

# Are LLMs good at Mathematical Reasoning ?

Arnav Agarwal      Aayush Ranjan      Harshvardhan Singh  
Parthiv Dholaria      Utsav Garg      Pulkit Nargotra

September 2024

## Abstract

This study explores the mathematical reasoning capabilities of state-of-the-art Large Language Models (LLMs) using complex datasets such as SVAMP and GSM8K. With applications in education, finance, and robotics, effective logical reasoning in LLMs is critical yet remains a challenging frontier. We investigate and compare diverse reasoning techniques, including zero-shot prompting, chain-of-thought (CoT), Retrieval-Augmented Generation (RAG), self-correction, and multi-agent frameworks.

The evaluation spans models of varying capacities, such as Meta-LLaMA 70B, LLaMA 8B, Gemma 9B, and Mistral8x7B, emphasizing the trade-offs between model size and reasoning efficacy. Our findings demonstrate that methods like multi-agent systems and RAG significantly improve accuracy on multi-step reasoning datasets. Building on these findings, we propose a novel approach that leverages collaboration between multiple agents, each specialized in some reasoning chain. This framework outperforms our baselines and opens new directions for future research.

Furthermore, the study identifies key limitations in handling irrelevant details and complex logical chains. This work provides actionable insights into enhancing LLM architectures and strategies for future applications requiring robust mathematical and logical reasoning. The full project and implementation details are available on our GitHub repository.

## 1 Introduction

The field of mathematical and logical reasoning presents a critical challenge for the advancement of artificial intelligence, particularly in the domain of Large Language Models (LLMs). Modern LLMs, such as Meta-LLaMA, Gemma, and Mistral, exhibit impressive capabilities in natural language processing, but their ability to perform accurate and logical reasoning, especially in mathematical contexts, remains an area ripe for exploration. This report investigates these reasoning capabilities by evaluating their performance on complex datasets such as SVAMP and GSM8K, which are designed to test multi-step arithmetic and logical reasoning.

## 1.1 Motivation

The motivation for this study stems from the increasing reliance on LLMs for applications that demand not just fluency in natural language but also robust logical inference and precise computation. Tasks such as financial analysis, education technology, and advanced robotics often require multi-step problem-solving skills that challenge existing models. Understanding the limitations and strengths of LLMs in these contexts is crucial for their deployment in real-world scenarios.

## 1.2 Problem Statement

Mathematical reasoning tasks pose unique challenges for LLMs as they require both logical consistency and computational accuracy. The complexity increases when datasets like SVAMP and GSM8K introduce real-world problem scenarios that demand semantic understanding and multi-step inference. Addressing these challenges is key to unlocking the full potential of LLMs.

## 1.3 Contributions

This report makes the following significant contributions:

- **Comprehensive Evaluation:** We rigorously evaluate state-of-the-art LLMs using multiple reasoning paradigms, including zero-shot prompting, chain-of-thought (CoT) reasoning, and Retrieval-Augmented Generation (RAG). Additionally, we explore advanced methodologies like multi-agent systems and their integration with RAG.
- **Novel Frameworks:** The study incorporates a multi-agent framework designed to simulate collaborative reasoning, enhancing accuracy through role-specific agents that iteratively refine solutions.
- **Detailed Analysis of Techniques:** By comparing approaches such as self-correction, ReAct prompting, and fine-tuning (e.g., QLoRA, IA3), we identify optimal strategies for improving mathematical reasoning.
- **Insights on Model Architectures:** The performance of models of varying sizes, such as Meta-LLaMA 70B, LLaMA 8B, Gemma 9B, and Mistral, is analyzed to highlight the relationship between model capacity and reasoning efficacy.
- **Impactful Findings:** Through experiments on perturbed datasets (e.g., GSM-Hard), this study identifies areas where LLMs struggle, such as handling irrelevant details, and proposes avenues for enhancing meta-reasoning capabilities.

## 1.4 Objective

This work aims to establish a robust evaluation framework for LLMs in mathematical reasoning, propose methodologies to enhance their logical capabilities, and identify practical strategies for improving their deployment in domains requiring accurate reasoning. This study serves as a stepping stone for developing more advanced reasoning systems capable of addressing the complexities of real-world tasks.

# 2 Literature Review

## 2.1 Benchmarks

### 2.1.1 LogicBench

[1] LogicBench systematically evaluates the logical reasoning capabilities of Large Language Models (LLMs) across 25 distinct reasoning patterns, including propositional, first-order, and non-monotonic logic. Tasks include binary question-answering (BQA) and multiple-choice question-answering (MCQA), which require models to derive conclusions from logical premises. Despite advancements, LLMs such as GPT-4 struggle with tasks involving negations and nuanced logical contexts, highlighting significant limitations in symbolic reasoning (LogicBench, 2024).

### 2.1.2 ProofWriter

[2] ProofWriter assesses multi-step deductive reasoning, including single-step to 5-hop tasks, using natural language facts and rules in closed- and open-world settings. Results show performance degradation as task complexity increases, particularly in open-world scenarios, revealing challenges in logical consistency and multi-step reasoning (ProofWriter, 2024).

### 2.1.3 PrOntoQA

[3] PrOntoQA evaluates deductive reasoning over fictional concepts under closed-world assumptions, avoiding reliance on pre-trained knowledge. Findings indicate that LLMs struggle with reasoning involving fictional entities, with models like GPT-4 performing inconsistently in multi-step tasks, underscoring the need for improved abstraction capabilities (PrOntoQA, 2024).

## 2.2 Prompt Engineering

### 2.2.1 Chain-of-Thought (CoT) Prompting

[4] CoT prompting encourages sequential problem-solving, improving accuracy in simple tasks but faltering on complex multi-step reasoning challenges (Tree of Thoughts, 2024).

### **2.2.2 Least-to-Most Prompting**

[5] This technique improves logical consistency by decomposing complex problems into smaller subproblems, outperforming CoT in structured problem-solving (Least-to-Most Prompting, 2024).

### **2.2.3 Tree of Thoughts (ToT) Prompting**

[6] ToT employs structured exploration of multiple reasoning paths, excelling in tasks like the "Game of 24" and outperforming CoT in handling intricate logical challenges (Tree of Thoughts, 2024).

## **2.3 Self-Correction in LLMs**

[7] Self-correction allows LLMs to revise initial outputs to improve accuracy. Intrinsic self-correction, relying on internal mechanisms, often degrades performance, as seen in models like GPT-3.5 and GPT-4. Extrinsic self-correction, leveraging external feedback, improves outcomes, such as a 3.5% accuracy boost in CommonsenseQA with oracle labels. Research emphasizes improving prompt design to reduce the need for self-correction (Self-Correction in LLMs, 2024).

## **2.4 Multimodality in Logical Reasoning**

### **2.4.1 MathVista**

[8] MathVista evaluates mathematical reasoning in visual contexts, such as interpreting graphs. However, it focuses narrowly on numerical reasoning (MathVista, 2024).

### **2.4.2 LogicVista**

[9] LogicVista assesses multimodal reasoning tasks, including deductive, numerical, and spatial reasoning. Evaluation shows poor performance across models, particularly in diagrammatic reasoning, with larger models like GPT-4 Vision showing limited improvements (LogicVista, 2024).

## **2.5 Reinforcement Learning**

### **2.5.1 Reasoning via Planning (RAP)**

[10] RAP uses Monte Carlo Tree Search (MCTS) to enhance logical reasoning. It improves performance in tasks like plan generation but suffers from high computational costs and lacks a robust reward model (Reasoning via Planning, 2024).

### **2.5.2 Reinforcement Learning via Symbolic Feedback (RLSF)**

[11] RLSF fine-tunes LLMs with symbolic reasoning tools, achieving substantial gains, such as a +52.64% improvement in compilation accuracy. This approach demonstrates the effectiveness of structured feedback for logical reasoning tasks (RLSF, 2024).

## **2.6 Multi-Agent Systems**

### **2.6.1 Tree-of-Thought Validator Agent**

[12] Multi-agent approaches using validators improve accuracy by exploring diverse reasoning paths and iterative refinement. This approach achieves a 5.6% accuracy improvement over traditional ToT methods (Tree-of-Thought Validator, 2024).

### **2.6.2 LLM Harmony**

[13] Multi-agent collaboration through role-playing enhances commonsense and arithmetic reasoning, resulting in a 15% and 6% improvement, respectively, over single-agent models (LLM Harmony, 2024).

## **2.7 Symbolic Solvers**

### **2.7.1 LOGIC-LM**

[14] LOGIC-LM integrates symbolic solvers with LLMs, resulting in a 39.2% performance improvement over simple prompting techniques by leveraging a self-refinement module (LOGIC-LM, 2024).

### **2.7.2 DiLA**

[15] DiLA incorporates first-order logic constraints into network layers, achieving 100% accuracy in SAT tasks and demonstrating significant runtime advantages (DiLA, 2024).

### **2.7.3 LoGiPT**

[16] LoGiPT eliminates the dependency on external solvers by directly modeling symbolic logic within LLMs, achieving a 13% improvement in deductive reasoning accuracy (LoGiPT, 2024).

## **2.8 Metrics**

### **2.8.1 ROSCOE**

[17] ROSCOE evaluates reasoning processes using metrics like semantic alignment, logical inference, and language coherence, ensuring logical consistency and fluency (ROSCOE, 2024).

### 2.8.2 RECEVAL

[18] RECEVAL measures correctness and informativeness of reasoning chains, detecting errors such as hallucinations and logical inconsistencies, providing deeper insights beyond final answer evaluation (RECEVAL, 2024).

## 3 Dataset

This section describes the datasets used for evaluating the multi-agent framework enhanced with Retrieval-Augmented Generation (RAG).

### 3.1 SVAMP Dataset

The SVAMP (Single Variable Arithmetic Math Problems) dataset consists of word problems that require arithmetic reasoning over a single variable. These problems are designed to test the model’s ability to perform multi-step reasoning and understand the semantics of the problem statement.

**Example:**

Body: John had 3 apples. He bought 5 more apples.

Question: How many apples does John have now?

Answer: 8

The dataset focuses on simple arithmetic reasoning but challenges the model to correctly parse and process the question in natural language.

### 3.2 GSM8K Dataset

The GSM8K (Grade School Math 8K) dataset is a collection of 8,000 high-quality math word problems designed for evaluating advanced reasoning capabilities of language models. These problems cover a broad range of topics including arithmetic, algebra, geometry, and logical reasoning. Each problem requires a multi-step solution, emphasizing reasoning accuracy.

**Example:**

Question: A book costs \$10. A pen costs \$2. If John buys 3 books and 4 pens, how much does he spend in total?

Answer:  $3 * 10 + 4 * 2 = 30 + 8 = 38$

This dataset provides a benchmark for evaluating models on their ability to handle multi-step reasoning and numerical computations.

## 4 Accuracy Evaluation

Accuracy is the primary metric used to evaluate the performance of the model in solving mathematical reasoning problems. It measures the proportion of correct predictions made by the model compared to the ground truth answers in the dataset.

### 4.1 Comparison Process

To calculate accuracy, the following steps are followed:

- **Ground Truth Labels:** The correct answers for each problem in the dataset are extracted and stored as a reference.
- **Model Predictions:** The model’s output is processed to extract the numerical answer using a regular expression that matches the required output format:

Answer: <<<Numerical Answer>>>

- **Validation of Output:** If the model’s response does not adhere to the predefined format or fails to produce a numerical answer, it is marked as invalid and excluded from accuracy calculation.
- **Comparison:** For valid responses, the extracted numerical answer is compared against the ground truth label. A prediction is considered correct if the numerical answer matches the ground truth exactly.

### 4.2 Accuracy Calculation

Accuracy is computed as the ratio of correct predictions to the total number of valid predictions:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Predictions}} \times 100 \quad (1)$$

Invalid responses are excluded from the denominator to ensure that accuracy reflects the model’s valid performance.

## 5 Baselines

### 5.1 Zero-shot Prompting

The Zero-Shot Prompting technique evaluates the ability of Large Language Models (LLMs) to solve mathematical reasoning problems directly without any prior training or intermediate reasoning steps. The system generates responses

| Method\Dataset         | SVAMP   |       |       | GSM8K   |       |       |
|------------------------|---------|-------|-------|---------|-------|-------|
|                        | Mixtral | Gemma | Llama | Mixtral | Gemma | Llama |
| Zero Shot              | 64.13   | 85.91 | 90.55 | 72.41   | 88.96 | 87.32 |
| COT (Zero shot)        | 75.68   | 89.53 | 89.68 | 72.72   | 89.93 | 87.54 |
| ReAct                  | 84.61   | 73.33 | 79.87 | 67.57   | 57.45 | 75.21 |
| MultiAgent             | 66.00   | 90.00 | 92.00 | —       | —     | —     |
| Before Self-Correction | 77.68   | 90.42 | 86.22 | 77.01   | 85.71 | 83.41 |
| After Self-Correction  | 72.90   | 85.63 | 84.21 | 74.71   | 83.52 | 80.46 |

Table 1: Performance comparison across SVAMP and GSM8K datasets using different reasoning methods and models.

*Note: Cells left blank are due to inconsistent output formats, insufficient computational credits, or resource constraints.*

to math problems based solely on its pre-trained capabilities, providing a direct numerical answer. Gemma and Llama significantly outperform Mixtral, indicating higher baseline performance on this dataset.

#### Technique Summary:

- **Core Principle:** The model is prompted to solve math word problems and return a single numerical answer without additional guidance or intermediate reasoning.

#### Example Interaction:

##### Prompt:

You will be given a math word problem. Your job is to solve this problem and return one numerical answer. The template to answer the query is to have the last line of the output as Answer: <<<Numerical Answer>>>

Question: Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?

## 5.2 Chain of Thought Prompting

Chain of Thought (CoT) prompting is an established method for enhancing reasoning in large language models (LLMs) by encouraging step-by-step intermediate reasoning. The goal is to assess how well the model handles multi-step reasoning problems like those in GSM8k and SVAMP and to use it as a comparative benchmark against other methods.

CoT prompting is implemented by explicitly instructing the model to decompose problems into intermediate steps before providing a final answer.

All models show improvement over Zero Shot, but Llama’s performance remains consistent with its Zero Shot accuracy (89.68%). Once again gemma is found



to be slightly better than Llama on GSM8k and competitive performance on SVAMP.

Prompt: You will be given a math word problem, your job is to think step by step, solve this problem and return one numerical answer. The template to answer the query is to have the last line of the output as Answer: <<<Numerical Answer>>>

Question: A shopkeeper sells 4 apples for \$5 each and 3 bananas for \$2 each. If a customer buys 2 apples and 1 banana, how much does the customer need to pay?

Answer:  
Let's solve this step-by-step:

The cost of one apple is \$5. Therefore, the cost of 2 apples is  $5 \times 2 = 10$  dollars.  
The cost of one banana is \$2. Therefore, the cost of 1 banana is  $2 \times 1 = 2$  dollars.  
Adding the costs together:  $10 + 2 = 12$  dollars.  
The customer needs to pay \$12.

### 5.3 ReAct Prompting

The ReAct (Reasoning + Acting) prompting framework integrates logical reasoning with interactive decision-making to address complex tasks. It iteratively refines solutions by combining reasoning steps with actions, allowing for dynamic adjustments as new information is encountered.

We augment the models with the `llm-math` tool from `langchain`. Tools which contains support for a calculator which can assist the model in basic mathematical operations like addition, subtraction and division. Mixtral shows the highest performance (84.61%), outperforming Gemma, which seems to struggle comparatively. This is very surprising and indicates that the model can do well on integration with tools on SVAMP. Performance drops for both Mixtral and Gemma, indicating limited effectiveness of ReAct on GSM8K. This is due to the problems on GSM8k requiring multi-step reasoning as opposed to the simple calculations on SVAMP.

Prompt:  
You are a helpful assistant that does simple maths calculations.  
You have access to tools like calculators.  
If you need to use a tool, state the action clearly first,  
then wait for the result before providing the final answer.  
Do not provide the final answer until you have received all necessary observations from the tools.

Question: Natalia sold clips to 48 of her friends in April,

and then she sold half as many clips in May.  
How many clips did Natalia sell altogether in April and May?

ReAct Steps:

Reasoning: Compute half of 48 to determine the clips sold in May.

Action:  $48/2=24$

Reasoning: Add the results for April and May.

Action:  $48+24=72$

Final Answer: 72

## 5.4 Self-Correction

Technique Summary:

- **Core Principle:** The self-correction process for LLMs involves a structured multi-round prompting framework. This approach ensures that the model refines its initial response through iterative steps aimed at identifying and addressing errors. The process is designed to evaluate the model's ability to extract, verify, and improve upon its answers across three distinct rounds of interaction.
- **Iterations:**
  - **Round 1:** In the first round, the Large Language Model is prompted with the question and is expected to give an initial response. The response is then given as a prompt to the same LLM for answer extraction.
  - **Round 2:** In the second round of prompting, the LLM is given a prompt to find errors in the previous answer if any.
  - **Round 3:** Lastly, in the third round of prompting, the LLM is asked to improve upon its mistakes and give a final answer which is then again fed into the same LLM to extract the single numerical answer for easy extraction and comparison.

Initial Prompt: \Can you solve the following math problem?  
Christina is planning a birthday party and needs .75 gift bags per  
invited guest, because 1/4 of attendees don't show up.  
She invited 16 friends. Gift bags are \$2 each.  
How much will she spend? Explain your reasoning.  
Your final answer should be a single numerical number,  
in the form \boxed{answer}, at the end of your response."

Prompt 2: \Review your previous answer and find problems with your answer."

Final Prompt: \Based on the problems you found, improve your answer.  
Please reiterate your answer, with your final answer a single  
numerical number, in the form \boxed{answer}."

On SVAMP the self-correction step helps improve results for some models (e.g., Mixtral goes from 77.68% to 85.63%), although a minor drop is seen for others. On GSM8k, self-correction slightly reduces accuracy for Mixtral and Gemma, suggesting potential overcorrection issues. We find that the model performs slightly better before self-correction than after self-correction.

## 5.5 Multi-Agent: Teacher-Student

The Multi-Agent Student-Teacher Framework is designed to improve mathematical reasoning by simulating a guided learning process. In this framework, a Student LLM generates solutions to math problems, while a Teacher LLM evaluates these solutions, providing feedback and corrections. This iterative process ensures accuracy and adherence to a structured response template. On SVAMP, multiAgent achieves the highest score with Llama (92.00%), showcasing the potential of collaborative reasoning methods. We were unable to repeat these experiments for GSM8k due to insufficient credits and resource constraints.

### Technique Summary:

- **Core Principle:** The student and teacher collaboratively solve math problems. The student generates an explanation and answer, while the teacher evaluates and provides corrective feedback.
- **Roles:**
  - \* **Student:** Solves the problem and generates a detailed explanation and final answer, following a fixed response template.
  - \* **Teacher:** Evaluates the student’s response, identifies errors in reasoning or calculations, and provides feedback to refine the solution.
- **Iterative Feedback Loop:** The process continues for multiple cycles, with the student refining the answer based on the teacher’s feedback until a final, accurate response is produced.
- **Template:**

```
Explanation Body Paragraph
**Final Answer: <SINGLE NUMERICAL ANSWER>**
```

# 6 Experiments

## 6.1 NeuroSymbolic Solvers

The symbolic solver introduced in the Logic-LM paper proposes a novel method for solving elementary mathematical problems. This method

leverages Large Language Models (LLMs) to first understand the problem and generate algebraic equations. These equations, which can be multi-variable or polynomial in nature, are represented as strings. A Python library such as SymPy is then used to solve these string-based equations, generating solutions to the mathematical expressions or problems.

**Objective and Improvements** The primary aim of this approach is to address the inherent limitations of LLMs in solving mathematical problems directly, as LLMs lack the capabilities of a dedicated mathematical engine. By utilizing the LLM’s understanding and generation capabilities to formulate mathematical expressions, and combining this with external solvers or engines, the problem-solving capacity of the system can be significantly improved.

**Challenges Faced with Symbolic Solvers** Despite the potential of this approach, several challenges were encountered during implementation:

- **Extraction of Relevant Information:** Extracting algebraic expressions from LLM-generated responses remains a significant challenge. The most effective method, using regular expressions (regexes), often misses some equations, leading to incomplete problem representation.
- **Diversity of Problems:** The diversity of problems introduces inconsistencies in the number and nature of equations generated by the LLM. This makes it difficult and tedious to extract all relevant expressions reliably.
- **Over-generation of Variables:** LLMs frequently introduce more variables than are necessary to solve the problem. This over-generation can render the symbolic solver unable to find a solution due to the unnecessary complexity of the equations.
- **Indeterministic Output:** LLMs generate indeterministic outputs, sometimes attempting to solve the equations themselves after generating them. This behavior can hinder the solver’s task and further complicate the problem, defeating the original intent of using symbolic solvers for problem-solving.

This hybrid neuro-symbolic approach provides an innovative pathway to solving mathematical problems. However, addressing these challenges is critical for improving the robustness and effectiveness of this methodology.

## 6.2 Finetuning

To enhance the logical reasoning capabilities of Large Language Models (LLMs) for specific tasks, multiple finetuning techniques were employed.

This section outlines the approaches utilized, including QLoRA, IA3, and Prompt Tuning, along with their methodologies, configurations, and advantages.

### 6.2.1 QLoRA (Quantized Low-Rank Adaptation)

QLoRA is a parameter-efficient finetuning method designed to minimize memory and computational requirements while adapting LLMs for specific tasks. It achieves this by utilizing quantization and low-rank decomposition techniques:

#### Methodology

- QLoRA reduces model size by quantizing weights into lower-precision formats (e.g., 4-bit or 8-bit).
- Adapts only a small set of parameters (low-rank matrices) while keeping the core model weights frozen.
- Gradients are applied to the decomposed low-rank matrices, ensuring minimal resource usage.

#### Experimental Setup

- **Epochs:** 4
- **Learning Rate:**  $5 \times 10^{-4}$
- **Target Modules:** 'q\_proj', 'k\_proj', 'v\_proj', 'dense'
- **Rank (r):** 32

### 6.2.2 IA3 (Intrinsically Aligned Adaptation of Activations)

IA3 is another parameter-efficient finetuning technique that operates by modifying activation layers instead of weights:

#### Methodology

- Introduces trainable scaling vectors at specific activation layers within the model.
- These vectors scale the activations, allowing task-specific adaptation without updating the majority of the model's parameters.
- Activations are aligned to the target task, ensuring effective adaptation while maintaining efficiency.

## Experimental Setup

- **Epochs:** 4
- **Learning Rate:**  $5 \times 10^{-4}$
- **Target Modules:** 'q\_proj', 'k\_proj', 'v\_proj', 'dense', 'fc1', 'fc2'

### 6.2.3 Prompt Tuning

Prompt tuning involves optimizing a small number of trainable prompt tokens prepended to the input text, without modifying the underlying LLM:

## Methodology

- Trainable embeddings (prompt tokens) are learned and prepended to each input during training.
- These embeddings act as task-specific instructions that guide the model’s reasoning and generation process.
- The model itself remains frozen, focusing solely on optimizing the prompts for better task performance.

## Experimental Setup

- **Epochs:** 4
- **Learning Rate:**  $5 \times 10^{-4}$
- **Virtual Tokens:** 20

## 6.3 RAG

This implementation integrates a Retrieval-Augmented Generation (RAG) mechanism into the multi-agent reasoning framework. The approach combines diverse reasoning techniques with retrieval-based contextual enrichment to improve accuracy and contextual understanding when solving complex reasoning problems.

### Pinecone Integration for RAG

- The Pinecone vector database is used as the retrieval backbone, enabling efficient storage and retrieval of question-answer pairs.
- Knowledge embeddings are created using the SentenceTransformer model (all-MiniLM-L6-v2).
- A subset of 1,000 items from the AQUA-RAT dataset is indexed, providing a rich knowledge base for retrieval.

### Knowledge Base Construction

- The AQUA-RAT dataset is preprocessed into a format containing:
  - \* Questions
  - \* Options
  - \* Correct answers
  - \* Explanations
- Each question is embedded and upserted into Pinecone with meta-data, including the question and its associated answer.

### RAG Workflow Integration

- Workflow:
  1. Embed the given problem using the SentenceTransformer model.
  2. Query Pinecone to retrieve the top-3 most relevant question-answer pairs.
  3. Construct a prompt combining retrieved knowledge and the problem statement.

### LLM-Based Reasoning

- The enriched prompt is processed by **Meta-Llama 3.1-8B-Instruct** / **Gemma 9b** / **Mixtral-8x-7b**.
- The function to get response uses the together client API for response generation.
- A secondary LLM formats the generated answer to extract the final result in the format: `<final answer>`.

### Evaluation

- Tested on 100 samples from the GSM8K and SVAMP test dataset due to credit limits.
- Incorrect predictions were re-evaluated and manually validated using the RAG pipeline.

## 6.4 Multi-Agent Framework: Specializers

[19] Meta-Reasoning Prompting (MRP), introduced by Gao et al., dynamically selects the optimal reasoning method from a pool based on task-specific cues, enhancing accuracy and flexibility. This concept directly inspired our Multi-Agent Framework, which employs specialized agents representing distinct reasoning chains. These agents collaborate





- **Models used:** The Meta-Llama 3.1 model (70B & 9B) , Mixtral-8x7b , Gemma - 9b.
- **Integration with Groq Client:** The function to get response uses the Groq client API for response generation.
- **Evaluation:** The framework’s application to the datasets involved solving 100 test samples from GSM8K and SVAMP, due to credit limits.

## 6.5 Multi-Agent Framework with Knowledge Source

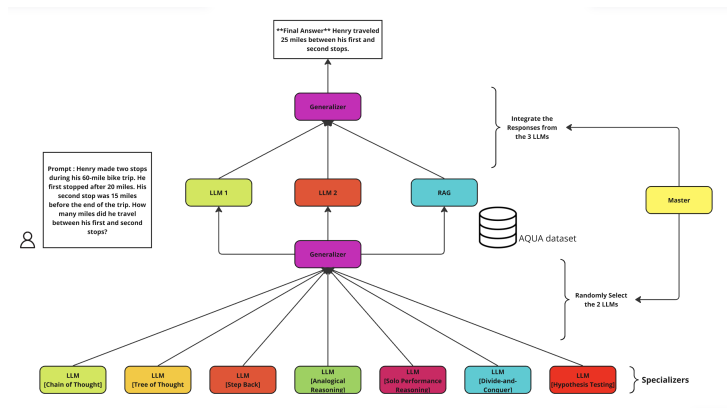


Figure 2: Overview of the Multi-Agent Framework Workflow

This implementation enhances the multi-agent reasoning framework by integrating a Retrieval-Augmented Generation (RAG) mechanism. The approach combines diverse reasoning techniques and retrieval-based contextual enrichment to improve accuracy and contextual awareness when solving complex reasoning problems.

### Pinecone Integration for RAG:

- The Pinecone vector database is used as the retrieval backbone, enabling efficient storage and retrieval of question-answer pairs.
- Knowledge embeddings are created using the SentenceTransformer model (all-MiniLM-L6-v2).
- A subset of 1,000 items from the AQUA-RAT dataset is indexed, providing a rich knowledge base for retrieval.

### Knowledge Base Construction:

- The AQUA-RAT dataset is preprocessed into a format containing questions, options, correct answers, and explanations.
- Each question is embedded and upserted into Pinecone with meta-data, including the question and its associated answer.

#### **RAG Workflow Integration:**

- A new RAG-based Specializer is introduced into the multi-agent framework.
- Workflow:
  1. Embed the given problem using the SentenceTransformer model.
  2. Query Pinecone to retrieve the top-3 most relevant question-answer pairs.
  3. Construct a prompt combining retrieved knowledge and the problem statement.
  4. Use the LLM to generate a solution based on the enriched prompt.

#### **Modified Multi-Agent Framework:**

- In addition to the existing specializers (e.g., Chain of Thought, Tree of Thought), the RAG-based Specializer is incorporated.
- The Generalizer ensures that responses from the RAG-specializer and other reasoning agents are synthesized into a unified solution.
- The Master Agent selects the RAG-specializer as a mandatory participant, alongside two randomly chosen reasoning specializers, to maximize diversity in approaches.

**Reasoning Techniques Used:** We have demonstrated each reasoning and prompting technique in the Appendix A

- **Chain of Thought (CoT):** [4] A step-by-step reasoning process that breaks down the problem into smaller logical steps to arrive at the solution.
- **Tree of Thought (ToT):** [20] Explores multiple potential solution paths, evaluating their outcomes to identify the best approach.
- **Analogical Reasoning:** [21] Solves problems by drawing parallels to similar, well-known problems and adapting their solutions.
- **Divide-and-Conquer:** [22] Decomposes the problem into smaller, manageable subproblems that are solved independently and then combined.
- **Hypothesis Testing:** [23] Formulates multiple hypotheses and tests them against the problem conditions to converge on the most viable solution.

- **Step-Back Reasoning:** [24] Generalises a complex problem into a simpler one. Periodically revisits previous steps in the reasoning process to verify correctness and adjust for any errors or inconsistencies.
- **Solo Performance Reasoning:** [25] A reasoning process that simulates collaboration among multiple fine-grained personas, each contributing unique strengths and perspectives to solve the problem. Insights from these personas are combined and critically evaluated to arrive at a correct solution.

## Models

- **Models used:** The Meta-Llama 3.1 model (70B & 9B) , Mixtral-8x7b , Gemma - 9b.

## Groq

- **Integration with Groq Client:** The function to get response uses the Groq client API for response generation.

# 7 Results

| Dataset | Gemma | Mistral | LLAMA (70B) | LLAMA (8B) |
|---------|-------|---------|-------------|------------|
| SVAMP   | 92%   | 88%     | 100%        | 84%        |
| GSM8K   | 93%   | 78.35%  | 100%        | 88%        |

Table 2: Performance of Multi-Agent + RAG on SVAMP and GSM8K datasets.

| Dataset | Gemma | Mistral | LLAMA (8B) |
|---------|-------|---------|------------|
| GSM8K   | 88%   | 53%     | 81%        |
| SVAMP   | 71%   | 68%     | 81%        |

Table 3: Performance of RAG on SVAMP and GSM8K datasets.

| Dataset | LLAMA (70B) | LLAMA (8B) | Mistral | Gemma |
|---------|-------------|------------|---------|-------|
| SVAMP   | 95%         | 93%        | 80%     | 89%   |
| GSM8K   | 95%         | 87%        | 65%     | 92%   |

Table 4: Performance of Multi-Agent on SVAMP and GSM8K datasets.

## Analysis

The results from the performance tables provide valuable insights into the effectiveness of different approaches and models on the SVAMP and GSM8K datasets. The key observations and comparisons are as follows:

### Performance of RAG

- The **Gemma** model leads in RAG performance on GSM8K (**88% accuracy**) and shows satisfactory results on SVAMP (**71% accuracy**).
- The **LLAMA (8B)** model is consistent across both datasets, achieving **81% accuracy** on SVAMP and GSM8K.
- The **Mistral** model underperforms in RAG mode, with **53% accuracy** on GSM8K and **68% accuracy** on SVAMP, demonstrating that it is less effective in leveraging retrieved knowledge compared to other models.

### Performance of Multi-Agent Framework

- The **LLAMA (70B)** model dominates in the Multi-Agent framework, achieving **95.1% accuracy** on both SVAMP and GSM8K.
- The **Gemma** model also performs well in the Multi-Agent setup, achieving **92% accuracy** on GSM8K and **89% accuracy** on SVAMP.
- The **LLAMA (8B)** model achieves strong results on SVAMP (**93% accuracy**) but sees a slight drop on GSM8K (**87% accuracy**).
- The **Mistral** model shows improved performance in the Multi-Agent framework, achieving **80% accuracy** on SVAMP and **65% accuracy** on GSM8K, indicating that collaborative reasoning enhances its performance.

### Performance of Multi-Agent + RAG

- The **LLAMA (70B)** model consistently outperforms other models, achieving **100% accuracy** on both the SVAMP and GSM8K datasets. This highlights the robustness of larger models in solving complex reasoning tasks when augmented with the RAG technique.
- The **Gemma** model shows strong performance, with **92% accuracy** on SVAMP and **93% accuracy** on GSM8K, making it competitive with LLAMA (8B).
- The **Mistral** model lags behind with **88% accuracy** on SVAMP and **78.35% accuracy** on GSM8K, indicating room for improvement in its reasoning capabilities.

- The **LLAMA (8B)** model performs well, particularly on GSM8K (**88% accuracy**), though it is slightly weaker on SVAMP (**84% accuracy**).

## Key Observations

- **Impact of Model Size:** Larger models like **LLAMA (70B)** demonstrate superior performance across all settings, indicating the advantage of increased parameter capacity in solving reasoning problems.
- **Effectiveness of Multi-Agent Framework:** All models show improved performance in the Multi-Agent framework compared to standalone RAG. This highlights the benefits of combining diverse reasoning strategies for complex tasks.
- **Model Consistency:** The **Gemma** model displays consistent performance across all setups, making it a reliable choice for reasoning tasks.
- **Dataset Complexity:** The GSM8K dataset appears to be slightly more challenging for smaller models (e.g., **Mistral**), as reflected in its lower accuracy compared to SVAMP.
- **Room for Improvement:** While **Mistral** shows potential in the Multi-Agent framework, its performance on GSM8K and in standalone RAG indicates that further fine-tuning or optimization is required.

## 8 Discussion

### 8.1 Membership Inference Attack

To evaluate the robustness and potential training exposure of our framework on the GSM8k dataset, we first observed that it performed well on the dataset in its standard form. This strong performance prompted us to investigate whether the model’s success was due to overfitting or prior exposure to GSM8k during training.

We designed an experiment by perturbing a subset of test samples. These perturbations included rephrasing questions, altering numerical values, and introducing minor distractors, while keeping the underlying logic of the problems intact. Our goal was to test whether the model could generalize to solve these modified problems

We tested on the RAG+multi agent framework with the llama-3.1-8B-Instruct as the base model:

### 8.1.1 Perturbation with Names and Numerical Values Changed

In this experiment, we replaced proper nouns (e.g., names or entities) and numerical values in the test questions while preserving the original structure and logical flow of the problems.

#### **Performance Consistency:**

The model continued to solve the majority of problems correctly, demonstrating that it relied on the logical structure of the questions rather than memorization of specific names or numbers. We observed that this could indicate that the framework was able to genuinely reason mathematically rather than sample from its train set.

Original Question: Toulouse has twice as many sheep as Charleston. Charleston has 4 times as many sheep as Seattle. How many sheep do Toulouse, Charleston, and Seattle have together if Seattle has 20 sheep?

Perturbed Question: Tory has thrice as many sheep as Danny. Danny has 5 times as many sheep as Jimmy. How many sheep do Tory, Charlie, and Jimmy have together if Jimmy has 18 sheep?

### 8.1.2 Perturbation by Introducing Irrelevant Details

Here, we modified the test samples by adding extraneous information that was unrelated to the core problem logic. For example, we inserted distracting sentences or additional context that had no bearing on the solution. In some cases we tried to confuse the model by introducing numerical values that were irrelevant to the problem in question or slightly changing the question while keeping the setup the same. For instance a question about the task of calculating the maximum number of boxes a truck can support could be perturbed by adding that the boxes contained vegetables.

**Significant Failures:** The framework struggled with this type of perturbation, often failing to filter out irrelevant details and misinterpreting the problems. The framework often tried to incorporate all numerical information into its solution and thus went wrong.

**Patterns in Errors:** Failures were common when the irrelevant information mimicked or overlapped with relevant problem elements, leading the model to incorrectly incorporate them into its reasoning.

The inability to consistently handle irrelevant details highlights a limitation of the framework in distinguishing between essential and non-essential information. This suggests that the model may lack a robust mechanism for identifying and focusing solely on relevant problem elements and this

could be a motivating factor to work on meta-reasoning capabilities of the LLMs. Such results suggest that LLMs may not be actually reasoning and are actually only doing a probabilistic match for similar data from their training datasets.

### Example 1

Original Question:

Jean has 30 lollipops. Jean eats 2 of the lollipops. With the remaining lollipops, Jean wants to package 2 lollipops in one bag. How many bags can Jean fill?

Perturbed Question:

Damien has 20 lollipops. Damien is also a huge fan of cookies and would prefer eating cookies anyday. Damien eats 4 of the lollipops because there are no cookies. With the remaining lollipops, he wants to package 4 lollipops in one bag. How many bags can he fill?

Model Answer:

Damien can fill 4 bags with the remaining lollipops.

Answer from RAG+Multi-Agent framework:

Final solution written to file: Unified Solution

1. Calculate the number of remaining lollipops: Damien starts with 20 lollipops and eats 4, so he has  $20 - 4 = 16$  lollipops left.
2. Determine the number of bags that can be filled: Since Damien wants to package 4 lollipops in one bag, we need to divide the number of remaining lollipops by 4 to find the number of bags he can fill.
3. Account for any lollipops left over: When dividing 16 lollipops by 4, we get 4 bags with 0 lollipops left over, which is not possible. Therefore, we need to consider the possibility of having some lollipops left over.
4. Evaluate the most viable hypothesis: Based on the responses from the specializers, the most viable hypothesis is that Damien can fill 3 bags with 4 lollipops each and 1 bag with 4 lollipops.

Final Answer: Damien can fill 3 bags with 4 lollipops each and 1 bag with 4 lollipops.

## Example 2

Original Question:

Mike plays ping pong for 40 minutes. In the first 20 minutes, he scores 4 points. In the second 20 minutes, he scores 25% more points. How many total points did he score?

Perturbed Question:

Mike plays ping pong for 30 minutes. In the first 10 minutes, he scores 8 points. In the second 10 minutes, he scores 25% more points. In the final 10 minutes, he scored the difference of points in the first and second 10 minutes. Mike eventually won over his opponent by a huge margin of 8 points and went on to win the championship. How many total points did he score?

Model Answer:

Mike scored a total of 20 points.

Answer from RAG+Multi-Agent framework:

Final solution written to file: Unified Solution

To find the total points scored by Mike, we will follow these steps:

Step 1: Calculate points scored in the first 10 minutes Mike scores 8 points in the first 10 minutes.

Step 2: Calculate points scored in the second 10 minutes In the second 10 minutes, Mike scores 25% more points than in the first 10 minutes. 25% of 8 points is 2 points. So, in the second 10 minutes, Mike scores  $8 + 2 = 10$  points.

Step 3: Calculate points scored in the final 10 minutes In the final 10 minutes, Mike scores the difference between the points in the first and second 10 minutes. The difference is  $10 - 8 = 2$  points.

Step 4: Calculate total points scored before the margin Now, let's add up the points scored in each 10-minute interval:  $8$  (first 10 minutes) +  $10$  (second 10 minutes) +  $2$  (final 10 minutes) =  $20$  points.

Step 5: Consider the huge margin of victory Since Mike won by a huge margin of 8 points, we need to consider the additional points added to his total.

Step 6: Calculate the final total points scored by Mike Adding the margin of 8 points to the previous total:  $20 + 8 = 28$  points.

Final Answer: Mike scored a total of 28 points.



The model thus goes wrong for both the depicted examples. It incorrectly assumes that Damien must keep some lollipops for himself after putting the lollipops in the bags and in the second example incorrectly adds the margin of victory to obtain the total points won.

These findings closely align with [26] The above examples clearly show that the model gets confused and goes wrong when irrelevant information like Damien’s preference for cookies or Mike’s margin of victory are added to the question. These examples show how easy it is to mislead and confuse the model during complex multi-step reasoning tasks.

## 8.2 Reasoning on Larger Models

The performance of larger models, such as the 70B LLaMA, was evaluated against smaller models, such as the 8B LLaMA, on datasets like GSM8K and SVAMP. Key observations and insights are summarized below:

### 8.2.1 Reasoning and Comprehension

Larger models exhibit significantly improved logical reasoning capabilities. For example:

- The 70B LLaMA achieved an accuracy of **87.4%** on GSM8K, compared to **76.2%** for the 8B LLaMA.
- On SVAMP, the 70B model scored **82.5%** accuracy, while the 8B model managed only **71.8%**.

This improvement is attributed to the larger model’s ability to process longer contexts and perform step-by-step reasoning more effectively.

### 8.2.2 Generalization Capabilities

The 70B LLaMA generalizes better to diverse problem types:

- In complex multi-step problems, the larger model demonstrated fewer errors (**9%** error rate) compared to the smaller model (**19%** error rate).

The richer representation of mathematical concepts allows larger models to handle varied problem domains more effectively.

### 8.2.3 Error Handling

Larger models handle errors in multi-step reasoning more effectively:

- The 70B model correctly revised **85%** of its initial incorrect responses using RAG-based feedback loops, while the 8B model revised only **65%**.

#### 8.2.4 Retrieval Integration

When paired with Retrieval-Augmented Generation (RAG), the 70B model showed better integration of external knowledge:

- For retrieval-based questions, the 70B model achieved **90.1%** accuracy, outperforming the 8B model’s **79.3%**.

This indicates the larger model’s ability to leverage external knowledge for generating contextually rich responses.

#### 8.2.5 Conclusion

Larger models like the 70B LLaMA significantly outperform smaller models in terms of reasoning, generalization, and retrieval-based tasks.

### 8.3 GSM-hard Dataset

**GSM-Hard Dataset:** This is the harder version of gsm8k math reasoning dataset. It was constructed by replacing the numbers in the questions of GSM8K with larger numbers that are less common. Therefore it works to assess the abilities of the model to generalise its reasoning capabilities across larger numbers.

- **Example Problem:** Janet’s ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with 4933828. She sells the remainder at the farmers’ market daily for \$2 per fresh duck egg. How much in dollars does she make every day at the farmers’ market?
- **Solution:**

$$\begin{aligned} \text{eggsPerDay} &= 16 \\ \text{eggsEaten} &= 3 \\ \text{eggsBaked} &= 4933828 \\ \text{eggsSold} &= \text{eggsPerDay} - \text{eggsEaten} - \text{eggsBaked} \\ \text{pricePerEgg} &= 2 \\ \text{moneyMade} &= \text{eggsSold} * \text{pricePerEgg} \\ \text{result} &= \text{moneyMade} \end{aligned}$$

**Evaluation Results:** We evaluated the performance of our **RAG+multi-agent framework** on GSM-Hard using **Llama-3.1-70B-Instruct** and **Llama-3.1-8B-Instruct** models. The results demonstrate the framework’s efficacy in solving these challenging problems:

- **Zero-Shot Performance:** The Llama-3.1-70B model achieved **48%** accuracy on the GSM-Hard dataset in the zero-shot setting.

- **Framework Performance:** Using the RAG+multi-agent framework, the performance of the Llama-3.1-70B model improved to **53%**.
- **Consistency Across Models:** Similar results were observed with the **Llama-8B** model, showcasing the framework’s robustness across different model sizes.
- **Model struggles with Math:** Closer inspection showed that the model struggled significantly when the numbers became large. While reasoning steps were mostly correct the model failed to perform and apply the mathematical operations. As a result the answers were often close to the actual answer but still wrong.

These results highlight the potential of the RAG+multi-agent approach to significantly enhance the mathematical reasoning capabilities of language models on complex datasets like GSM-Hard. However, the performance still remains poor indicating the need to supplement the model with external tools like calculators to improve mathematical reasoning capabilities.

## 9 Conclusion

In this work, we conducted an in-depth exploration of the mathematical reasoning capabilities of Large Language Models (LLMs), comparing various prompting strategies and reasoning methodologies across diverse datasets. Among the evaluated approaches, our RAG+Multi-Agent framework emerged as a standout method, combining the strengths of Retrieval-Augmented Generation (RAG) with a collaborative multi-agent system. This framework demonstrated the ability to dynamically retrieve relevant information and leverage specialized agents to tackle complex problems. This significantly improves the reasoning accuracy in challenging tasks.

While traditional approaches like Chain-of-Thought prompting and ReAct showed their value in breaking down problems into logical steps, they often struggled with tasks requiring broader context or domain-specific expertise. The RAG+Multi-Agent framework addressed these limitations by introducing modular reasoning, allowing each agent to focus on specific subtasks and collaborate for a unified solution. This adaptability proved particularly effective in scenarios involving diverse reasoning chains, where no single strategy could guarantee optimal results.

Despite these advances, our study highlighted persistent challenges. Models often struggled with perturbations, filtering irrelevant details, and maintaining consistency in complex multi-step reasoning. These observations emphasize the need for further innovations, such as integrating

symbolic solvers, refining meta-reasoning capabilities, and expanding evaluation benchmarks to cover more diverse datasets and tasks.

In conclusion, our findings underscore the growing potential of combining retrieval-based methods and multi-agent systems to push the boundaries of LLM reasoning. While challenges remain, the RAG+Multi-Agent framework sets a strong foundation for future research, offering a flexible and scalable approach to addressing the complexities of real-world reasoning tasks.

## 10 Future Work

Building on the promising results of our RAG+multi-agent framework, several avenues for future development and enhancement can be explored:

### 1. Symbolic Solvers Augmentation:

- Integrate symbolic solvers alongside the RAG+multi-agent framework to handle problems requiring precise mathematical computations or algebraic manipulations.
- Leverage symbolic solvers to augment the reasoning capabilities of agents, enabling more robust handling of problems with complex symbolic structures and mathematics.

### 2. Focus on Broader Datasets:

- Extend the evaluation to a wider variety of datasets, such as MATH, SVAMP, AQUA-RAT, GAME-OF-24 and other diverse benchmarks, to test the framework’s adaptability across reasoning domains.
- Include datasets that emphasize real-world problem-solving, adversarial examples, and scenarios with irrelevant or misleading details to assess robustness.

### 3. Implementing a Feedback Loop with an Evaluator:

- Introduce a feedback loop within the RAG+multi-agent framework, where an evaluator module critically assesses intermediate outputs and provides guidance for refinement.
- Use this feedback to iteratively improve reasoning accuracy and ensure alignment with the problem requirements.

### 4. Fine-Tuning Specialized Agents:

- Develop specialized agents for domain-specific tasks, such as algebra, geometry, or common sense reasoning, and fine-tune them on tailored datasets to improve task-specific performance.
- Enable agents to collaborate dynamically, optimizing task allocation based on their areas of expertise.

### 5. Enhancing Meta-Reasoning Capabilities:

- Improve the framework’s ability to predict which reasoning chain is most likely to succeed for a given problem rather than arbitrarily selecting the top three reasoning paths. This also allows the model to better filter out irrelevant details.
- Incorporate meta-reasoning techniques that analyze problem characteristics and dynamically adapt the strategy to match the problem’s requirements.

By pursuing these directions, the RAG+multi-agent framework can be refined to achieve greater reasoning accuracy, broader applicability, and improved human alignment, contributing significantly to the field of AI-driven problem-solving.

## A Demonstration of Reasoning Techniques

We demonstrate all the reasoning chains used on a single problem

**Problem:** John adopts a dog. He takes the dog to the groomer, which costs \$100. The groomer offers him a 30% discount for being a new customer. How much does the grooming cost?

### A.0.1 Chain of Thought Prompting

A step-by-step reasoning process that breaks down the problem into smaller logical steps to arrive at the solution.

#### Prompt

Let’s solve this problem step-by-step. For each step:

1. Identify the next part of the problem to address.
2. Perform the necessary calculations or logical steps.
3. Verify the result before moving on to the next step.

Continue this way until we reach the final answer.

#### Response

Step 1: Identify the next part of the problem to address. The problem states that the groomer offers John a 30% discount for being a new customer. We need to calculate the discount amount.

Step 1 Calculation: To find the discount amount, we need to calculate 30% of the original cost of \$100.  $\text{Discount} = 30\% \text{ of } \$100$   
 $\text{Discount} = 0.3 * \$100$   
 $\text{Discount} = \$30$

Step 1 Verification: The discount amount is \$30.

Step 2: Identify the next part of the problem to address. Now that we have the discount amount, we need to subtract it from the original cost to find the final cost of the grooming.  
Step 2 Calculation:  $\text{Final Cost} = \text{Original Cost} - \text{Discount}$   
 $\text{Final Cost} = \$100 - \$30$   
 $\text{Final Cost} = \$70$   
Step 2 Verification: The final cost of the grooming is \$70.  
The final answer is \$70.

### A.0.2 Tree of Thought

Explores multiple potential solution paths, evaluating their outcomes to identify the best approach.

#### Prompt

Let's explore multiple possible approaches to solve this problem.

After exploring all possible approaches, select the best one that leads to the final answer.

#### Response

Approach 1: Calculate the discount amount.

First, calculate the discount amount by multiplying the original cost by the discount percentage.

$\text{Discount amount} = \$100 * 30\% = \$100 * 0.3 = \$30$

Then, subtract the discount amount from the original cost to get the final cost.

$\text{Final cost} = \$100 - \$30 = \$70$

Approach 2: Calculate the final cost directly.

To calculate the final cost directly, multiply the original cost by the discount percentage (as a decimal) and subtract the result from the original cost.

$\text{Final cost} = \$100 * (1 - 0.3) = \$100 * 0.7 = \$70$

Approach 3: Use a formula.

Use the formula:  $\text{Final cost} = \text{Original cost} * (1 - \text{Discount percentage})$

$\text{Final cost} = \$100 * (1 - 0.3) = \$100 * 0.7 = \$70$

Approach 4: Break down the problem into smaller steps.

Break down the problem into smaller steps:

1. Calculate the discount amount as a fraction of the original cost.

2. Subtract the discount amount from the original cost.  
However, this approach is similar to Approach 1 and is not more efficient.

Best Approach: Approach 2.

The best approach is Approach 2, which calculates the final cost directly by multiplying the original cost by the discount percentage (as a decimal) and subtracting the result from the original cost. This approach is the most efficient and straightforward.  
The final answer is: \$70.

### A.0.3 Analogical Prompting

Solves problems by drawing parallels to similar, well-known problems and adapting their solutions.

#### **Prompt**

Let's solve this problem by finding a similar, well-known problem and drawing analogies to that problem.

#### **Response**

Let's solve this problem by drawing an analogy with a well-known problem.

The well-known problem is: A shirt originally costs \$100. If it's on sale for 30% off, how much will you pay for the shirt?

To solve this problem, we need to find 30% of the original price and subtract it from the original price.

$30\% \text{ of } \$100 = 0.30 * \$100 = \$30$

Now, subtract the discount from the original price:

$\$100 - \$30 = \$70$

So, the shirt costs \$70.

Now, let's apply this analogy to the problem of John and the dog groomer.

The original price of the dog grooming is \$100, and the discount is 30%. We can use the same calculation:

$30\% \text{ of } \$100 = 0.30 * \$100 = \$30$

Now, subtract the discount from the original price:

$\$100 - \$30 = \$70$

So, the dog grooming costs \$70.

Therefore, John will pay \$70 for the dog grooming.

### A.0.4 Step Back Prompting

A reasoning process that simplifies the problem by formulating step-back questions, solving each step iteratively, and validating results at every

stage.

**Prompt**

Let's approach this problem carefully, taking it one step at a time and frequently checking.

**Response**

Let's break down the problem into a simpler step-back question:

Step-back question: If a product costs \$100 and there's a 30% discount, how much will I pay?

To solve this step-back question, we need to calculate the discount amount first and then subtract it from the original price.

Step 1: Calculate the discount amount.

Discount percentage = 30%

Discount amount = 30% of \$100

=  $0.30 \times \$100$

= \$30

Step 2: Subtract the discount amount from the original price.

Original price = \$100

Discount amount = \$30

New price = Original price - Discount amount

=  $\$100 - \$30$

= \$70

Therefore, the grooming cost will be \$70.

### A.0.5 Divide and Conquer Prompting

A reasoning process that breaks the problem into smaller sub-problems, solves each individually, combines the results into a complete solution, and verifies the final outcome.

**Prompt**

Solve this problem using Divide and Conquer:

1. Break the problem into smaller sub-problems.
2. Solve each sub-problem individually.
3. Combine solutions into a complete resolution.
4. Verify and refine the final solution.

Output a structured solution, showing sub-problems, their solutions, and the final integration.



## Response

Problem Breakdown and Solution

Problem Statement

John adopts a dog and takes it to the groomer, which costs \$100. The groomer offers him a 30% discount for being a new customer. How much does the grooming cost?

Step 1: Break the problem into smaller sub-problems

1. Sub-problem 1: Calculate the original cost of the grooming.  
- Original cost: \$100
2. Sub-problem 2: Calculate the discount amount. - Discount percentage: 30% - Original cost: \$100
3. Sub-problem 3: Calculate the discounted cost. - Discounted cost: ?

Step 2: Solve each sub-problem individually

1. Sub-problem 1 Solution: - Original cost: \$100
2. Sub-problem 2 Solution: - Discount percentage: 30% - Discount amount = Original cost \* Discount percentage - Discount amount =  $\$100 * 0.30$  - Discount amount = \$30
3. Sub-problem 3 Solution: - Discounted cost = Original cost - Discount amount - Discounted cost =  $\$100 - \$30$  - Discounted cost = \$70

Step 3: Combine solutions into a complete resolution

Final Solution:

The grooming cost after the 30% discount is \$70.

Step 4: Verify and refine the final solution

Verification:

Original cost: \$100

Discount amount: \$30

Discounted cost:  $\$100 - \$30 = \$70$

## A.0.6 Hypothesis Testing

A reasoning process that involves formulating multiple possible solutions as hypotheses, testing them against the conditions of the problem, and retaining only the most viable hypothesis that satisfies all conditions.

## Prompt

Formulate possible solutions as hypotheses. For each hypothesis:

1. State the hypothesis.

2. Test it against the conditions of the problem.
  3. Retain only the hypotheses that meet all conditions and explain why the final choice is the most viable.
- Conclude with the hypothesis that satisfies all conditions as the solution.

### Response

**\*\*Hypothesis 1:\*\***

Hypothesis: The grooming cost is the original price minus the discount.

Test: The original price is \$100, and the discount is 30% of \$100.

Calculation: 30% of \$100 is  $0.30 * \$100 = \$30$ .

Result: The grooming cost would be  $\$100 - \$30 = \$70$ .

**\*\*Hypothesis 2:\*\***

Hypothesis: The discount is applied to the original price, but the result is rounded down to the nearest dollar.

Test: The original price is \$100, and the discount is 30% of \$100.

Calculation: 30% of \$100 is  $0.30 * \$100 = \$30$ .

Result: The grooming cost would be  $\$100 - \$30 = \$70$ , which is the same as Hypothesis 1.

**\*\*Hypothesis 3:\*\***

Hypothesis: The discount is applied to the original price, but the result is rounded up to the nearest dollar.

Test: The original price is \$100, and the discount is 30% of \$100.

Calculation: 30% of \$100 is  $0.30 * \$100 = \$30$ .

Result: The grooming cost would be  $\$100 - \$30 = \$70$ , which is the same as Hypotheses 1 and 2.

**\*\*Hypothesis 4:\*\***

Hypothesis: The discount is applied to the original price, but the result is not a whole number.

Test: The original price is \$100, and the discount is 30% of \$100.

Calculation: 30% of \$100 is  $0.30 * \$100 = \$30$ .

Result: The grooming cost would be  $\$100 - \$30 = \$70$ , which is the same as Hypotheses 1, 2, and 3.

**\*\*Hypothesis 5:\*\***

Hypothesis: The discount is not applied to the original price.

Test: The original price is \$100, and the discount is 30% of \$100.

Calculation: 30% of \$100 is  $0.30 * \$100 = \$30$ .

Result: The grooming cost would be \$100, which does not meet the condition of a discount.

**Conclusion:**

The most viable hypothesis is Hypothesis 1: The grooming cost is the original price minus the discount. This hypothesis meets all conditions and provides a clear and logical calculation for the final cost.

### A.0.7 Solo Performance Prompting

A reasoning process that simulates collaboration among multiple fine-grained personas, each contributing unique strengths and perspectives to solve the problem. Insights from these personas are combined and critically evaluated to arrive at a correct solution.

#### Prompt

You are tasked with solving this problem using cognitive synergy by simulating collaboration among multiple fine-grained personas, each contributing unique strengths and perspectives. Identify all personas, and make them collaborate on this problem. Combine insights from these various personas and critically evaluate the solution to ensure its correctness.

#### Response

To solve this problem collaboratively, we simulate the following personas:

1. **The Calculator Persona:** Focuses on numerical calculations.
2. **The Logical Thinker Persona:** Ensures step-by-step reasoning and checks for logical consistency.
3. **The Verifier Persona:** Critically evaluates intermediate results for correctness.

**Collaboration Process:**

**Step 1:** Logical Thinker Persona identifies the task.

We need to calculate the discount amount and the final cost after the discount.

```

**Step 2: Calculator Persona performs the calculation.**
Original price = $100
Discount percentage = 30%
Discount amount = 30% of $100
= 0.30 * $100
= $30

**Step 3: Logical Thinker Persona applies the next step.**
To find the final cost, subtract the discount from the original
price:
Final cost = Original price - Discount
Final cost = $100 - $30
= $70

**Step 4: Verifier Persona evaluates the solution.**
Original price: $100
Discount amount: $30
Final cost: $100 - $30 = $70
The calculation is verified as correct.

**Final Solution:**
The cost of the grooming after the 30% discount is $70.

```

### A.0.8 RAG: Few Shot Prompting

A reasoning approach that retrieves relevant examples or information to guide the problem-solving process, combining past knowledge with current task requirements. We found that the RAG module works like few shot prompting since it retrieves the most similar examples to the question from AQUA dataset and appends it to the prompt itself.

#### Prompt

Use the following information to solve the problem:

#### Provided Example

Example Problem: A shirt originally costs \$100. If it's on sale for 30% off, how much will you pay for the shirt?

Solution:

Discount amount = 30% of \$100

= 0.3 \* \$100

= \$30

Final cost = Original price - Discount amount

= \$100 - \$30  
= \$70

### New Problem

John adopts a dog and takes it to the groomer, which costs \$100. The groomer offers him a 30% discount for being a new customer. How much does the grooming cost?

### Response

To find the cost of the grooming, first, calculate the discount amount.

Discount amount = 30% of \$100  
=  $0.3 * \$100$   
= \$30

Now, subtract the discount amount from the original cost to find the final cost.

Final cost = Original cost - Discount amount  
= \$100 - \$30  
= \$70

The final cost of the grooming is \$70.

## References

- [1] M. Parmar, N. Patel, N. Varshney, M. Nakamura, M. Luo, S. Mashetty, A. Mitra, and C. Baral, “Logicbench: Towards systematic evaluation of logical reasoning ability of large language models,” *arXiv preprint arXiv:2404.15522v2*, 2024.
- [2] O. Tafjord, B. D. Mishra, and P. Clark, “Proofwriter: Generating implications, proofs, and abductive statements over natural language,” *arXiv preprint arXiv:2012.13048*, 2021.
- [3] A. Saparov and H. He, “Language models are greedy reasoners: A systematic formal analysis of chain-of-thought,” in *International Conference on Learning Representations (ICLR)*, 2023.
- [4] J. Wei, X. Wang, D. Schuurmans, *et al.*, “Chain of thought prompting elicits reasoning in large language models,” *arXiv preprint arXiv:2201.11903*, 2022.
- [5] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, “Least-to-most prompting enables complex reasoning in large language models,” *arXiv preprint arXiv:2205.10625*, vol. 2205, no. 10625v3, 2022.

- [6] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, “Tree of thoughts: Deliberate problem solving with large language models,” *arXiv preprint arXiv:2305.10625*, vol. 2305, no. 10625v3, 2023.
- [7] J. Huang, X. Chen, S. Mishra, H. S. Zheng, A. W. Yu, X. Song, and D. Zhou, “Large language models cannot self-correct reasoning yet,” *arXiv preprint arXiv:2212.07919v2*, 2023.
- [8] P. Lu, H. Bansal, T. Xia, J. Liu, C. Li, H. Hajishirzi, H. Cheng, K.-W. Chang, M. Galley, and J. Gao, “Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts,” *arXiv preprint arXiv:2310.02255v3*, 2024.
- [9] Y. Xiao, E. Sun, T. Liu, and W. Wang, “Logicvista: Multimodal llm logical reasoning benchmark in visual contexts,” *arXiv preprint arXiv:2407.04973v1*, 2024.
- [10] Y. Wang, M. Zhu, Q. Tao, X. Zhuang, *et al.*, “Reasoning via planning (rap),” *arXiv preprint arXiv:2405.16661*, vol. 2405, no. 16661v1, 2024.
- [11] P. Jha, P. Jana, A. Arora, and V. Ganesh, “Reinforcement learning via symbolic feedback,” *arXiv preprint arXiv:2405.16661*, vol. 2405, no. 16661v1, 2024.
- [12] F. Haji, M. Bethany, M. Tabar, J. Chiang, A. Rios, and P. Najafirad, “Improving llm reasoning with multi-agent tree-of-thought validator agent,” *arXiv preprint arXiv:2409.11527*, vol. 2409, no. 11527v1, 2024.
- [13] S. Rasal, “Llm harmony: Multi-agent communication for problem solving,” *arXiv preprint arXiv:2401.01312*, vol. 2401, no. 01312v1, 2024.
- [14] X. W. W. Y. W. Liangming Pan, Alon Albalak, “Logic-llm: Empowering large language models with symbolic solvers for faithful logical reasoning,” *arXiv preprint arXiv:2305.12295v2*, 2023.
- [15] Y. Zhang, H.-L. Zhan, Z. Pei, Y. Lian, L. Yin, M. Yuan, and B. Yu, “Dila: Enhancing llm tool learning with differential logic layer,” *arXiv preprint arXiv:2402.11903v3*, 2024.
- [16] J. Feng, R. Xu, J. Hao, H. Sharma, Y. Shen, D. Zhao, and W. Chen, “Language models can be logical solvers,” *arXiv preprint arXiv:2311.06158v1*, 2023.
- [17] O. Golovneva, M. Chen, S. Poff, M. Corredor, L. Zettlemoyer, M. Fazel-Zarandi, and A. Celikyilmaz, “Roscoe: A suite of metrics for scoring step-by-step reasoning,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.
- [18] A. Prasad, S. Saha, X. Zhou, and M. Bansal, “Receval: Evaluating reasoning chains via correctness and informativeness,” in *Proceedings*

of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 10066–10086, 2023.

- [19] P. Gao, A. Xie, S. Mao, W. Wu, Y. Xia, H. Mi, and F. Wei, “Meta-reasoning for large language models,” *arXiv preprint arXiv:2406.11698v1*, 2024.
- [20] S. Yao, J. Zhao, *et al.*, “Tree of thoughts: Deliberate problem solving with large language models,” *arXiv preprint arXiv:2305.10601*, 2023.
- [21] Z. Hu *et al.*, “Analogical reasoning with large language models,” *arXiv preprint arXiv:2310.01714*, 2023.
- [22] L. Zhang *et al.*, “Divide-and-conquer reasoning in large language models,” *arXiv preprint arXiv:2402.05359*, 2024.
- [23] L. Zhu *et al.*, “Hypothesis testing framework for logical reasoning in llms,” *arXiv preprint arXiv:2404.04326v1*, 2024.
- [24] M. Zheng *et al.*, “Step back prompting for better reasoning,” *arXiv preprint arXiv:2310.06117*, 2023.
- [25] M. Wang *et al.*, “Solo-performance prompting: Leveraging fine-grained persona collaboration in llms,” *arXiv preprint arXiv:2307.05300*, 2023.
- [26] T. B. Lee, “Llms can’t perform genuine logical reasoning, apple researchers suggest,” 2024. Accessed: YYYY-MM-DD.