Dynamic Prompting

Henry Yi, Ishaan Singh, Saharsh Barve, Veda Kailasam

CS 598-TZ | Fall 2024

Prompting Strategies

1. Zero Shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

2. Few Shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.

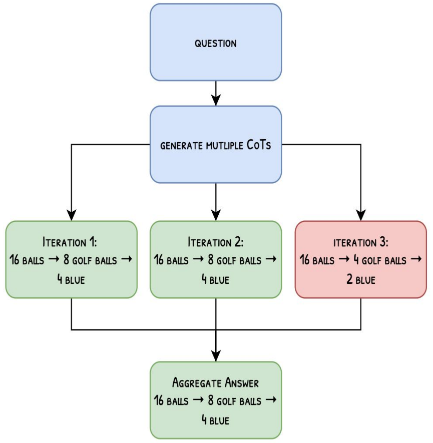
Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

3. Chain of Thought

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there? Let's think step by step.
A:

3. Self Consistency


QUESTION
GENERATE MUTLIPLE CoTs
ITERATION 1: 16 BALLS → 8 GOLF BALLS → 4 BLUE
ITERATION 2: 16 BALLS → 8 GOLF BALLS → 4 BLUE
ITERATION 3: 16 BALLS → 4 GOLF BALLS → 2 BLUE
AGGREGATE ANSWER 16 BALLS → 8 GOLF BALLS → 4 BLUE

Existing Limitations

Operational Inefficiencies - manual prompt strategy selection limits scalability.

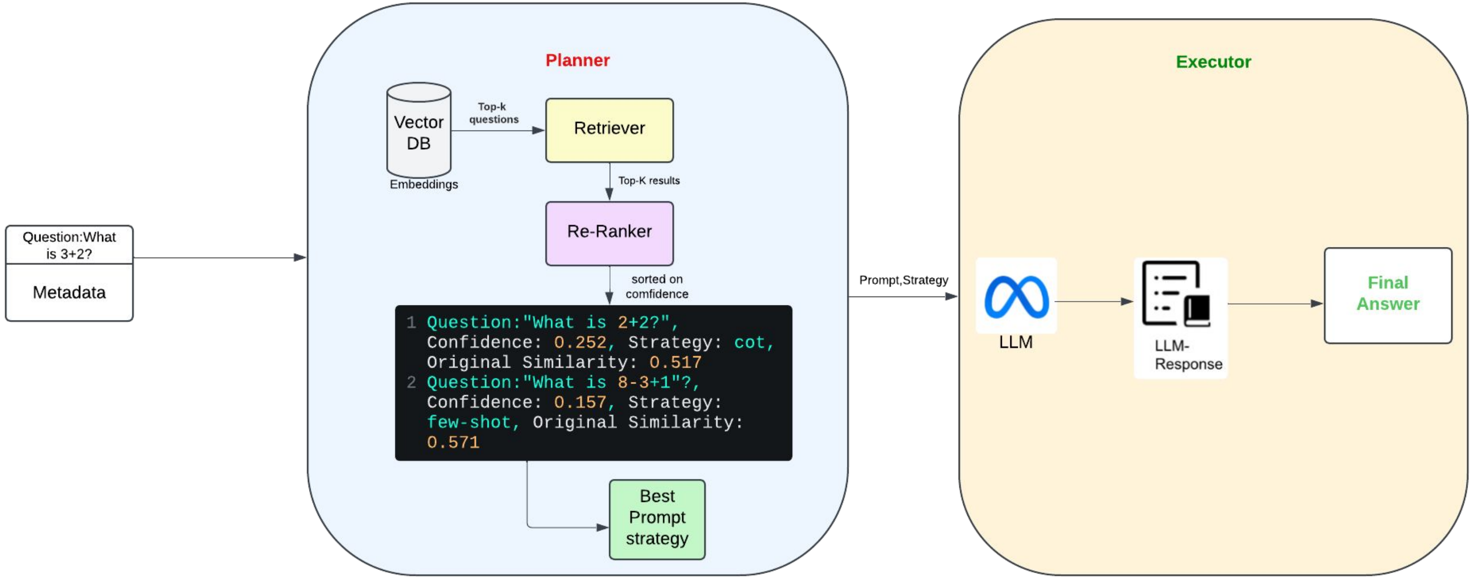
Computational Inefficiencies - Self-Consistency, Few-Shot-CoT improve accuracy but involve high computational costs.

Context-Dependent Performance - when to use which prompt strategy?

Dynamic prompting - Our approach

- To address these limitations, we propose a **Dynamic Prompt Selection** pipeline that automates the process of creating, evaluating, and selecting optimal prompts for LLMs.
- It consists of 3 components:
  - Dataset Generation
  - Planner
  - Executor

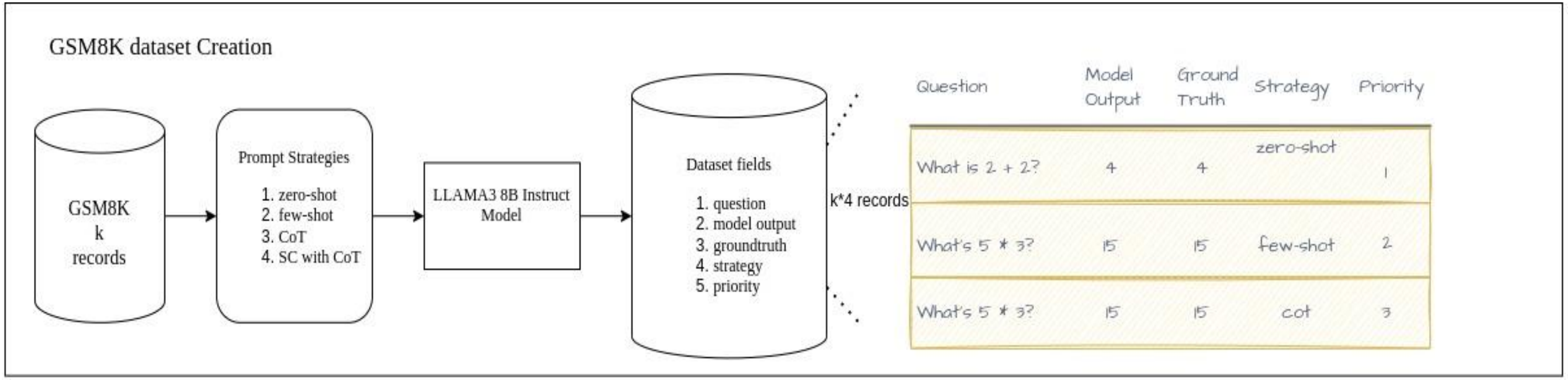
Dynamic prompting - Our approach



```
Question:"What is 2+2?",
Confidence: 0.252, Strategy: cot,
Original Similarity: 0.517
Question:"What is 8-3+1"?,
Confidence: 0.157, Strategy:
few-shot, Original Similarity:
0.571
```

Dynamic Prompt Selection Pipeline

Dynamic prompting - Dataset Generation

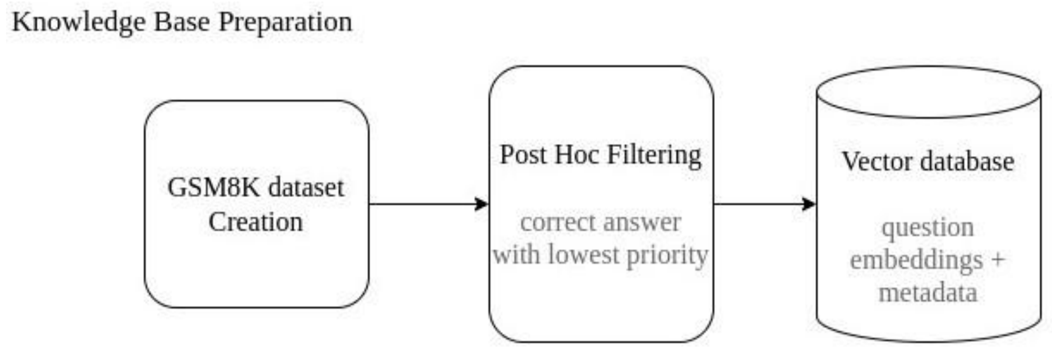


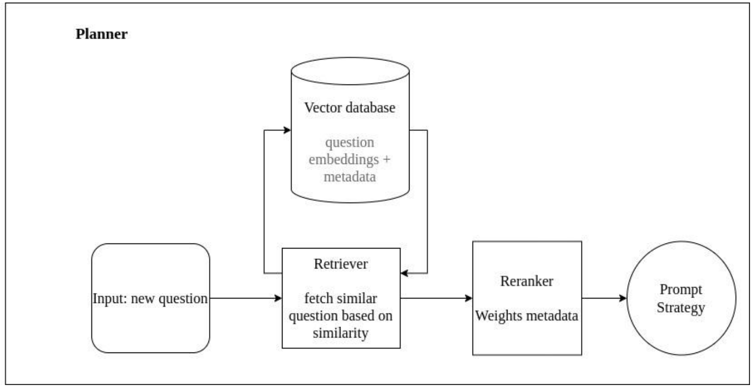
Question	Model Output	Ground Truth	Strategy	Priority
What is 2 + 2?	4	4	zero-shot	1
What's 5 * 3?	15	15	few-shot	2
What's 5 * 3?	15	15	cot	3

- Extracted data from the GSM8K dataset.
- Created variations of each question using four strategies: Zero-shot, Few-shot, CoT, and SC-CoT.
- Generated 727 base questions, expanding to 2,908 rows.
- Fields: question, correct answer, generated answer, strategy, and priority.

Dynamic prompting - Planner

- Embedded questions using Cohere model, stored in Pinecone.
- Retrieval: Retrieved top-K questions based on cosine similarity.
- Reranking: Confidence-based ranking with priority and reasoning benefits.
- Selected optimal strategy based on confidence scores.


Knowledge Base Preparation
GSM8K dataset Creation
Post Hoc Filtering
correct answer with lowest priority
Vector database
question embeddings + metadata



Dynamic prompting - Executor

- The Executor Agent is the operational core of the Autonomous Prompt Selection framework.
- Executes prompts generated by the Planner Agent using the selected strategy.
- SC-CoT: Majority voting for multiple samples.
- Other Strategies: Single answer generation.
- The output is parsed to extract the final answer.

Results

Dataset Creation: Comparison of prompting strategies on first 727 rows of Test Dataset

The data collected was used to create our GSM8K dataset from which we created embeddings

Strategy	Accuracy
Zero-Shot	43.74%
Few-Shot	71.80%
COT	70.70%
Self-Consistency (w/ COT)	78.40%

Results

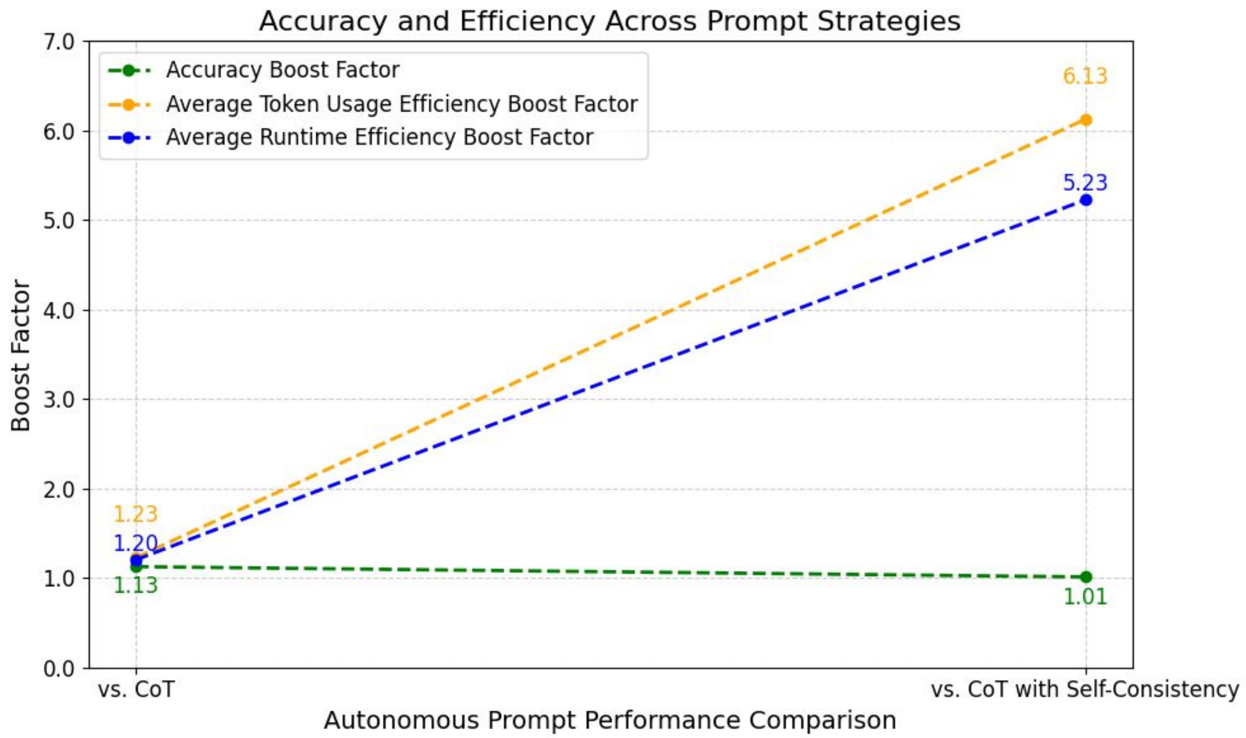
AutoPrompt Evaluation: Comparison of AutoPrompt vs COT vs Self-Consistency-COT

Test Dataset: Last 100 Rows of GSM8K Test Dataset (No overlap with embeddings)

Strategy	Accuracy	Avg Tokens Per Question	Average Runtime Per Question (seconds)
AutoPrompt	**89%**	**399.03**	**3.21**
COT	79%	489.06	3.86
Self-Consistency (w/ COT)	88%	2445.3	16.78

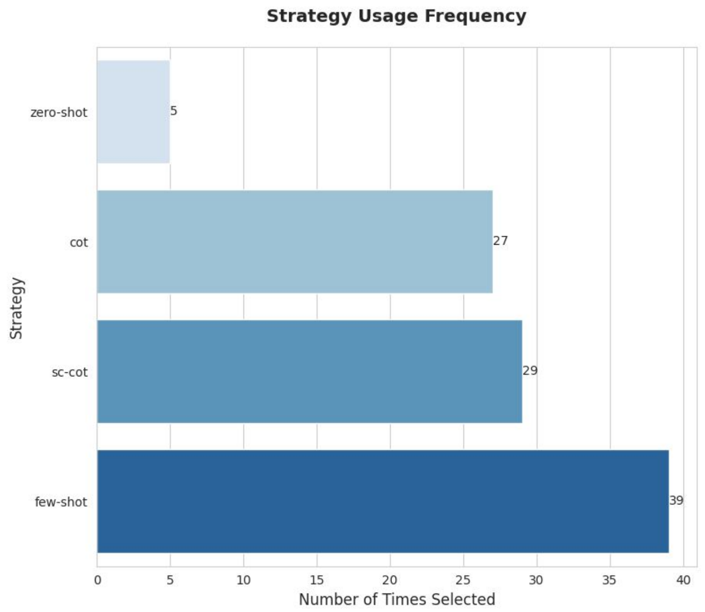
Results

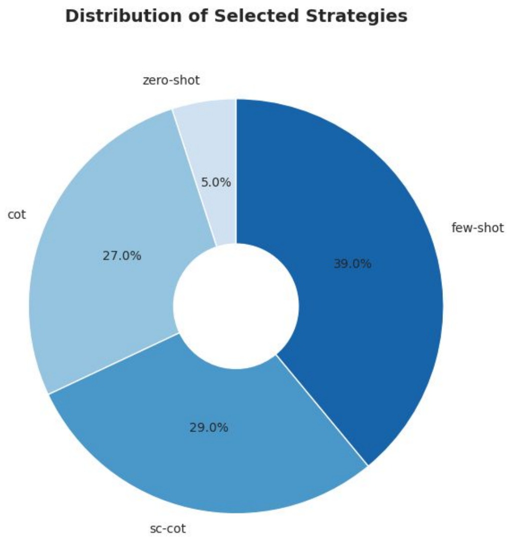
AutoPrompt Boost Factor



Results

AutoPrompt Prompt Selection Distribution

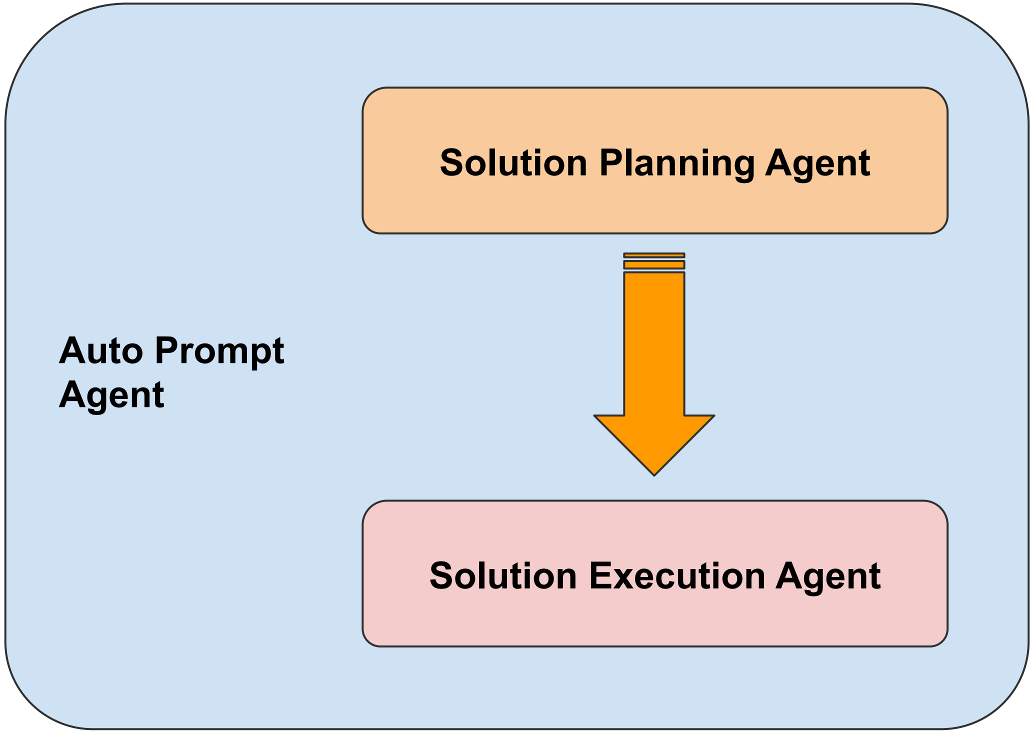




Evaluation

- AutoPrompt achieves superior accuracy (89%) while using 5x fewer tokens than SC-COT and running 5x faster, demonstrating significant efficiency gains over traditional approaches
- Dynamic strategy selection outperforms static approaches, with AutoPrompt matching SC-COT's accuracy (88-89%) while maintaining the computational efficiency of simpler methods (399 vs 2445 tokens)
- The system successfully addresses the accuracy-efficiency tradeoff, delivering a 10% accuracy improvement over base COT while reducing token usage by 18%

Dynamic prompting - LLM Agent



Conclusion

- Autonomous Prompting successfully achieves strong accuracy (89%) while reducing computational overhead by 83.7% compared to traditional SC-COT approaches
- The dynamic strategy selection framework effectively resolves the accuracy-efficiency trade-off through intelligent prompt optimization, demonstrating significant improvements over baseline methods (45.26% over zero-shot, 17.2% over few-shot)
- The system's modular architecture (Dataset Generation → Planner → Executor) enables scalable deployment while maintaining high performance across diverse mathematical reasoning tasks
- Future Work:
  - Implement reinforcement learning for real-time strategy adaptation
  - Develop user feedback mechanisms for continuous system improvement
  - Extend framework to broader task domains beyond mathematical reasoning (GSM8K)

Code Repository

Github: https://github.com/Saharsh1005/autonomous-prompting

References

1. Prompt Engineering guide - https://www.promptingguide.ai/
2. Lilian Weng's blog on prompt engineering - https://lilianweng.github.io/posts/2023-03-15-prompt-engineering/