguang Li, Hongrae Lee, Huaixiu Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Yanqi Zhou, Chung-Ching Chang, I. A. Krivokon, Willard James Rusch, Marc Pickett, Kathleen S. Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Hartz Søraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Díaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravindran Rajakumar, Alena Butryna, Matthew Lamm, V. O. Kuzmina, Joseph Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed Huai hsin Chi, and Quoc Le. 2022. Lamda: Language models for dialog applications. *ArXiv preprint*, abs/2201.08239.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *ArXiv preprint*, abs/2302.13971.

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. 2023b. Llama 2: Open foundation and fine-tuned chat models. *ArXiv preprint*, abs/2307.09288.

Miles Turpin, Julian Michael, Ethan Perez, and Sam Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *ArXiv preprint*, abs/2305.04388.

Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2023a. Towards understanding chain-of-thought prompting: An empirical study of what matters. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2717–2739, Toronto, Canada. Association for Computational Linguistics.

Jiaan Wang, Yunlong Liang, Fandong Meng, Haoxiang Shi, Zhixu Li, Jinan Xu, Jianfeng Qu, and Jie Zhou. 2023b. Is chatgpt a good nlg evaluator? a preliminary study. *ArXiv preprint*, abs/2303.04048.

Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023c. Large language models are not fair evaluators. *ArXiv*, abs/2305.17926.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023d. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022b. 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.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning. In *NeurIPS*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Z. Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jianyun Nie, and Ji rong Wen. 2023. A survey of large language models. *ArXiv*, abs/2303.18223.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Haotong Zhang, Joseph Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *ArXiv preprint*, abs/2306.05685.

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. 2023a. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023b. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.