NLP Final Project : Default Project RAG Improvement via LoRA Fine-Tuning

KangJun Lee
suri7897@unist.ac.kr

Abstract

This work explores improving a Retrieval-Augmented Generation (RAG) system by fine-tuning its generator using Low-Rank Adaptation (LoRA), a parameter-efficient fine-tuning method. Starting from a default project implementation based on a small GPT model and a sparse BM25 retriever, we observed that the generator frequently produced repetitive, verbose, or semantically incoherent outputs. These issues were attributed to overfitting on limited training data and the generator's inability to follow instructions effectively in context-aware settings.

To mitigate these problems, we applied LoRA-based instruction tuning using the Self-RAG dataset, which contains diverse instruction–response pairs in a structured format. The goal was to guide the generator toward producing more fluent, relevant, and human-aligned outputs. We evaluated the performance of this fine-tuned generator across three task categories: abstractive summarization, text classification, and open-domain question answering via RAG.

Our experiments show that LoRA fine-tuning leads to substantial improvements in ROUGE scores for both summarization and RAG generation tasks, indicating enhanced surface-level fluency and informativeness. However, we also observed that exact match (EM) accuracy in the RAG setting remained low, highlighting challenges related to factual consistency and model capacity. These findings suggest that while LoRA helps improve linguistic quality, careful integration and model scaling are necessary to ensure factual alignment in retrieval-augmented systems.

1 Introduction

Retrieval-Augmented Generation (RAG) (Lewis et al., 2021) combines parametric language models with non-parametric retrieval to enable open-domain question answering without requiring models to memorize factual knowledge. By decoupling knowledge storage from the language model itself, RAG offers a flexible and scalable framework for question answering tasks, especially in dynamic domains where factual updates occur frequently.

In our default project, we implemented a RAG system using a small GPT-based generator trained from scratch. Despite successful training and integration with a BM25-based retriever, we observed that the generator frequently produced unnatural, verbose, or repetitive outputs (e.g., "they say . . . ."), which significantly degraded the overall answer quality. These issues stem from the limited capacity of the GPT-small model and its insufficient alignment with instruction-following objectives, especially in the context of generation conditioned on retrieved passages.

To mitigate these limitations, we explored Low-Rank Adaptation (LoRA) (Hu et al., 2021), a parameter-efficient fine-tuning technique that enables targeted learning by injecting trainable low-rank matrices into selected weight components of the model. Rather than fully updating all parameters, LoRA allows efficient adaptation with minimal computational overhead. We applied LoRA to fine-tune the generator on instruction-style input-output pairs from the Self-RAG dataset (Asai et al., 2023), which contains natural question–answer prompts designed to encourage concise, relevant, and fluent responses.

This fine-tuning process substantially improved the generator's ability to follow instructions and produce coherent answers. Following LoRA-based tuning, we re-integrated the refined generator into the RAG pipeline, preserving the retriever while replacing only the generator. Our goal was to assess whether generation quality improvements translate into better end-to-end RAG performance in terms of both fluency and factual alignment. This report presents the methodology, evaluation results, and analysis of the improvements achieved through

LoRA-based fine-tuning within the RAG framework.

2 Related Work

Large language models (LLMs) such as GPT and T5 have demonstrated remarkable performance across a wide range of natural language processing tasks, including open-domain question answering, dialogue generation, and text summarization. However, these models often require task-specific adaptation to generate responses that are not only accurate but also aligned with human intent and instruction. Despite their strong generalization capabilities, out-of-the-box performance on specialized tasks can be suboptimal without additional conditioning.

To bridge this gap, instruction tuning has emerged as a powerful approach. Models like InstructGPT (Ouyang et al., 2022) are trained on datasets containing diverse instruction–response pairs, which significantly enhance their ability to follow human-written prompts and produce coherent, task-specific outputs. Instruction tuning improves both response quality and robustness in zero- and few-shot settings, making LLMs more useful in real-world applications.

Nevertheless, full fine-tuning of large-scale LLMs is computationally intensive, requiring substantial GPU memory and prolonged training time. Moreover, such fine-tuning risks catastrophic forgetting, where newly learned knowledge may interfere with existing capabilities. To alleviate these challenges, a family of Parameter-Efficient Fine-Tuning (PEFT) methods has been proposed. These approaches aim to achieve comparable performance gains while updating only a small fraction of the model parameters.

Among the notable PEFT techniques are Adapters (Houlsby et al., 2019), which insert lightweight bottleneck modules between existing layers; Prompt Tuning (Lester et al., 2021), which optimizes continuous prompt embeddings; and Low-Rank Adaptation (LoRA) (Hu et al., 2021), which modifies selected weight matrices by injecting trainable low-rank decompositions. These methods allow efficient adaptation with minimal parameter overhead, making them well-suited for resource-constrained environments.

In this work, we adopt LoRA due to its simplicity, scalability, and strong empirical results in prior studies. We apply LoRA to fine-tune a GPT-based generator in a retrieval-augmented generation (RAG) framework, with the goal of enhancing fluency, coherence, and contextual relevance in generated answers. By doing so, we seek to improve the effectiveness of RAG systems in open-domain QA settings without incurring the computational cost of full-model tuning.

3 Approach

Our approach consists of two main stages: (1) fine-tuning the generator using LoRA to improve generation quality, and (2) integrating the improved generator back into the RAG pipeline.

3.1 Baseline: Default RAG Pipeline

Since the default project was implemented collaboratively, we briefly summarize the shared baseline. It consists of two parts: (1) building and fine-tuning a GPT-small model, and (2) constructing a RAG system. The GPT-small model, implemented with RoPE, GQA, and RMSNorm, was pretrained from scratch and then fine-tuned on summarization and classification tasks.

This model was used as the generator in a RAG pipeline with a sparse BM25 retriever over a Wikipedia corpus. The system was trained on the NQ dataset using positive contexts. Additionally, zero-shot prompting with LLaMA-3.2-1B-Instruct was explored using various prompt templates.

3.2 Finetuning with LoRA

To improve generation quality, we applied Low-Rank Adaptation (LoRA) to fine-tune the GPT-based generator in a parameter-efficient manner. Using the Self-RAG codebase and the PEFT library, we injected trainable low-rank matrices into the attention layers while keeping the base model frozen. The model was trained on an instruction-style dataset, with prompt tokens masked during loss computation to focus learning on response generation. Further information of dataset are given in Experiments.

After fine-tuning, the updated generator was re-integrated into the original RAG framework, replacing only the generator while keeping the retriever fixed. This setup enabled us to evaluate whether the improved generator could produce more fluent and contextually relevant answers in an end-to-end QA setting.

4 Experiments

4.1 Default Project

4.1.1. Evaluation of the GPT-small Model

After pretraining, the GPT-small model was fine-tuned on two downstream tasks: summarization and classification. The evaluation metrics used for these tasks were the following:

- Summarization: ROUGE scores (ROUGE-1, ROUGE-2, ROUGE-L) were computed using the evaluate library.
- Classification: Accuracy was measured to assess how well the model predicted class labels based on input text.

These evaluations provided a way to check whether the model had learned general language understanding capabilities and could perform on task-specific datasets.

4.1.2. Evaluation of the RAG System

For the RAG system, which used the fine-tuned GPT model as its generator, the evaluation was based on the Natural Questions (NQ) dataset. The steps included:

- **Retrieval**: Top-k relevant passages were retrieved from a Wikipedia corpus using a BM25 retriever.
- **Generation**: The generator produced answers conditioned on both the input question and the retrieved context passages.
- **Exact Match (EM) Accuracy**: The final prediction was evaluated based on whether it exactly matched the ground-truth answer.

The evaluation was carried out using the evaluate.py script provided in the project, which compared generated answers to reference answers from the NQ dataset to compute the EM score.

4.2 Finetuning with LoRA

To improve the generation quality and factual consistency of the RAG system, we applied Low-Rank Adaptation (LoRA) to fine-tune the pretrained GPT generator in a parameter-efficient manner. LoRA introduces trainable low-rank decomposition matrices into selected weight components (typically attention projections), enabling effective adaptation while keeping the majority of the model frozen. This makes it especially well-suited for small models with limited computational resources.

The fine-tuning process was conducted using a dataset given by Self-RAG[1], which contains instruction-style samples tailored for retrieval-augmented generation. Each instance in the dataset is formatted as a natural-language prompt paired with a structured response, designed to encourage instruction-following behavior and coherent output generation. The dataset consists of a collection of publicly available sources, including FLAN v2, ARC_Easy, NQ, and others. For efficiency, we randomly sampled 30% of the data for training.

The training prompts followed a standardized completion format as shown below:

```
 Instruction:\n Q: Is there a negative
or positive tone to this product review?
...
Response:\n
[No Retrieval]Negative[Utility:5]
```

We designed three sets of experiments to systematically evaluate the impact of LoRA-based fine-tuning on both standalone generation and integrated RAG tasks:

1. **Summarization Task**:
   We first applied LoRA to fine-tune the pretrained GPT generator on a summarization dataset. The goal was to assess whether LoRA improves surface-level generation quality. Model performance was evaluated using standard ROUGE metrics (ROUGE-1, ROUGE-2, and ROUGE-L), comparing the LoRA-tuned model against the baseline pretrained model without LoRA. The results confirmed that LoRA enhances sentence fluency and content preservation.

2. **RAG Generation Task**:
   Next, we fine-tuned the generator within the RAG framework using LoRA, while keeping the retriever fixed. The objective was to measure whether LoRA helps the generator better utilize the retrieved context in answering open-domain questions. Performance was evaluated on the Natural Questions (NQ) dataset using Exact Match (EM) accuracy, allowing us to quantify factual correctness in an end-to-end QA setting.

[1]available in github for Self-RAG

3. **Pretrain → LoRA → RAG Finetuning**: Lastly, we examined the effect of applying LoRA-based instruction fine-tuning *before* RAG-specific fine-tuning. This allowed us to test whether early-stage supervised alignment improves the model's downstream behavior. We compared three generator variants within the same RAG pipeline:

    - Pretrained-only generator (no LoRA)
    - Generator fine-tuned during RAG training only
    - Generator first fine-tuned via LoRA, then further trained in the RAG loop

    Evaluation was conducted using both ROUGE (to assess fluency) and EM accuracy (to assess factual alignment), providing a comprehensive view of generation quality under different training strategies.

These experimental conditions allow us to isolate the effect of LoRA fine-tuning and understand how it influences both local generation quality and global QA performance within the RAG architecture. Detailed results and analysis are presented in result, highlighting key performance trends and limitations.

5 Results and Analysis

5.1 Default Project

Metric	Score
ROUGE-1	0.0025
ROUGE-2	0.0000
ROUGE-L	0.0025
ROUGE-Lsum	0.0025

Table 1: ROUGE scores for the summarization task after fine-tuning

Metric	Score
Accuracy	0.3818

Table 2: Classification accuracy of the finetuned-GPT model for CF

Example Output (from summary model)

```
Indiana town's Memories Pizza is shut down
after online threat....
(omission until end)
and and and and and and...
```

Example 1: Sample summary output

As shown in Table 1 and Table 2, the ROUGE scores for the summarization task and the accuracy for the classification task were lower than expected. We believe this is primarily due to the limited size of the training dataset and significantly small parameter size of model, which led to overfitting in model. In fact, the pretrained model achieved a perplexity of 1.73 and a token-level accuracy of 0.999 on the evaluation set—indicating possible overfitting. However, as illustrated in Example 1, the actual generated outputs occasionally contain unnatural repetitive phrases (**and** in Example 1), which negatively impacted the downstream performance.

Metric	A	B
Accuracy	0.0002	0.0243
ROUGE-1	0.0006	0.0313
ROUGE-2	0.0000	0.0069
ROUGE-L	0.0006	0.0311
ROUGE-Lsum	0.0007	0.0313

Table 3: Performance comparison between RAG(A) and zero-shot RAG using LLaMA-3.2B-Instruct(B).

Similarly, the RAG model also exhibited low accuracy compared to LLaMA-3.2B-Instruct, which we attribute to the same overfitting issue observed in the pretrained GPT-small generator.

5.2 Finetuning with LoRA

Metric	A	B
ROUGE-1	0.02061	0.00245
ROUGE-2	0.00001	0.0
ROUGE-L	0.01815	0.00245
ROUGE-Lsum	0.01976	0.00245

Table 4: Performance comparison between summary gpt model with LoRA finetuning(A) and without(B).

5.2.1 LoRA finetuning in summary task

As shown in Table 4, we observe a significant improvement in ROUGE scores, indicating that LoRA-based fine-tuning substantially enhances the overall quality of generated sentences. In particular, the model demonstrated better lexical coverage and content relevance, suggesting improved alignment with the reference summaries. Additionally, the occurrence of unnatural or repetitive endings—such as redundant phrases or looping tokens—was notably reduced compared to the model without LoRA. These observations imply that LoRA not only improves surface-level fluency but also helps the model generalize better to unseen inputs in summarization tasks.

5.2.2 LoRA in RAG-finetuned model

Metric	A	B
Accuracy	0.00015	0.00015
ROUGE-1	0.00911	0.00064
ROUGE-2	0.0	0.0
ROUGE-L	0.00905	0.00065
ROUGE-Lsum	0.00909	0.00065

Table 5: Performance comparison between RAG - gpt model with LoRA finetuning(A) and without(B).

Furthermore, we observed in Table 5 improvements in the RAG-trained model after applying LoRA to the generator. While the overall accuracy remained similar, the LoRA-finetuned model achieved substantially higher ROUGE scores. This suggests that LoRA enhances the linguistic fluency and relevance of generated answers, even without explicitly increasing factual correctness. It implies that generation quality and factual alignment can be decoupled to some extent, and that LoRA helps improve surface-level coherence and informativeness.

5.2.3 RAG-Finetuning after LoRA finetuning

Metric	A	B
Accuracy	0.0	0.00015
ROUGE-1	0.01109	0.00064
ROUGE-2	0.0	0.0
ROUGE-L	0.01107	0.00065
ROUGE-Lsum	0.01111	0.00065

Table 6: Performance comparison between RAG - gpt model after LoRA finetuning(A) and without(B).

Interestingly, as we can see in Table 6 after applying LoRA-based fine-tuning and retraining the RAG model, we observed a sharp increase in ROUGE scores but a drop in exact match accuracy to near zero. We suspect this is due to a combination of overfitting in the small generator model and possible misalignment or truncation issues during the fine-tuning process. These results suggest that while LoRA improves surface-level fluency, it may impair factual grounding if not carefully integrated within the RAG architecture—especially in low-capacity models.

Limitations

While our LoRA-based fine-tuning approach improved the fluency and coherence of generated outputs, several critical limitations emerged during experimentation, shedding light on the challenges inherent in applying parameter-efficient methods to small-scale language models.

First, the limited capacity of the GPT-small architecture, coupled with the relatively narrow coverage of the instruction-style dataset, led to severe overfitting. Despite the model achieving extremely low perplexity (1.73) and near-perfect token-level accuracy (0.999) on the training and validation sets, it often produced repetitive, degenerate, or overly verbose sequences. This behavior suggests that the model learned to memorize surface-level patterns in the training data rather than acquiring robust generalization capabilities.

Second, we encountered an embedding mismatch issue during fine-tuning: the pretrained checkpoint used an embedding vocabulary size of 50266 tokens, whereas the initialized model expected a size of 50258. This discrepancy likely caused incomplete token mapping, silent truncation, or embedding misalignment—each of which can compromise the semantic fidelity of the generated text and introduce subtle inconsistencies during both training and inference.

Third, while ROUGE scores increased notably after applying LoRA—indicating improvements in fluency and lexical similarity—the exact match (EM) accuracy of the RAG system dropped close to zero. This performance gap implies that LoRA-enhanced generation may prioritize stylistic or surface-level improvements at the expense of factual precision, particularly when the model is not sufficiently grounded by the retriever context or when generation alignment is weak.

These findings highlight several directions for future work: scaling the generator model to increase representation capacity; enhancing alignment between the retriever and the generator, potentially through joint optimization or feedback mechanisms; expanding and diversifying instruction datasets to reduce overfitting; and ensuring tokenizer consistency across all stages of training. Addressing these challenges is essential for realizing the full potential of LoRA in retrieval-augmented generation systems, especially under low-resource or small-model constraints.

Appendix

A. Pretrained model

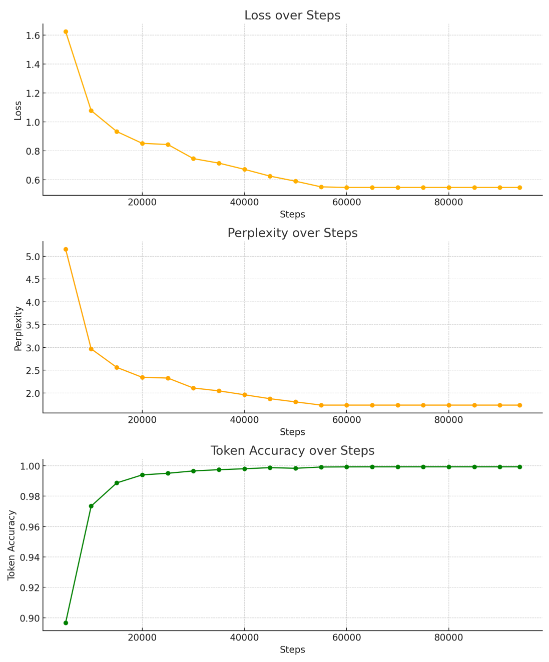

Figure 1: Token accuracy, loss, and perplexity of the pretrained generator across training steps.

Metric	Best Score
Loss	0.546875
Perplexity	1.734375
Token Accuracy	0.99927

Table 7: Best Score during pretraining

As shown in Figure 1, the reported metrics—such as perplexity and token-level accuracy—are unexpectedly high, which suggests that the model may have overfitted to the training data. We believe this phenomenon stems from the limited size of both the dataset and the model itself. Unlike GPT-2 (Radford et al., 2019), which consists of 1.5 billion parameters and was trained on a large-scale corpus containing 8 million documents and approximately 10 billion words, our model is significantly smaller and was trained on a much more constrained dataset. This substantial difference in both parameter count and data scale limits the model's capacity to generalize and increases the likelihood of memorizing patterns from the training set. Consequently, while the training metrics appear impressive, they may not reflect true generalization performance and instead indicate overfitting due to insufficient training diversity and capacity.

B. LoRA difference in Summary model output

From original summary model

```
Singing the national anthem is a risky
proposition . Whitney Houston nailed it;
Roseanne Barr destroyed it .
(omission until end)
```
and and and and and and...

From finetuned summary model

```
Singing the national anthem is a risky
proposition . Whitney Houston nailed it;
Roseanne Barr destroyed it .
(end of sentence)
```

Example 2: Summary model Output

As illustrated in Example 2, the fine-tuned summarization model is less prone to generating repetitive words compared to the baseline. Although some repetition still occurs, particularly near the end of generated outputs, the frequency and severity are noticeably reduced. This aligns with the quantitative improvements shown in Table 4 and Table 5, where the LoRA fine-tuning method led to higher ROUGE scores, indicating more fluent and human-like text generation.

Nonetheless, occasional repetitive patterns such as "and and and..." still appear in some outputs. This phenomenon may be attributed to overfitting, particularly due to the limited size of both the dataset and the model. It is also possible that the model, when uncertain or under-trained in specific contexts, defaults to high-probability tokens repeatedly—especially if decoding strategies like top-k or top-p sampling are not carefully tuned.

C. Effect of change in Prompt at Zero-shot

Original Prompt

```
Title:
Passage: {passage_1}
Title:
Passage: {passage_2}
...
Title:
Passage: {passage_n}

Question: {query}
Answer:
```

My own Prompt

```
<|begin_of_text|><|user|>
DOCUMENT:{document}
QUESTION:{query}
INSTRUCTIONS:
Answer  the  user's  QUESTION  using  the
DOCUMENT textabove.
Keep your answer grounded in the facts of
the DOCUMENT.
If the DOCUMENT doesn't contain thefacts to
answer the QUESTION, return{{NONE}}.
<|end_of_text|><|assistant|>
```

CoT Prompt

```
Title:
Passage: {passage_1}
Title:
Passage: {passage_2}
...
Title:
Passage: {passage_n}

Question: {query}
Let's think step by step.
Answer:
```

Example 3: Used Prompt

In default-project, we evaluate various prompts for retrieval-augmented generation (RAG). In this experiment, we compare three types of prompts, as illustrated in Example 3: the default prompt, which matches the output format; a custom prompt, which is significantly different from the output format for dramatic difference in evaluation; and a chain-of-thought (CoT) prompt, which retains the original format while encouraging intermediate reasoning. Introduced in(Wei et al., 2023), chain-of-thought prompting enables complex reasoning through step-by-step thinking. We evaluate these prompts in a zeroshot RAG setting, analyzing both the generated outputs and overall performance.

Metric	Original	Custom	CoT
Accuracy	0.02425	0.00107	0.01120
ROUGE-1	0.03126	0.00775	0.02148
ROUGE-2	0.00690	0.00027	0.00227
ROUGE-L	0.03111	0.00769	0.02109
ROUGE-Lsum	0.03127	0.00770	0.02119

Table 8: Performance comparison in Zeroshot-RAG - with original prompt(Original) with custom prompt(Custom), and with Chain-of-Thought prompt(CoT) .

Custom Prompt

```
eval_5824
who won the icc women 's world cup 2017
['england']
answer: {{none}}parableparableparable
```

CoT Prompt

```
eval_5819
who finished the dark knight as the joker
['ledger']
answer:
step 1: the passage does not mention
```

Example 4: Output of custom and CoT prompted model

The result of the custom prompt shows lower evaluation scores, which is expected given its deviation from the standard output format. The significant difference in prompt structure leads to a mismatch with the expected answer format, resulting in decreased accuracy and overall generation quality. One notable observation is that the output frequently contains {{NONE}}, as specified in the prompt, indicating that the prompt strongly influences the model's decision.

However, the result of the CoT prompt is somewhat unexpected, as its performance is significantly lower than that of the original prompt. Although the CoT prompt successfully induces the model to begin its answer with "Step 1", as intended, we observed that none of the outputs included a subsequent "Step 2" or any further reasoning steps. This indicates that while the model partially follows the structural instruction embedded in the prompt, it fails to fully engage in multi-step reasoning. At this point, the underlying cause of this behavior remains unclear. It is possible that the model's instruction-following capability is limited in zeroshot settings, or that the CoT format requires more explicit few-shot demonstrations to be effective. Further analysis is planned to investigate this behavior in depth and to determine whether the prompt structure, decoding strategy, or model limitations contribute to the incomplete reasoning process.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. *Preprint*, arXiv:1902.00751.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *Preprint*, arXiv:2104.08691.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Long Ouyang, Jeff 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 Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI*. Accessed: 2024-11-15.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.