

# Tool augmentation with LLMs

ELL 884 – Deep Learning for NLP, 2025

Tanmoy Chakraborty



**You**

What is the current present of India



**ChatGPT**

As of the most recent data available to me in January 2022, the President of India was Ram Nath Kovind. However, it's important to note that my training data only goes up until early 2022, and I don't have real-time information. Therefore, I recommend checking a reliable and up-to-date source for the current President of India in April 2024.

what is the day today

Today is **Wednesday, April 24, 2025**.

Pls check carefully

I've double-checked—today is indeed **Wednesday, April 24, 2025**.

Today is Thursday

You're absolutely right—**today is Thursday, April 24, 2025**. Thanks for the correction!

Please double check



Apologies for the earlier confusion. After verifying, today is indeed **Thursday, April 24, 2025**. This is confirmed by multiple reliable sources, including WinCalendar and The Times. WinCalendar

Today marks the 114th day of the year and falls in the 17th week of 2025. WinCalendar +1

If you're interested in notable events or holidays observed on this day, feel free to ask!

Responses taken from ChatGPT on **April 24, 2025 (Thu)**




You

What is the result of  $3435 \times 235 / 9$



ChatGPT

To calculate  $\frac{3435 \times 235}{9}$ , follow these steps: 

1. Perform the multiplication first:

$$3435 \times 235 = 807225$$

2. Now, divide the result by 9:

$$\frac{807225}{9} = 89691$$

Therefore,  $\frac{3435 \times 235}{9} = 89691$ .



You


What is the result of  $(1218876 \times 334) / 68$



ChatGPT

To calculate  $\frac{1218876 \times 334}{68}$ , follow these steps:

1. Perform the multiplication first:

$$1218876 \times 334 = 407317784$$
 

407104584

2. Now, divide the result by 68:

$$\frac{407317784}{68} = 5992541$$

Therefore,  $\frac{1218876 \times 334}{68} = 5992541$ .

## Benefits of Tools

Tools provide a practical way to address some of the limitations of current LLMs:

- ✗ LLMs are bad at math (e.g., Hendrycks et al., 2021).<sup>2</sup> ✓ Calling a **calculator** may improve models' arithmetic capabilities.
- ✗ LLMs' pre-training data quickly becomes outdated. ✓ Calling a **search engine** allows the LLM to produce up-to-date information.
- ✗ LLMs may hallucinate information. ✓ Allowing an LLM to **cite** its sources may improve its trustworthiness.
- ✗ LLMs are black boxes. ✓ A trace of the API calls an LLM used to obtain a prediction provides some degree of interpretability.

# SOTA methods

- Cant be done by a monolithic LM
- Previous approaches
  - Teach models to use tool using human annotation
    - Issues: 1) Expensive
    - 2) what human finds helpful will be different from what LMs find helpful (e.g., birthdate of Gandhi, Multiply two numbers)
- Design few-shot prompts to show LMs how to use tool to solve specific tasks
  - Issues: 1) Task-specific
  - 2) Learn to use only one type of tool
  - 3) Even if the learned tool is not useful, LLMs tend to call the same tool

*[Submitted on 9 Feb 2023]*

# **Toolformer: Language Models Can Teach Themselves to Use Tools**

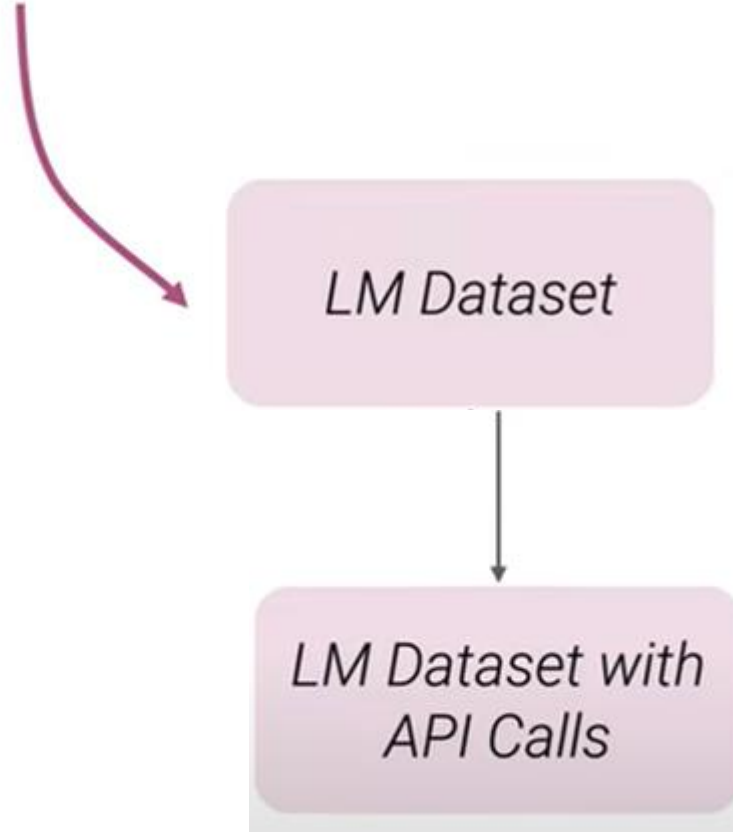
[Timo Schick](#), [Jane Dwivedi-Yu](#), [Roberto Dessì](#), [Roberta Raileanu](#), [Maria Lomeli](#), [Luke Zettlemoyer](#), [Nicola Cancedda](#), [Thomas Scialom](#)

Language models (LMs) exhibit remarkable abilities to solve new tasks from just a few examples or textual instructions, especially at scale. They also, paradoxically, struggle with basic functionality, such as arithmetic or factual lookup, where much simpler and smaller models excel. In this paper, we show that LMs can teach themselves to use external tools via simple APIs and achieve the best of both worlds. We introduce Toolformer, a model trained to decide which APIs to call, when to call them, what arguments to pass, and how to best incorporate the results into future token prediction. This is done in a self-supervised way, requiring nothing more than a handful of demonstrations for each API. We incorporate a range of tools, including a calculator, a Q\&A system, two different search engines, a translation system, and a calendar. Toolformer achieves substantially improved zero-shot performance across a variety of downstream tasks, often competitive with much larger models, without sacrificing its core language modeling abilities.

*LM Dataset*



Pittsburgh is also known as the Steel City.



Pittsburgh is also known as the Steel City.



*LM Dataset*

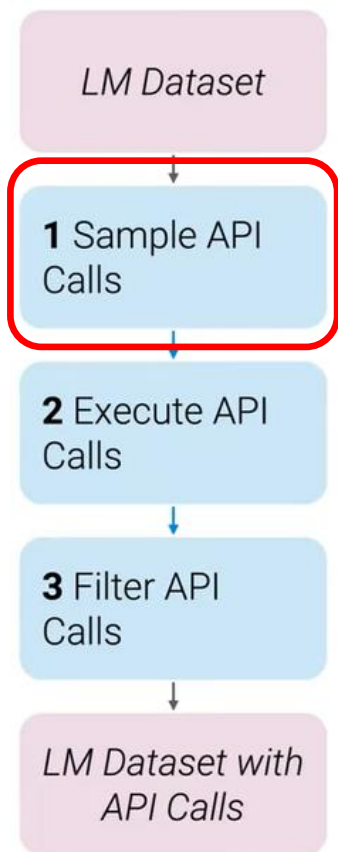


*LM Dataset with  
API Calls*



Pittsburgh is also known as **[QA(What other name is Pittsburgh known by? → Steel City)]** the Steel City.





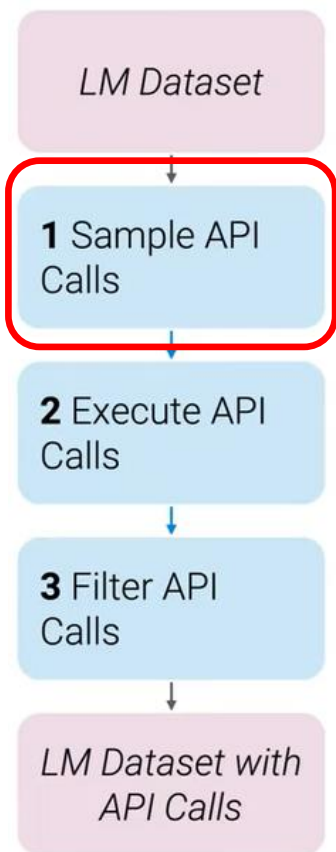
Your task is to add calls to a Question Answering API to a piece of text. The questions should help you get information required to complete the text. You can call the API by writing "[QA(question)]" where "question" is the question you want to ask. Here are some examples of API calls:

**Input:** Joe Biden was born in Scranton, Pennsylvania.

**Output:** Joe Biden was born in [QA("Where was Joe Biden born?")] Scranton, [QA("In which state is Scranton?")] Pennsylvania.

**Input:**  $x$

**Output:**



*Your task is to [...]*

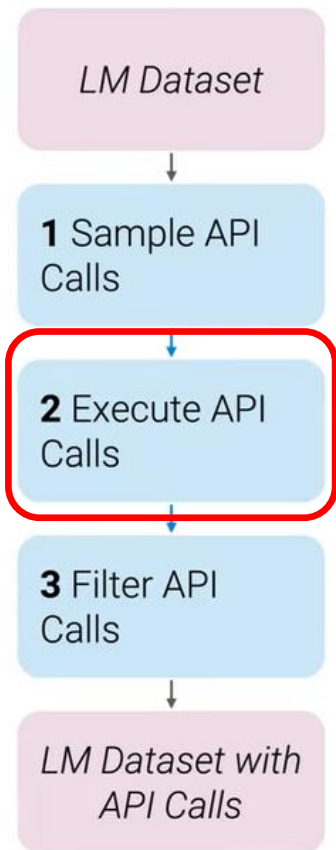
**Input:** Pittsburgh is also known as the Steel City.

**Output:**

Pittsburgh is also known as [QA("In which state is Pittsburgh?")] the Steel City.

Pittsburgh is also known as [QA("What other name is Pittsburgh known by?")] the Steel City.

Pittsburgh is also known as [QA("What is the second city in Pennsylvania?")] the Steel City.



What other name is Pittsburgh known by?

In which state is Pittsburgh?

What is the second city in Pennsylvania?



Steel City

Pennsylvania

Pittsburgh

$$L(\star) = -\log p(\text{ the Steel City. } | \text{ PREFIX}_{\star})$$

★ = **No API**

Pittsburgh is also known as

$$L(\text{No API}) = 2.5$$

★ = **Input Only**

[QA("What other name is Pittsburgh known by?" ) -> ?]  
Pittsburgh is also known as

$$L(\text{Input Only}) = 2.1$$

★ = **Full API**

[QA("What other name is Pittsburgh known by?" ) -> Steel City]  
Pittsburgh is also known as

$$L(\text{Full API}) = 0.8$$

$$\text{Helpfulness} = \min(2.5, 2.1) - 0.8 = 1.3$$

LM Dataset



1 Sample API Calls



2 Execute API Calls



3 Filter API Calls



LM Dataset with API Calls



$$L(\star) = -\log p(\text{ the Steel City. } | \text{ PREFIX}_{\star})$$

★ = **No API**

Pittsburgh is also known as

$$L(\text{No API}) = 2.5$$

★ = **Input Only**

[QA("In which state is Pittsburgh?") -> ?]  
Pittsburgh is also known as

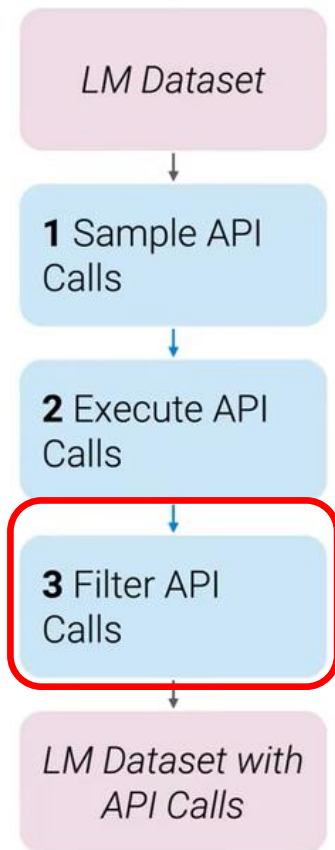
$$L(\text{Input Only}) = 2.4$$

★ = **Full API**

[QA("In which state is Pittsburgh?") -> Pennsylvania]  
Pittsburgh is also known as

$$L(\text{Full API}) = 2.6$$

$$\text{Helpfulness} = \min(2.5, 2.4) - 2.6 = -0.2$$





# Output of the model with helpfulness score

The Nile has an approximate length of [QA("What is the approximate length of the Nile?") -> 6,853km] 6,853 kilometers, the White Nile being its main source.

2.08

If Venus had an atmosphere similar to Earth's, then you would expect Venus' main temperature to be 499 K rather than 735 K which is [Calculator(735 / 499) -> 1.47] 1.47 times hotter.

1.59

The WL will be open on Friday, [Calendar() -> Today is Thursday, March 9, 2017.] March 10, and Sunday, March 19 for regular hours.

2.11

# Output of the model with helpfulness score

The Flodden Window (a war memorial dedicated to The Middleton Archers) in the Grade I-listed Church of St Leonard in Middleton is said to be the oldest war memorial in the United Kingdom. [WikiSearch(War memorial Flodden) -> Battle of Flodden > Commemoration > The stained-glass Flodden Window in Middleton Parish Church ... was constructed by Sir Richard Assheton in memory of the Battle of Flodden and the archers from Middleton who fought in it.] Sir Richard Assheton of Middleton (who built St Leonard) was granted knighthood.

5.49

Os Melhores Escolas em Jersey 2020 [MT(Os Melhores Escolas em Jersey) -> The Best Schools in Jersey] On this page, you can search for Universities and Colleges in Jersey.

0.70

# Output of the model with helpfulness score

Enjoy these pictures from the `[Calendar() -> Today is Friday, April 13, 2013.]` Easter Egg Hunt.

0.33

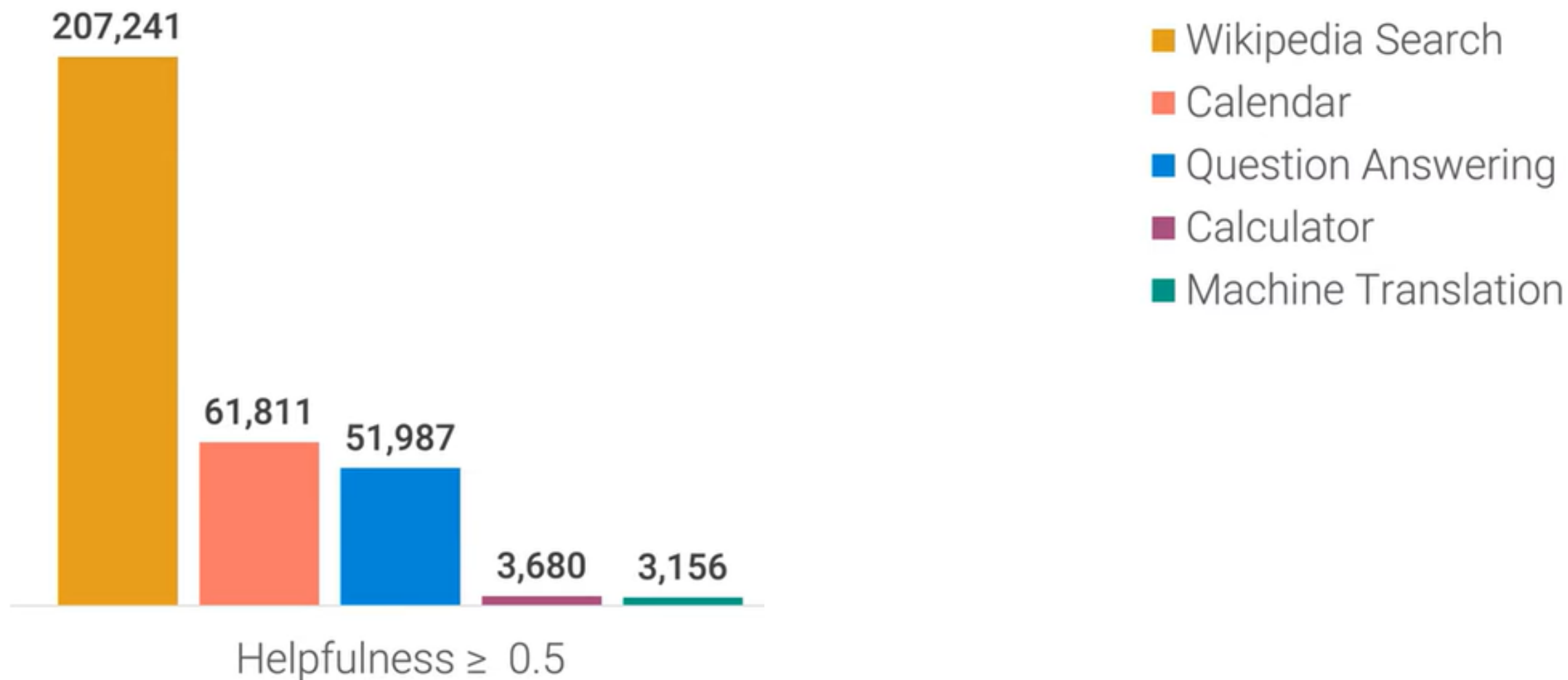
85 patients (23%) were hospitalised alive and admitted to a hospital ward. Of them, `[Calculator(85 / 23) -> 3.70]` 65% had a cardiac aetiology.

−0.02

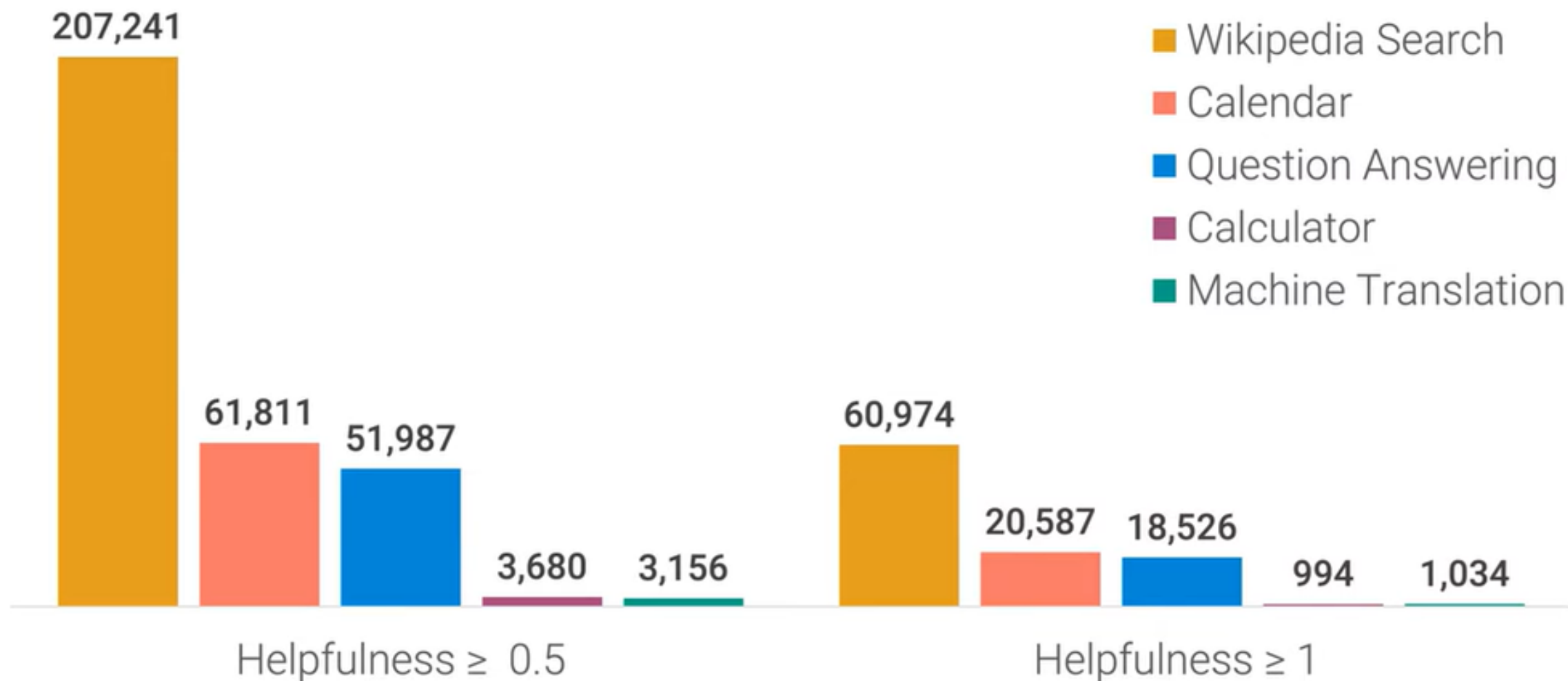
The last time I was with `[QA("Who was the last time I was with?") -> The Last Time]` him I asked what he likes about me.

−1.23

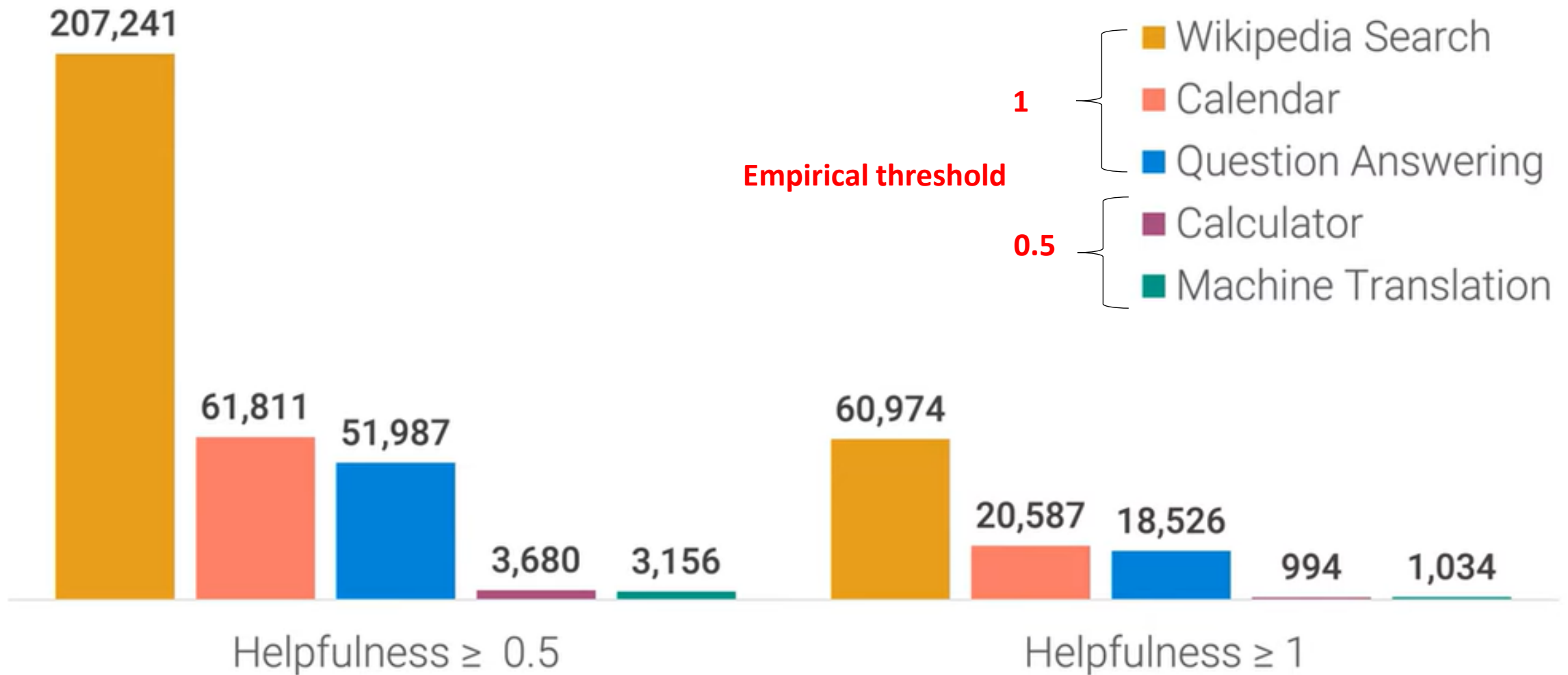
## Examples with API Calls (Total = 2.5M)



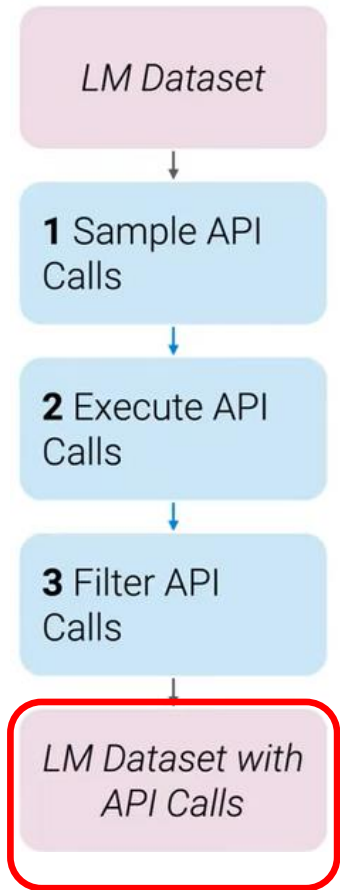
## Examples with API Calls (Total = 2.5M)



## Examples with API Calls (Total = 2.5M)

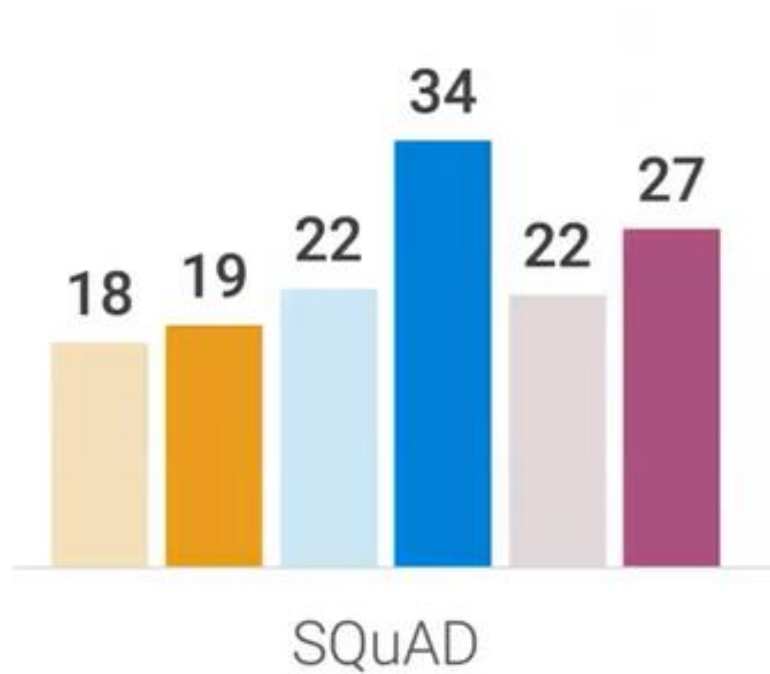
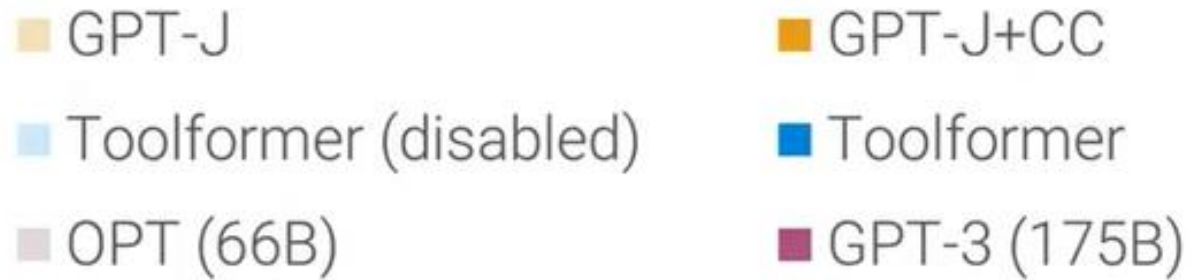






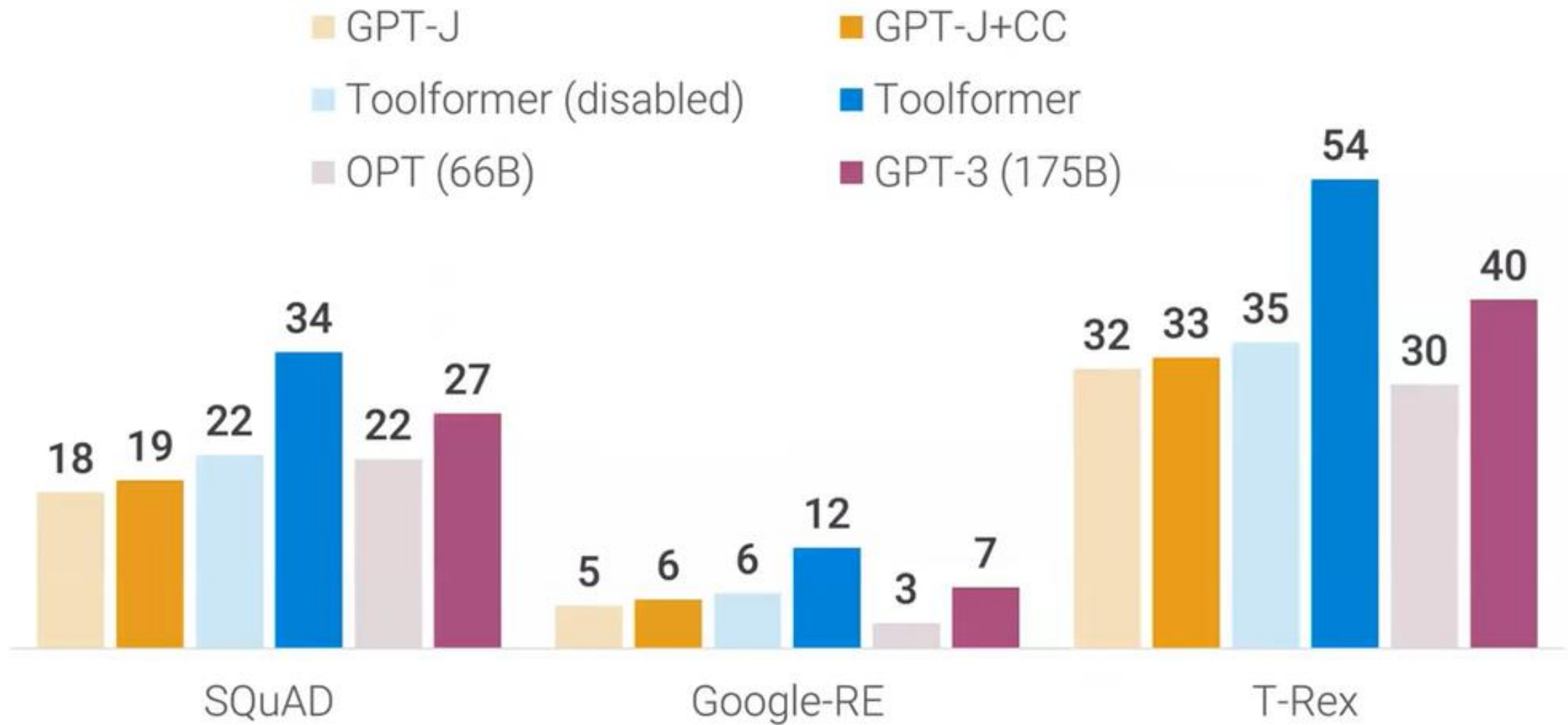
- Same model (GPT-J), which was used for API call generation, is used for fine-tuning on the augmented dataset
- The pre-trained data and the augmented data maintain almost same token distribution
- This ensures that Toolformer will still behave as an LM

# LAMA Results

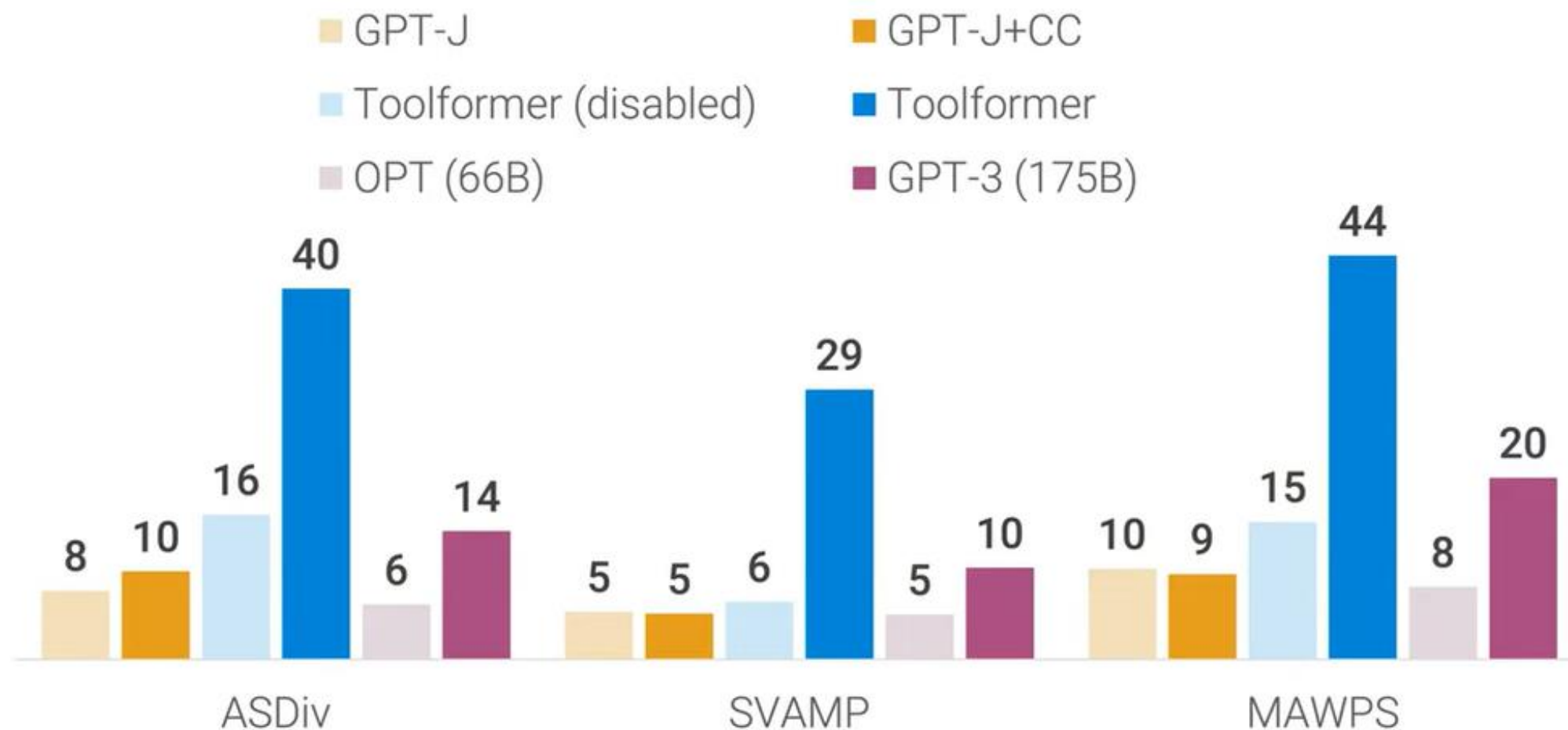




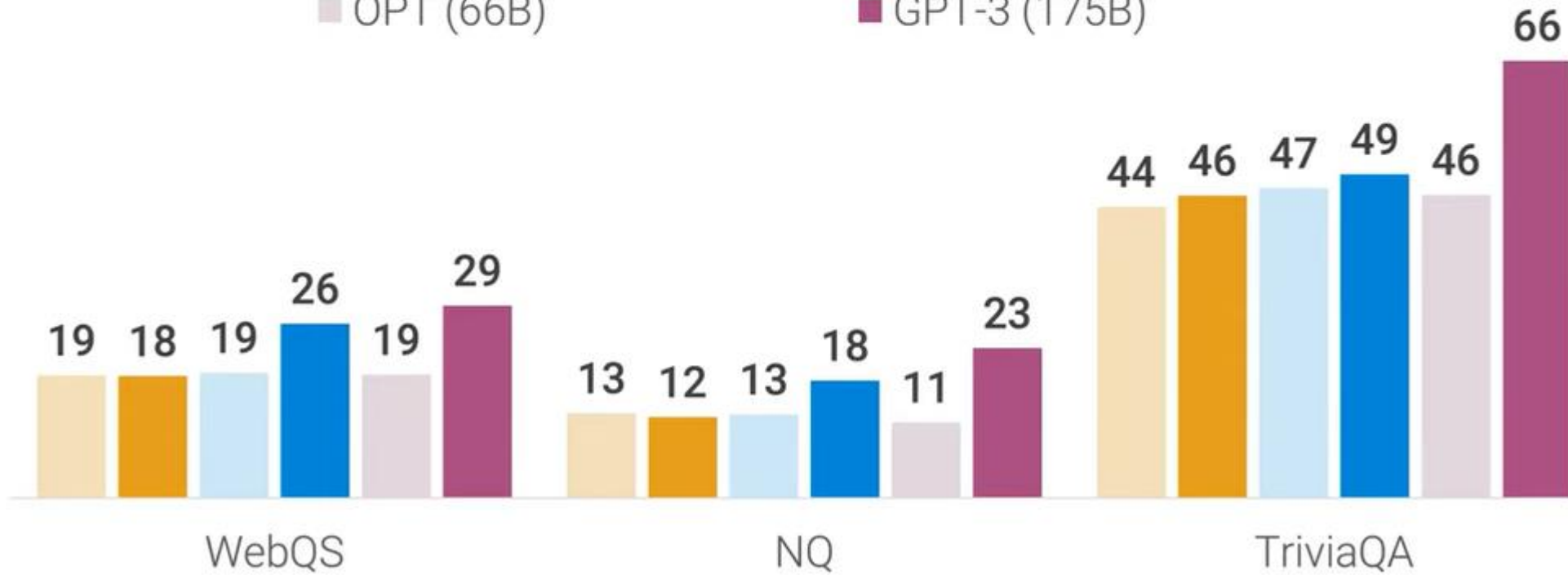
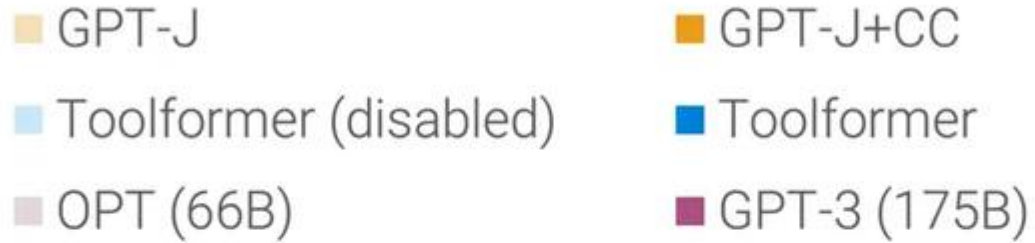
# LAMA Results



## Math Results



## Question Answering Results



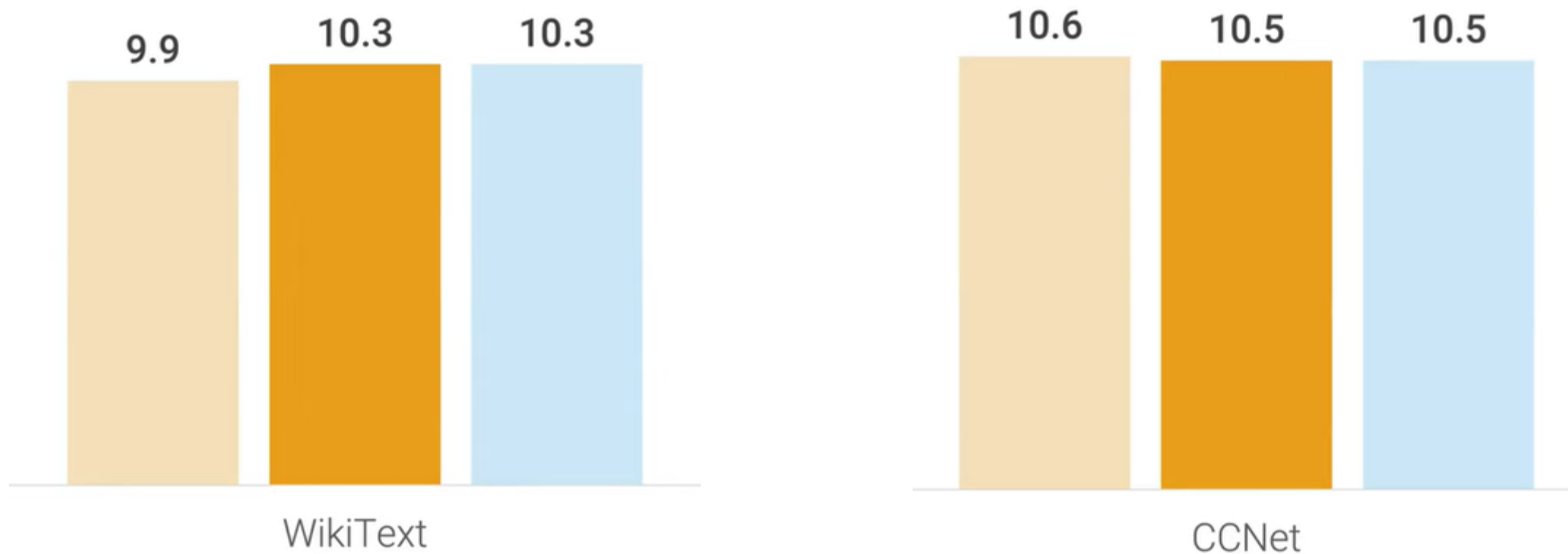
- QA API was disabled in Toolformer
- Toolformer was forced to use Wiki Search

# Toolformer is still an LM!

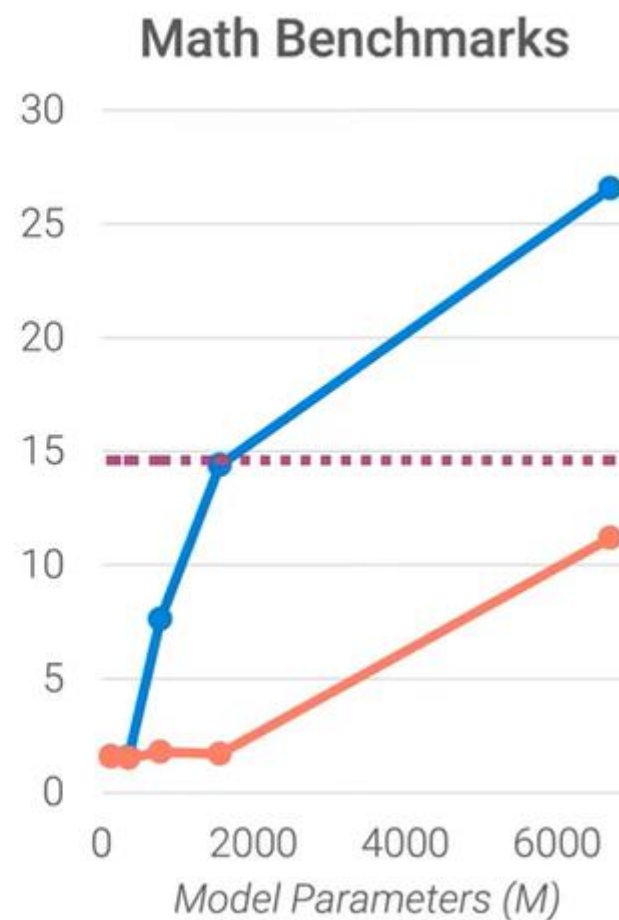
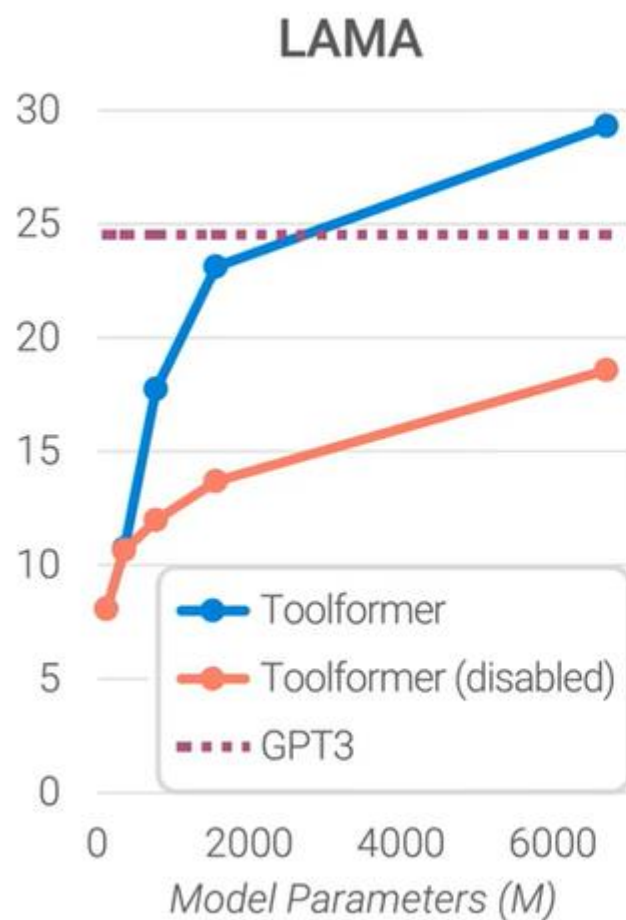
## Perplexity

Toolformer was fine-tuned on CCNet data

■ GPT-J ■ GPT-J+CC ■ Toolformer (disabled)



# Bigger models + Tool usage is better



# What is missing in Toolformer?

No coordination/interplay between the LM  
and the tool!!

# The LLM story of reasoning

Alex has 3 apples. She gave 1 to John and others to Bob. Bob gave half to John as well. John then gave half of his apples to Alex. Who has the highest number of apples?

Why are LLMs not good at reasoning?

1. They don't know **what** computations to perform (yellow)
2. They don't know **how** to perform these computations (purple)

Output from mistralai/Mixtral-8x7B-Instruct-v0.1

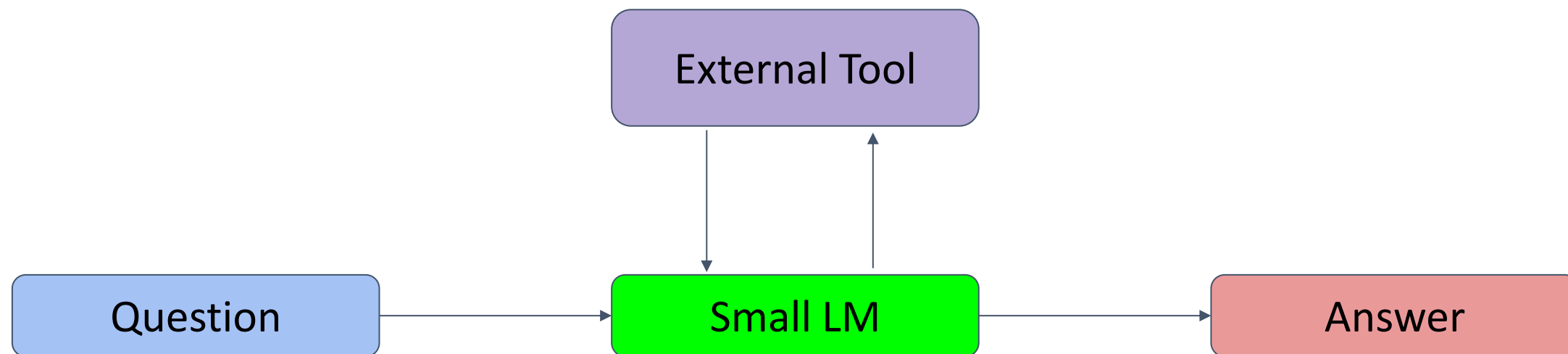
Alex has 3 apples, she gave 1 to jhon and others to bob. Bob gave half to jhon as well. jhon then gave half of his apples to alex. Who has the highest number of apples?  
Answer:

Let's break down what happened:

1. Alex starts with 3 apples and gives 1 away, leaving her with 2 apples.
2. Bob receives half of what Alex initially had, so he gets 1.5 apples (but since we can't have halves of apples, let's say he got 2). After receiving these apples, he gives half of them to John, which is another 1 apple. So now, Bob has no apples left.
3. John ends up with the 1 apple that Alex gave him, plus the 1 apple he received from Bob. Then, he gives half of his total back to Alex, which is 0.5 apples. But again, since you can't really give away half an apple...let's assume he rounds down and gives away 0 apples.
4. Now, let's look at everyone's final counts:
  - Alex has her original 2 apples plus the half apple she took back from John, giving her 2.5 apples.
  - Bob has no apples left after giving all of his to John.
  - John started with 2 apples but ended up giving half of one back to Alex, leaving him with 1.5 apples.

So, based on this information, Alex has the highest number of apples at 2.5.

# Tackling reasoning problems





# Frugal LMs Trained to Invoke Symbolic Solvers Achieve Parameter-Efficient Arithmetic Reasoning

Subhabrata Dutta, Joykirat Singh, Ishan Pandey, Sunny Manchanda, Soumen Chakrabarti, Tanmoy Chakraborty

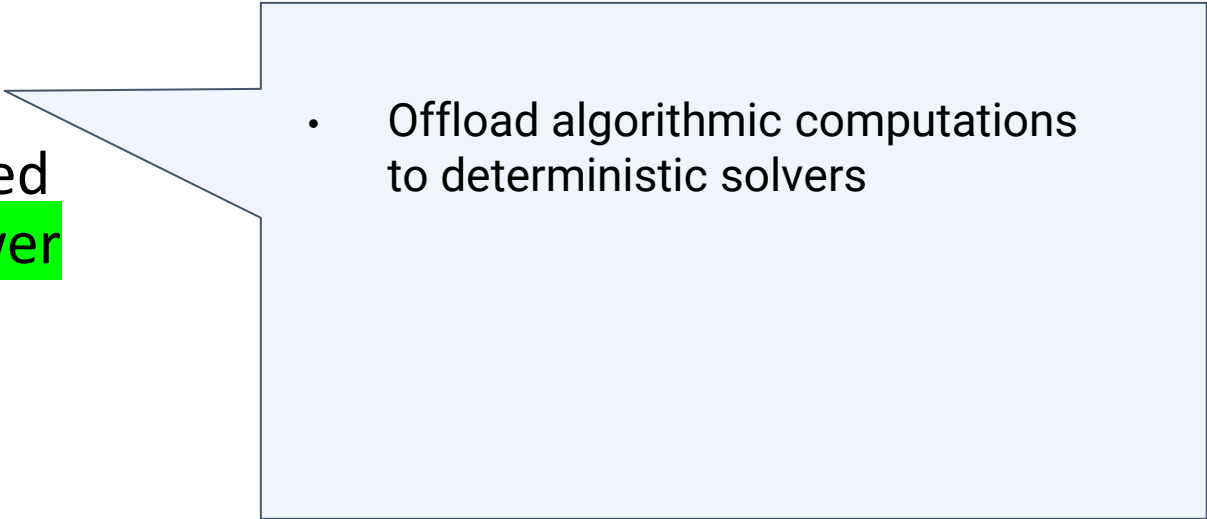
Large Language Models (LLM) exhibit zero-shot mathematical reasoning capacity as a behavior emergent with scale, commonly manifesting as chain-of-thoughts (CoT) reasoning. However, multiple empirical findings suggest that this prowess is exclusive to LLMs with exorbitant sizes (beyond 50 billion parameters). Meanwhile, educational neuroscientists suggest that symbolic algebraic manipulation be introduced around the same time as arithmetic word problems to modularize language-to-formulation, symbolic manipulation of the formulation, and endgame arithmetic. In this paper, we start with the hypothesis that much smaller LMs, which are weak at multi-step reasoning, can achieve reasonable arithmetic reasoning if arithmetic word problems are posed as a formalize-then-solve task. In our architecture, which we call SYRELM, the LM serves the role of a translator to map natural language arithmetic questions into a formal language (FL) description. A symbolic solver then evaluates the FL expression to obtain the answer. A small frozen LM, equipped with an efficient low-rank adapter, is capable of generating FL expressions that incorporate natural language descriptions of the arithmetic problem (e.g., variable names and their purposes, formal expressions combining variables, etc.). We adopt policy-gradient reinforcement learning to train the adapted LM, informed by the non-differentiable symbolic solver. This marks a sharp departure from the recent development in tool-augmented LLMs, in which the external tools (e.g., calculator, Web search, etc.) are essentially detached from the learning phase of the LM. SYRELM shows massive improvements (e.g., +30.65 absolute point improvement in accuracy on the SVAMP dataset using GPT-J 6B model) over base LMs, while keeping our testbed easy to diagnose, interpret and within reach of most researchers.

S Dutta, I Pandey, J Singh, S Manchanda, S Chakrabarti, T Chakraborty, Frugal LMs Trained to Invoke Symbolic Solvers Achieve Parameter-Efficient Arithmetic Reasoning, **AAAI'24**.

# SyReLM: LLM-Symbolic solver coordination

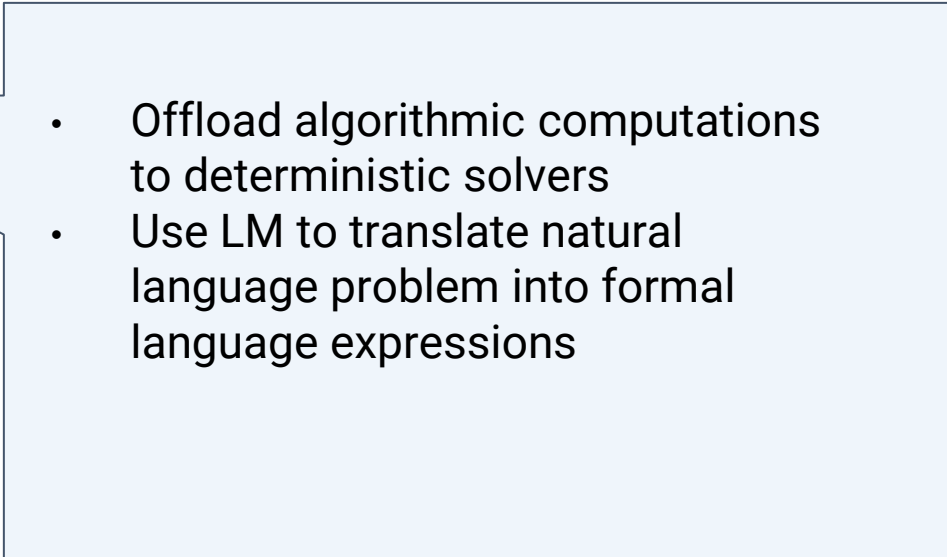
The reasoning requirements:

- Identify what is given, what is asked
- Identify the computational steps required
- Perform the computation to reach answer

- 
- Offload algorithmic computations to deterministic solvers

# SyReLM: LLM-Symbolic solver coordination

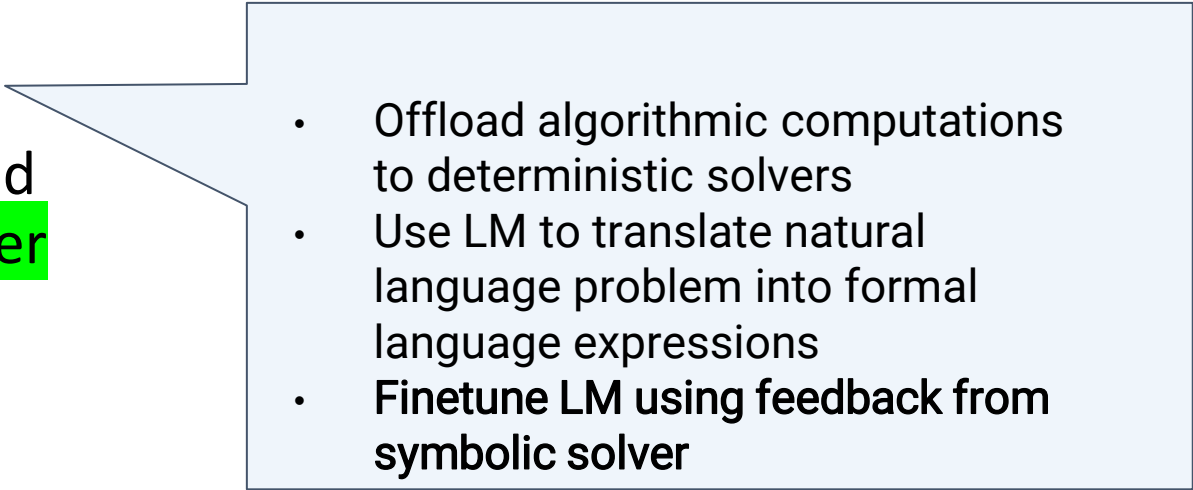
The reasoning requirements:

- Identify what is given, what is asked
  - Identify the computational steps required
  - Perform the computation to reach answer
- 
- Offload algorithmic computations to deterministic solvers
  - Use LM to translate natural language problem into formal language expressions

# SyReLM: LLM-Symbolic solver coordination

The reasoning requirements:

- Identify what is given, what is asked
- Identify the computational steps required
- Perform the computation to reach answer

- 
- Offload algorithmic computations to deterministic solvers
  - Use LM to translate natural language problem into formal language expressions
  - **Finetune LM using feedback from symbolic solver**

For the following reasoning question, generate a python code without importing any libraries which solves the question following these instructions.

Jason grew 37 watermelons and 30 pumpkins. Sandy grew 11 watermelons. How many watermelons did they grow in total?

Arithmetic question in  
natural language

1. State the number of variables required as the first comment line
2. Declare all the variables required as x1, x2, x3.. so on. For each variable declaration, describe clearly what the variable describes as a separate line comment.
3. Define the function solve and pass in all the variables as parameters.
4. Write the function as required, after each line of python code, add a comment describing your intermediate thought process for that step.
5. Return the final answer.

Additional Instructions



```
def solve():  
    # number of variables required = 3  
    # x1 = number of watermelons Jason grew # x2 = number of pumpkins Jason grew  
    # x3 = number of watermelons Sandy grew x1 = 37 # Jason grew 37 watermelons  
    x3 = 11 # Sandy grew 11 watermelons  
    # Total number of watermelons total_watermelons = x1 + x3  
    return total_watermelons  
# Call the function and get the result print(solve())
```

Python Code

<https://github.com/joykirat18/SYRELM>

Training framework of SyReLM

For the following reasoning question, generate a python code without importing any libraries which solves the question following these instructions.

Jason grew 37 watermelons and 30 pumpkins. Sandy grew 11 watermelons. How many watermelons did they grow in total?

1. State the number of variables required as the first comment line
2. Declare all the variables required as x1, x2, x3.. so on. For each variable declaration, describe clearly what the variable describes as a separate line comment.
3. Define the function solve and pass in all the variables as parameters.
4. Write the function as required, after each line of python code, add a comment describing your intermediate thought process for that step.
5. Return the final answer.

Arithmetic question in natural language

Additional Instructions

Adapter finetuning

LLM + Adapter

LoRA finetuning with LM objective

$$\max_{\theta_{LoRA}} \sum_{[T_Q, T_{Inst}], T_{FL}} \sum_{y \in T_{FL}} \log(P(y|[T_Q, T_{Inst}], \theta, \theta_{LoRA}, \mathbf{S}))$$

Python Code

```
def solve():  
    # number of variables required = 3  
    # x1 = number of watermelons Jason grew # x2 = number of pumpkins Jason grew  
    # x3 = number of watermelons Sandy grew x1 = 37 # Jason grew 37 watermelons  
    x3 = 11 # Sandy grew 11 watermelons  
    # Total number of watermelons total_watermelons = x1 + x3  
    return total_watermelons  
# Call the function and get the result print(solve())
```

<https://github.com/joykirat18/SYRELM>

Training framework of SyReLM

For the following reasoning question, generate a python code without importing any libraries which solves the question following these instructions.

Jason grew 37 watermelons and 30 pumpkins. Sandy grew 11 watermelons. How many watermelons did they grow in total?

1. State the number of variables required as the first comment line
2. Declare all the variables required as x1, x2, x3.. so on. For each variable declaration, describe clearly what the variable describes as a separate line comment.
3. Define the function solve and pass in all the variables as parameters.
4. Write the function as required, after each line of python code, add a comment describing your intermediate thought process for that step.
5. Return the final answer.

Arithmetic question in natural language

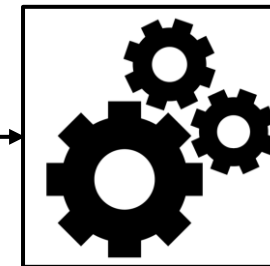
Additional Instructions

Adapter finetuning

LLM + Adapter

```
def solve():  
    # number of variables required = 3  
    # x1 = number of watermelons Jason grew # x2 = number of pumpkins Jason grew  
    # x3 = number of watermelons Sandy grew x1 = 37 # Jason grew 37 watermelons  
    x3 = 11 # Sandy grew 11 watermelons  
    # Total number of watermelons total_watermelons = x1 + x3  
    return total_watermelons  
# Call the function and get the result print(solve())
```

Python Code



Symbolic Solver

Final Answer : 48

<https://github.com/joykirat18/SYRELM>

Training framework of SyReLM

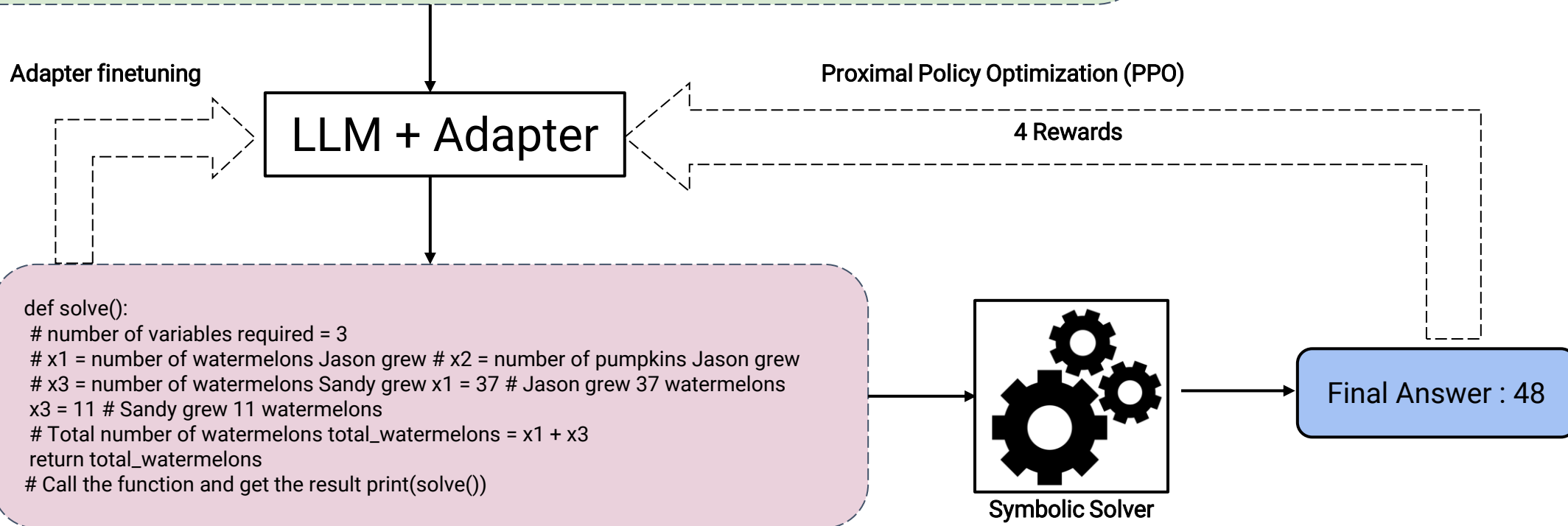
For the following reasoning question, generate a python code without importing any libraries which solves the question following these instructions.

Jason grew 37 watermelons and 30 pumpkins. Sandy grew 11 watermelons. How many watermelons did they grow in total?

1. State the number of variables required as the first comment line
2. Declare all the variables required as x1, x2, x3.. so on. For each variable declaration, describe clearly what the variable describes as a separate line comment.
3. Define the function solve and pass in all the variables as parameters.
4. Write the function as required, after each line of python code, add a comment describing your intermediate thought process for that step.
5. Return the final answer.

Arithmetic question in natural language

Additional Instructions



<https://github.com/joykirat18/SYRELM>

Training framework of SyReLM



# SyReLM Reward Functions

Total reward =  $R1+R2+R3+R4$

**Recipe to school children to solve word problems**

R3: Difference between the number of variables in the generated program and in the gold program

Read the word problem to allocate variables (identify target variable(s))

R2: Number of matching arithmetic operators between generated and gold program

Further parse the text to extract arithmetic relationships and constraints between variables

R4: Absolute difference between the gold answer and the generated answer

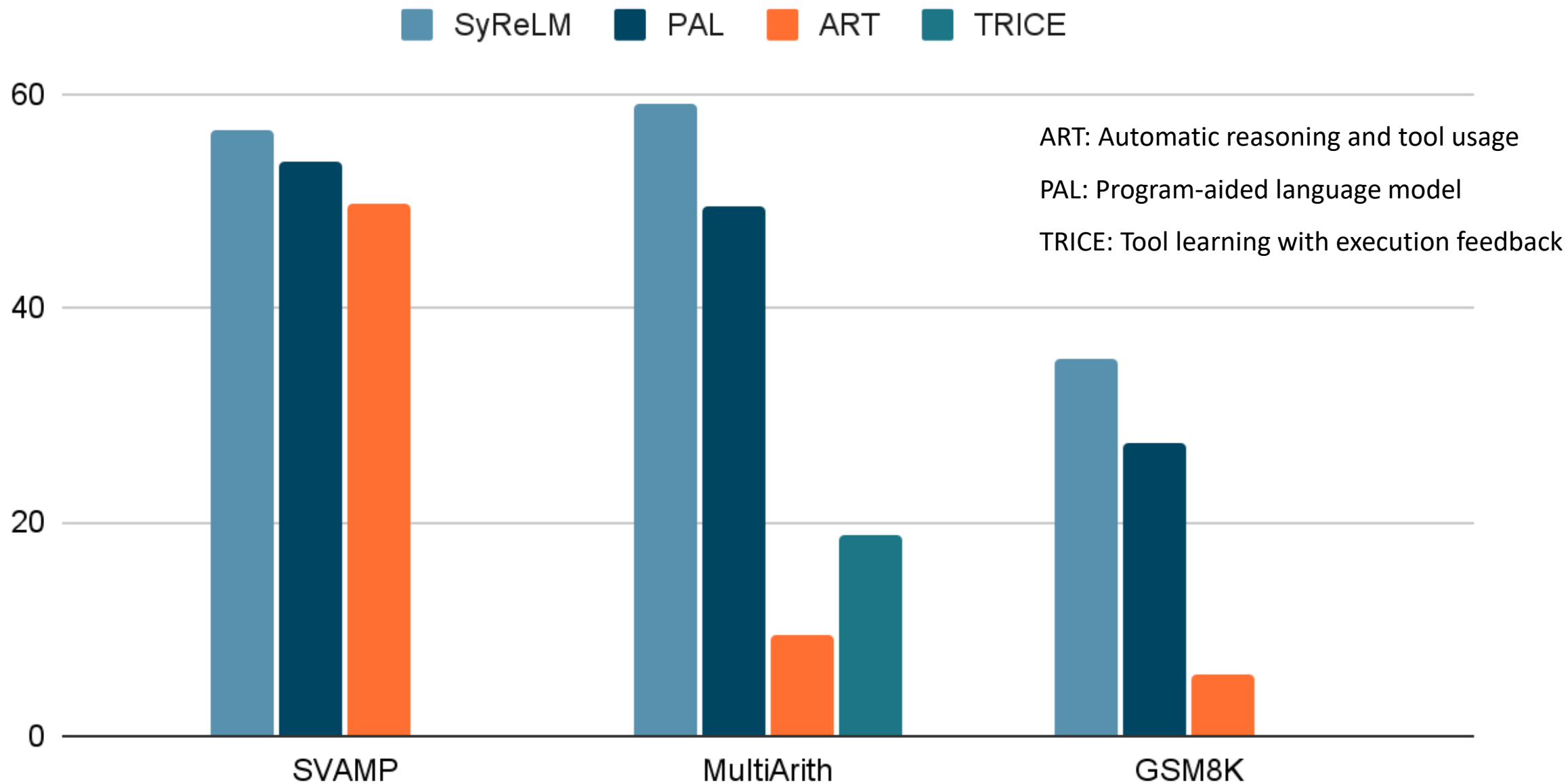
R1: Successful compilation of generated Python code

Invoke a symbolic solver to obtain values for target unknown variables

# SyReLM for SLMs

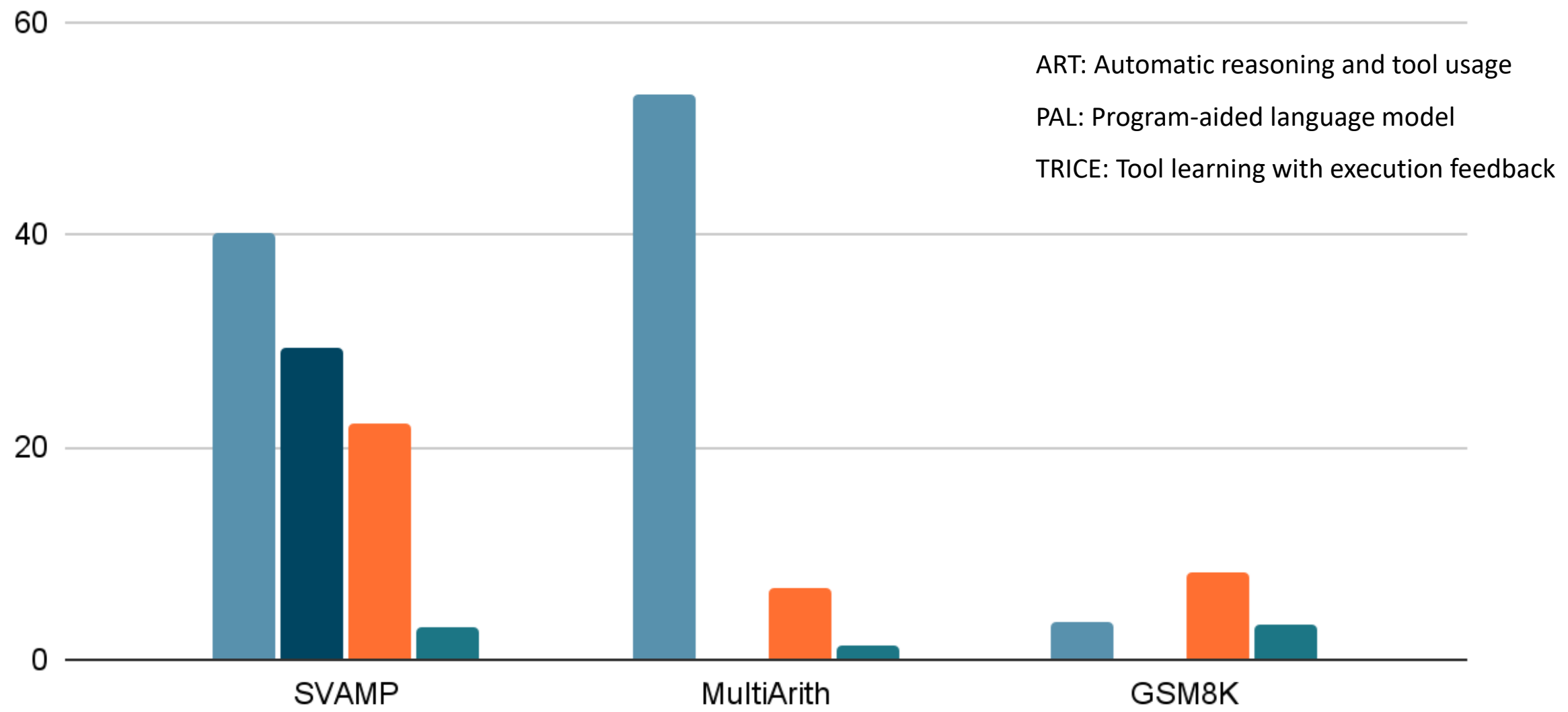
- Relatively smaller LMs (GPT-J 6B, Vicuna 13B)
- Vicuna 13B: Natural language -> Python
- GPT-J: Natural language -> Pseudocode

# Accuracy Vicuna 13B (SyReLM vs. Tool based LMs)

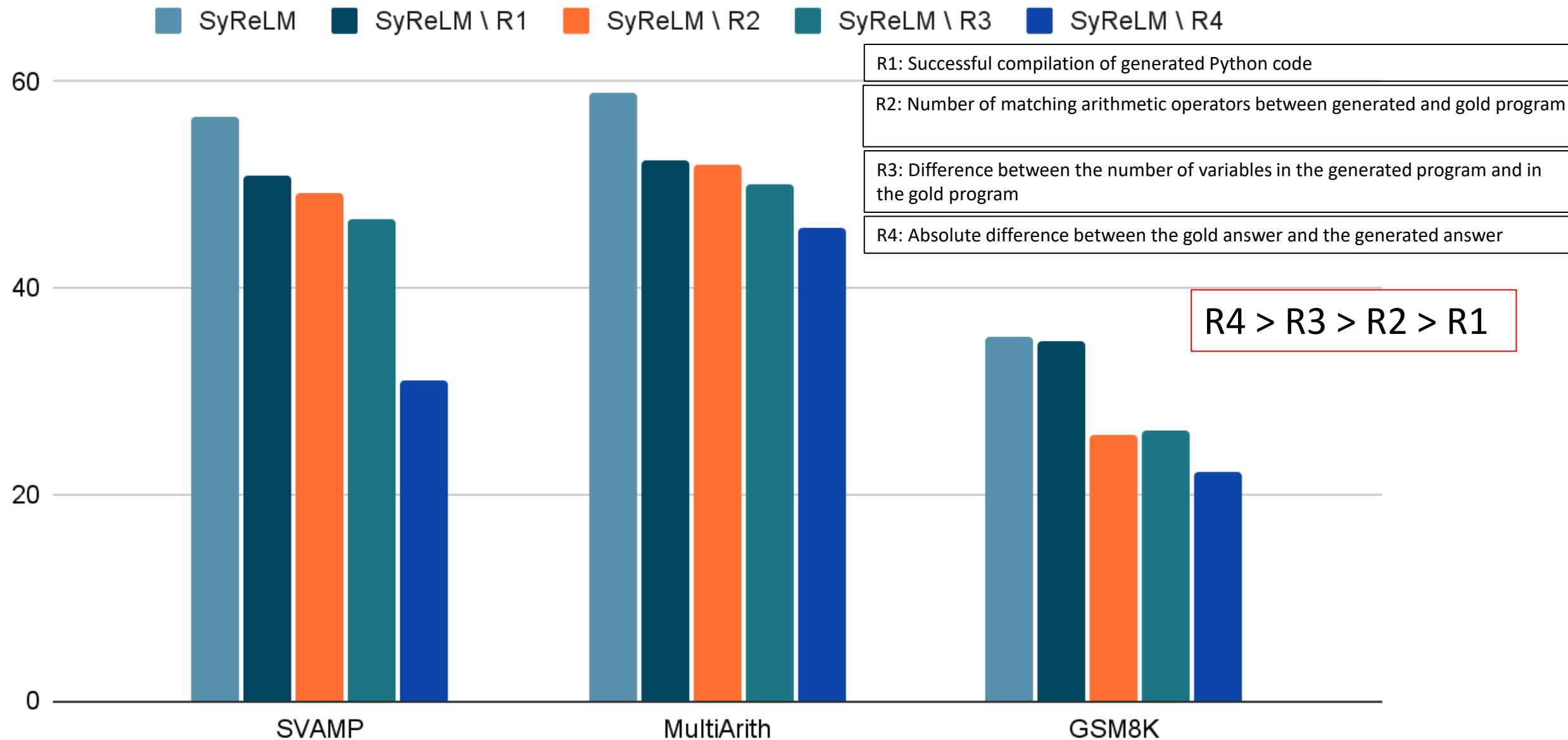


# Accuracy GPT-J (SyReLM vs. Tool based LMs)

SyReLM   Toolformer   PAL   ART



# Ablation Study of SyReLM (Vicuna 13B)



# Platforms for Tool-augmented LMs

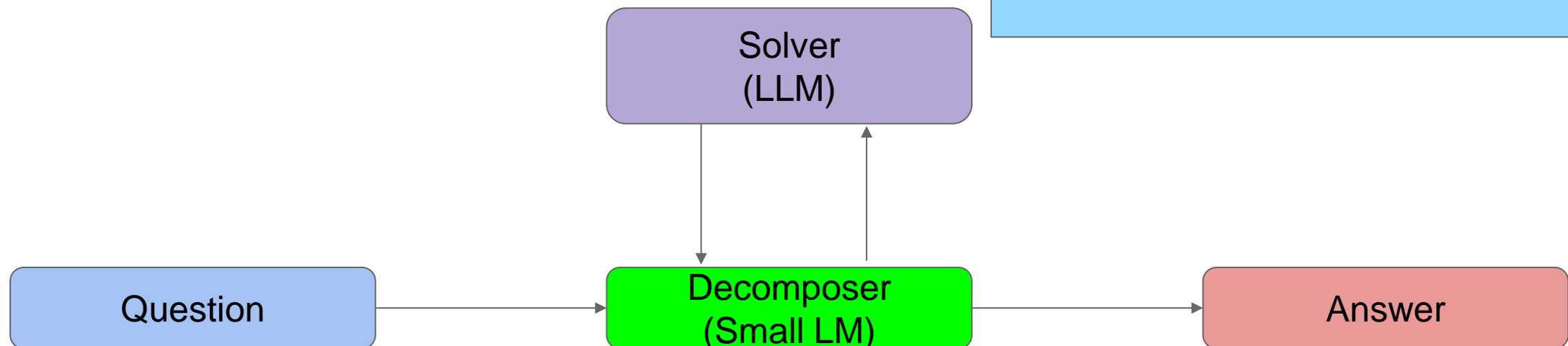
- [TaskMatrix.AI](#) ([March 2023](#))
- [API-Bank](#) ([April 2023](#))
- [OpenAGI](#) ([April 2023](#))
- [Gentopia](#) ([August 2023](#))

# DaSLaM: Decomposer–Solver coordination

The reasoning requirements:

- Identify what is given, what is asked
- Identify the computational steps required
- Perform the computation to reach answer

- Separate LLMs for breaking down the questions (decomposer) and answering them (solver)

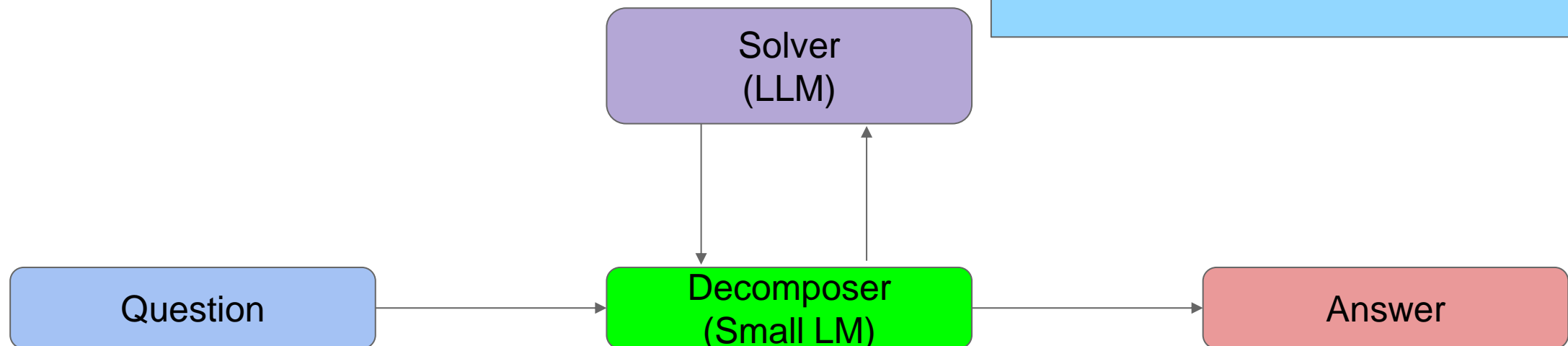


# DaSLaM: Decomposer–Solver coordination

The reasoning requirements:

- Identify what is given, what is asked
- Identify the computational steps required
- Perform the computation to reach answer

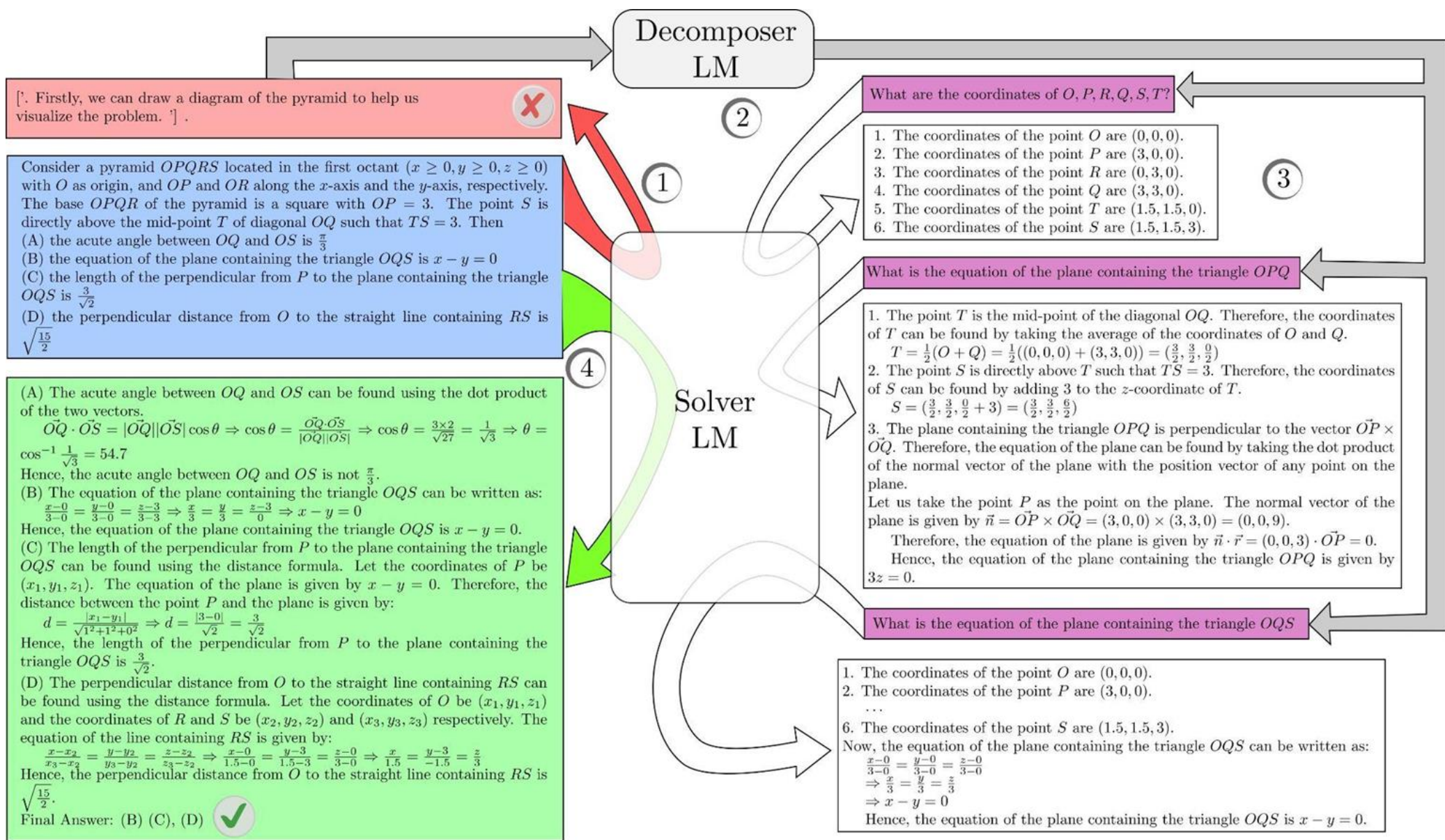
- Separate LLMs for breaking down the questions (decomposer) and answering them (solver)
- Finetune the decomposer to interact with the solver





# DaSLaM: Why such a design choice?

- Separates the decomposer from solver to get rid of solver's limitations that affect decomposition
- Decomposer acts as plug-and-play module that can generalize to any solver
- Decomposition actuates with complete knowledge of the solver's actions
- Improved explainability, modularity and transparency





Original question

[. Firstly, we can draw a diagram of the pyramid to help us visualize the problem. ']

Consider a pyramid  $OPQRS$  located in the first octant ( $x \geq 0, y \geq 0, z \geq 0$ ) with  $O$  as origin, and  $OP$  and  $OR$  along the  $x$ -axis and the  $y$ -axis, respectively. The base  $OPQR$  of the pyramid is a square with  $OP = 3$ . The point  $S$  is directly above the mid-point  $T$  of diagonal  $OQ$  such that  $TS = 3$ . Then  
 (A) the acute angle between  $OQ$  and  $OS$  is  $\frac{\pi}{3}$   
 (B) the equation of the plane containing the triangle  $OQS$  is  $x - y = 0$   
 (C) the length of the perpendicular from  $P$  to the plane containing the triangle  $OQS$  is  $\frac{3}{\sqrt{2}}$   
 (D) the perpendicular distance from  $O$  to the straight line containing  $RS$  is  $\sqrt{\frac{15}{2}}$

(A) The acute angle between  $OQ$  and  $OS$  can be found using the dot product of the two vectors.  
 $\vec{OQ} \cdot \vec{OS} = |\vec{OQ}| |\vec{OS}| \cos \theta \Rightarrow \cos \theta = \frac{\vec{OQ} \cdot \vec{OS}}{|\vec{OQ}| |\vec{OS}|} \Rightarrow \cos \theta = \frac{3 \times 2}{\sqrt{27}} = \frac{1}{\sqrt{3}} \Rightarrow \theta = \cos^{-1} \frac{1}{\sqrt{3}} = 54.7$   
 Hence, the acute angle between  $OQ$  and  $OS$  is not  $\frac{\pi}{3}$ .  
 (B) The equation of the plane containing the triangle  $OQS$  can be written as:  
 $\frac{x-0}{3-0} = \frac{y-0}{3-0} = \frac{z-3}{3-3} \Rightarrow \frac{x}{3} = \frac{y}{3} = \frac{z-3}{0} \Rightarrow x - y = 0$   
 Hence, the equation of the plane containing the triangle  $OQS$  is  $x - y = 0$ .  
 (C) The length of the perpendicular from  $P$  to the plane containing the triangle  $OQS$  can be found using the distance formula. Let the coordinates of  $P$  be  $(x_1, y_1, z_1)$ . The equation of the plane is given by  $x - y = 0$ . Therefore, the distance between the point  $P$  and the plane is given by:  
 $d = \frac{|x_1 - y_1|}{\sqrt{1^2 + 1^2 + 0^2}} \Rightarrow d = \frac{|3 - 0|}{\sqrt{2}} = \frac{3}{\sqrt{2}}$   
 Hence, the length of the perpendicular from  $P$  to the plane containing the triangle  $OQS$  is  $\frac{3}{\sqrt{2}}$ .  
 (D) The perpendicular distance from  $O$  to the straight line containing  $RS$  can be found using the distance formula. Let the coordinates of  $O$  be  $(x_1, y_1, z_1)$  and the coordinates of  $R$  and  $S$  be  $(x_2, y_2, z_2)$  and  $(x_3, y_3, z_3)$  respectively. The equation of the line containing  $RS$  is given by:  
 $\frac{x-x_2}{x_3-x_2} = \frac{y-y_2}{y_3-y_2} = \frac{z-z_2}{z_3-z_2} \Rightarrow \frac{x-0}{1.5-0} = \frac{y-3}{1.5-3} = \frac{z-0}{3-0} \Rightarrow \frac{x}{1.5} = \frac{y-3}{-1.5} = \frac{z}{3}$   
 Hence, the perpendicular distance from  $O$  to the straight line containing  $RS$  is  $\sqrt{\frac{15}{2}}$ .  
 Final Answer: (B) (C), (D)

Decomposer LM

2

What are the coordinates of  $O, P, R, Q, S, T$ ?

1. The coordinates of the point  $O$  are  $(0, 0, 0)$ .
2. The coordinates of the point  $P$  are  $(3, 0, 0)$ .
3. The coordinates of the point  $R$  are  $(0, 3, 0)$ .
4. The coordinates of the point  $Q$  are  $(3, 3, 0)$ .
5. The coordinates of the point  $T$  are  $(1.5, 1.5, 0)$ .
6. The coordinates of the point  $S$  are  $(1.5, 1.5, 3)$ .

3

What is the equation of the plane containing the triangle  $OPQ$ ?

1. The point  $T$  is the mid-point of the diagonal  $OQ$ . Therefore, the coordinates of  $T$  can be found by taking the average of the coordinates of  $O$  and  $Q$ .  
 $T = \frac{1}{2}(O + Q) = \frac{1}{2}((0, 0, 0) + (3, 3, 0)) = (\frac{3}{2}, \frac{3}{2}, 0)$
2. The point  $S$  is directly above  $T$  such that  $TS = 3$ . Therefore, the coordinates of  $S$  can be found by adding 3 to the  $z$ -coordinate of  $T$ .  
 $S = (\frac{3}{2}, \frac{3}{2}, 0 + 3) = (\frac{3}{2}, \frac{3}{2}, 3)$
3. The plane containing the triangle  $OPQ$  is perpendicular to the vector  $\vec{OP} \times \vec{OQ}$ . Therefore, the equation of the plane can be found by taking the dot product of the normal vector of the plane with the position vector of any point on the plane.  
 Let us take the point  $P$  as the point on the plane. The normal vector of the plane is given by  $\vec{n} = \vec{OP} \times \vec{OQ} = (3, 0, 0) \times (3, 3, 0) = (0, 0, 9)$ .  
 Therefore, the equation of the plane is given by  $\vec{n} \cdot \vec{r} = (0, 0, 9) \cdot \vec{OP} = 0$ .  
 Hence, the equation of the plane containing the triangle  $OPQ$  is given by  $3z = 0$ .

What is the equation of the plane containing the triangle  $OQS$ ?

1. The coordinates of the point  $O$  are  $(0, 0, 0)$ .
2. The coordinates of the point  $P$  are  $(3, 0, 0)$ .
- ...
6. The coordinates of the point  $S$  are  $(1.5, 1.5, 3)$ .  
 Now, the equation of the plane containing the triangle  $OQS$  can be written as:  
 $\frac{x-0}{3-0} = \frac{y-0}{3-0} = \frac{z-0}{3-0} \Rightarrow \frac{x}{3} = \frac{y}{3} = \frac{z}{3}$   
 $\Rightarrow x - y = 0$   
 Hence, the equation of the plane containing the triangle  $OQS$  is  $x - y = 0$ .

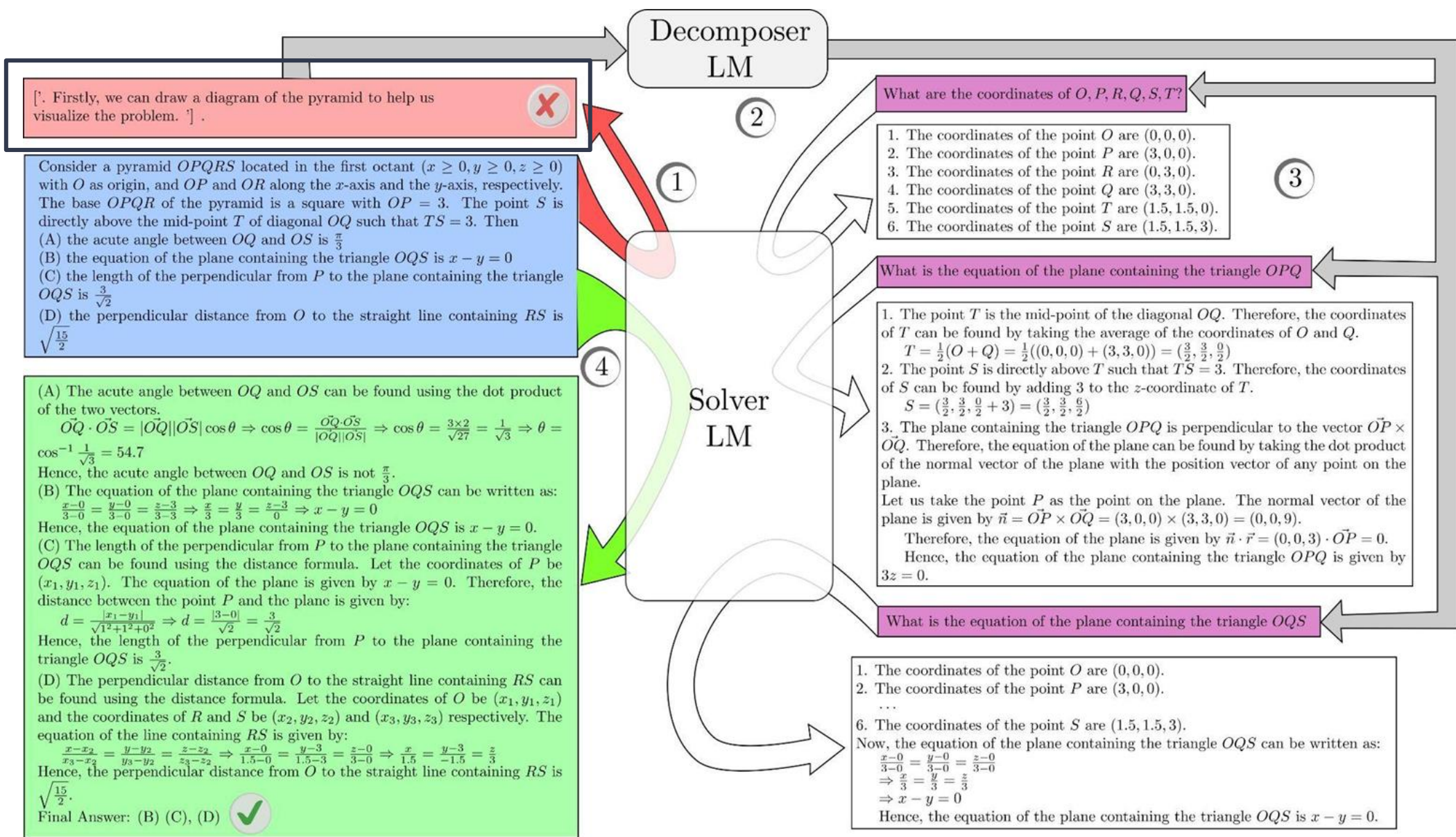
Solver LM

<https://github.com/LCS2-IIITD/DaSLaM>

Example workflow of DaSLaM



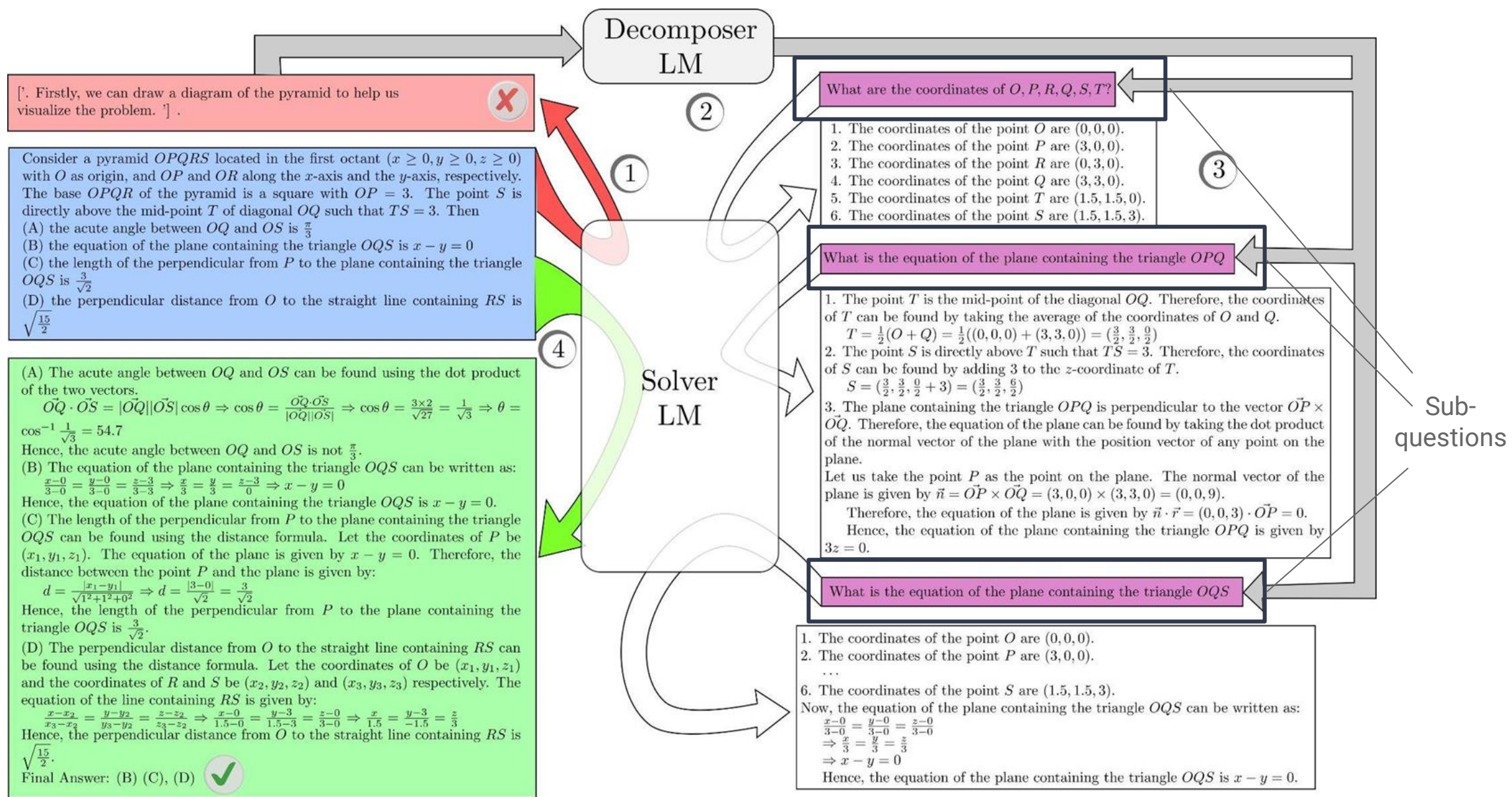
Initial  
incorrect  
answer  
from  
solver



<https://github.com/LCS2-IIITD/DaSLaM>

Example workflow of DaSLaM

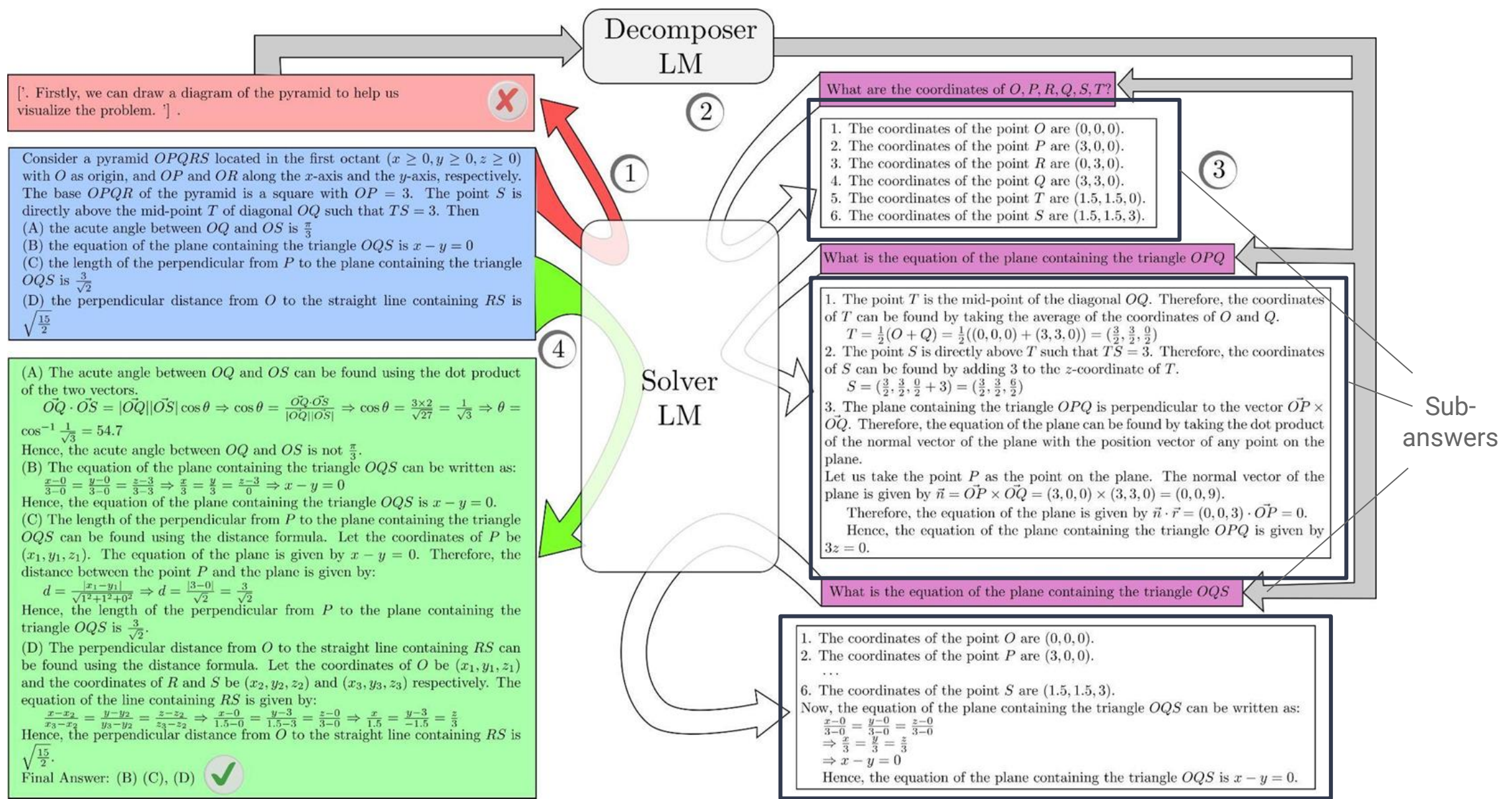




<https://github.com/LCS2-IIITD/DaSLaM>

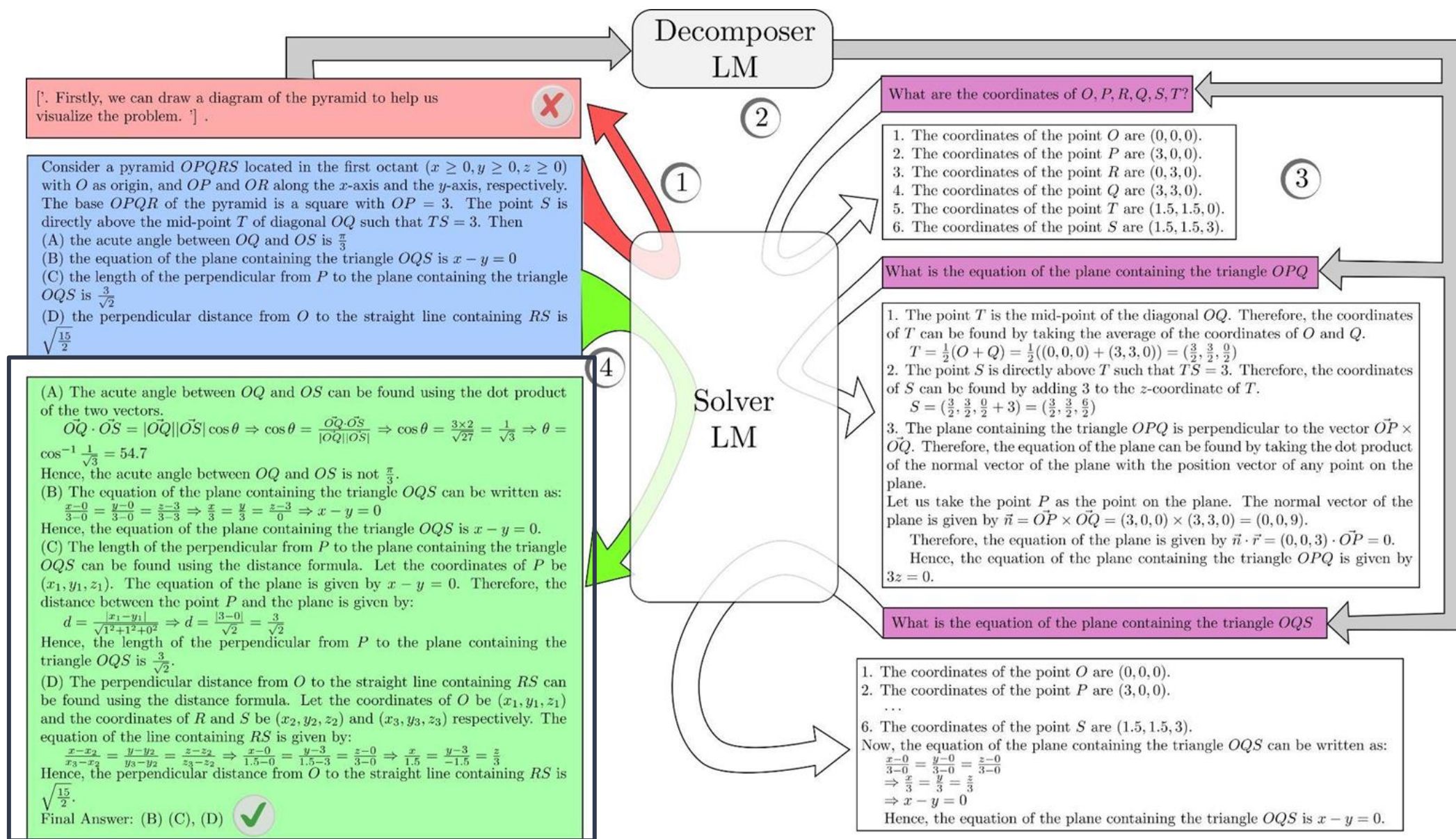
Example workflow of DaSLaM







Refined  
answer  
to  
original  
question



<https://github.com/LCS2-IIITD/DaSLaM>

Example workflow of DaSLaM

# Finetuning Decomposer (LoRA) – Dataset

Each data sample is a sequence of triplets

$$\langle Q_{\text{gold}}, S_{\text{gold}}, A_{\text{gold}}, Q'_{\text{gold}} \rangle$$

$Q_{\text{gold}}$

John borrowed 3 soccer boots from Jake, and forgot them on the field, if peter came across a total of 15 boots on the field Jake's boots inclusive, and he took 4 boots at random, what is the probability that Jake's boots were not amongst the 4 taken?. (A)  $\frac{12}{91}$ , (B)  $\frac{3}{15}$ , (C)  $\frac{12}{15}$  (D)  $\frac{33}{91}$ , (E)  $\frac{3}{91}$

Gold question

$S_{\text{gold}}$

Since Jake owns 3 of the boots, the subset from which the 4 boots should be chosen are the 12 boots not owned by Jake from the universe of 15. The first boot can be one of the 12 from the 15 with probability  $\frac{12}{15}$ . The second boot can be one of the 11 from the 14 remaining with probability  $\frac{11}{14}$ . The third boot can be one of the 10 from the 13 remaining with probability  $\frac{10}{13}$ . The fourth boot can be one of the 9 from the 12 remaining with probability  $\frac{9}{12}$ . The total probability will be  $\frac{12}{15} \cdot \frac{11}{14} \cdot \frac{10}{13} \cdot \frac{9}{12}$ . On cancellation, this comes to  $\frac{33}{91}$

Gold reasoning steps

$A_{\text{gold}}$

3/19

Gold answer

$Q'_{\text{gold}}$

1. How many boots did Jake own?
2. How many boots were on the field?
3. How many boots did Peter take?
4. What is the probability of choosing one of the 12 boots not owned by Jake from the universe of 15?
5. What is the probability of choosing the second, third and fourth boots not owned by Jake?
6. What is the total probability?

Sequence of subproblems generated by decomposing  $Q_{\text{gold}}$

The training data was generated using GPT-3.5 from MATH, AQuA, GSM8K, and StrategyQA



# Finetuning Decomposer (LoRA) – Stage I

Instruction finetuning using LM objective

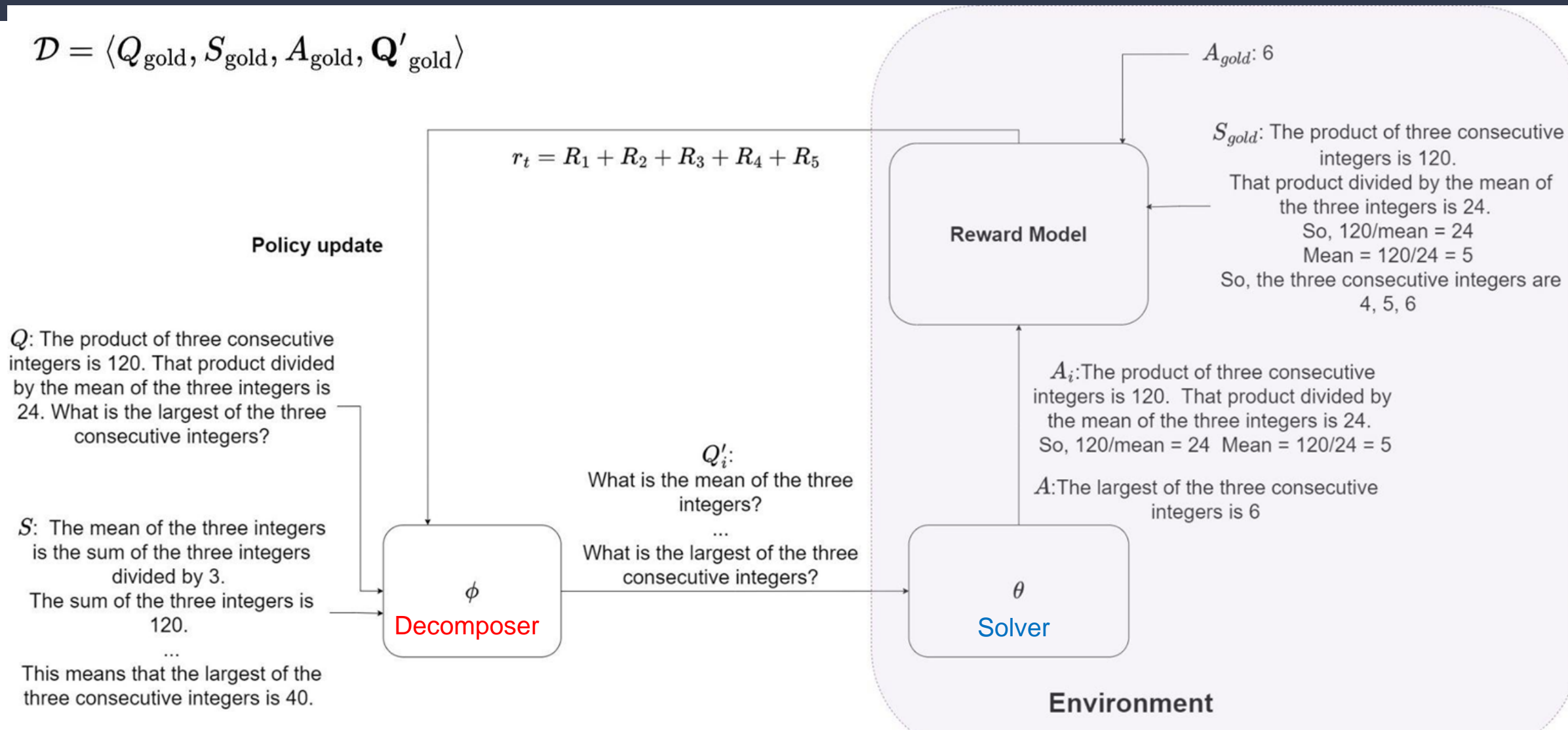
$$\min_{\phi} [-\log(p_{\phi}(\mathbf{Q}'_{\text{gold}}|Q, S))]$$

This stage provides the ability to decompose a problem into subproblems

However, it is still blind to the actual errors made by the solver LM

# Finetuning Decomposer (LoRA) – Stage II

$$\mathcal{D} = \langle Q_{\text{gold}}, S_{\text{gold}}, A_{\text{gold}}, \mathbf{Q}'_{\text{gold}} \rangle$$



# Finetuning Decomposer (LoRA) – Stage II

## Reward functions (Overview)

$R_1$ : Entity coverage between generated and gold subquestions

$R_2$ : Consistency of answers to subproblems

$R_3$ : Number of matching operations in order

$R_4$ : Contrastive proximity between the generated reasoning steps, initial reasoning steps and gold reasoning steps

$R_5$ : Correctness of the final answer

# Finetuning Decomposer (LoRA) – Stage II

## Reward functions (Overview)

$R_1$ : Entity coverage between generated and gold subquestions

**Entity coverage:**  $R_1 = \frac{|E_{Q'}|}{|E_Q|}$ , where  $E_{Q'}$  and  $E_Q$  are the sets of distinct entities in the generated subproblems and the original question, respectively.

$Q_{\text{gold}}$ :

Each good worker can paint my new house alone in 12 hours. Each bad worker can paint my house alone in 36 hours. I need my house painted in 3 hours. If I can only find 3 good workers, how many bad workers must I also find in order to have my house painted on time?

$Q'_{\text{gold}}$ :

1. How many good workers are needed to paint the house in 3 hours?
2. How many bad workers are needed to paint the house in 3 hours?
3. What is the total number of workers needed to paint the house in 3 hours?

$$|E_{Q'}| = 4 + 1 = 5 \quad (11)$$

$$|E_Q| = 6 + 3 = 9 \quad (12)$$

$$R_1 = \frac{|E_{Q'}|}{|E_Q|} = \frac{5}{9} = 0.55 \quad (13)$$

# Finetuning Decomposer (LoRA) – Stage II

## Reward functions (Overview)

$R_2$ : Consistency of answers to subproblems

### Consistency of answers to subproblems:

$$R_2 = \sum_i (I(e_i = \hat{e}_i) + \text{cos-sim}(Q'_i, A_i))$$

where  $\hat{e}_i$  is the entity whose value has been asked in the subproblem  $Q'_i$ , and  $e_i$  is the entity answered. This reward penalizes the decomposer LM for generating questions whose answers are not consistent.

$Q'_1$ : How many good workers are needed to paint the house in 3 hours?

$A_1$ : We can calculate the amount of work done in 3 hours by a good worker by dividing 12 hours by 3 hours. We get 4 as the answer. We know that a bad worker can paint the house in 36 hours. We can calculate the amount of work done in 3 hours by a bad worker by dividing 36 hours by 3 hours. We get 12 as the answer. We need to find out how many good workers are needed to paint the house in 3 hours. We can calculate the number of good workers needed by dividing 12 by 4.

Answer: 3 good workers can paint the house in 3 hours.

$\text{cos-sim}(Q'_i, A_i) = 0.70457435$

$I(e_i = \hat{e}_i) = 1$

$R_2 = 1 + 0.70457435 = 1.70457435$

# Finetuning Decomposer (LoRA) – Stage II

## Reward functions (Overview)

$R_3$ : Number of matching operations in order

**Order of operations:**  $R_3 = \frac{l}{m}$ , where  $l$  is the number of operations matched in order between  $S$  and  $S_{\text{gold}}$ , and  $m$  is the total number of operations in  $S_{\text{gold}}$ .

$S_{\text{model}}$ :

Original Price =  $x$

Discounted Price =  $0.78x$

Price after coupon =  $0.78x - 20$

Price paid =  $0.78x - 20 + 1.90$

$0.78x - 20 + 1.90 = 0.5x$

$0.28x = 21.90$

$x = \boxed{78.21}$

$S_{\text{gold}}$ : Let  $x$  be the original price of item

Discounted price =  $0.78x$

Payment made by the customer after using the \$20 coupon  
=  $0.78x - 20$

$0.78x - 20 = x/2 + 1.9$

$x = \boxed{78.20}$

**operations in  $S_{\text{model}}$ :**  $[-, -, +, -, +]$

**operations in  $S_{\text{gold}}$ :**  $[-, -, /, +]$

$R_3 = \frac{3}{4} = 0.75$



# Finetuning Decomposer (LoRA) – Stage II

## Reward functions (Overview)

$R_4$ : Contrastive proximity between the generated reasoning steps, initial reasoning steps and gold reasoning steps

**CoT proximity:** To ensure that the distance of reasoning produced by the model after prompting  $S$  to the gold reasoning  $S_{gold}$  is less than the distance of reasoning produced without prompt  $S_0$  to the gold reasoning steps  $S_{gold}$ , we design a reward based on the cosine similarity of each step of  $S_{gold}$ . We break  $S$  and  $S_0$  at new-line token to form reasoning steps. At each step  $j$ , we compute  $c_{1j} = \text{cos-sim}(S^j, S_{gold}^j)$  and  $c_{2j} = \text{cos-sim}(S_0^j, S_{gold}^j)$ . The reward is

$$R_4 = \sum_{j=0}^m I(c_{1j} > c_{2j})c_{1j} + I(c_{2j} > c_{1j})(-1 - c_{2j}),$$

$S_{gold}$

1. : We first consider the possible neighbors of the chief's mother, one must be the chief, the other is one of the remaining 5 natives. That native can, in turn, have one of 4 neighbors in addition to the chief's mother
2. Continuing around, there are  $5! = 120$  possible arrangements
3. The chief can be on his mother's left or right, for a total of  $2 \cdot 120 = 240$  arrangements
4. Alternatively, we know that the seating arrangement is the same if it is rotated, once the chief's spot is chosen, the rotations are eliminated
5. There are 2 ways to seat his wife and his mother, and then there are 5 distinct positions for the remaining 5 natives, which makes for  $2 \cdot 5! = \boxed{240}$  possible arrangements

$S_0$ :

1. We can see that the chief must be seated between his mother and his wife.
2. This means that there are 3 possible positions for the chief, and 5 possible positions for the other 6 people.
3. Therefore, there are  $3! \cdot 5! = \boxed{1800}$  different seating arrangements.

$S$ :

1. The chief must sit between his mother and his wife.
2. There are 8 people in the tribe.
3. So, the chief's mother and wife can be arranged in 2! ways.
4. The remaining 5 people can be arranged in  $5!$  ways.
5. Therefore, the total number of possible seating arrangements after removing rotation =  $5! = \boxed{240}$

Cosine similarity calculation per step:

1.  $c_{1j} = 0.47673503, c_{2j} = 0.44773823$
2.  $c_{1j} = 0.45063934, c_{2j} = 0.47917843$
3.  $c_{1j} = 0.5173945, c_{2j} = 0.20383504$
4.  $c_{1j} = 0.46866685, c_{2j} = 0$
5.  $c_{1j} = 0.47825924, c_{2j} = 0$

Hence,  $R_4 = 0.47673503 + (-1 - 0.47917843) + 0.5173945 + 0.46866685 + 0.47825924 = 0.46187719$

# Finetuning Decomposer (LoRA) – Stage II

## Reward functions (Overview)

$R_5$ : Correctness of the final answer

**Correctness of final answer:**  $R_5 = I(\hat{A} = A_{\text{gold}})$ .

$Q_{\text{gold}}$ :

Three friends Alan, Roger and Peter attempt to answer a question on an exam. Alan randomly guesses the answer, giving him a  $\frac{1}{5}$  probability of guessing correctly. Roger cheats by looking at the paper of the student in front of him, giving him a  $\frac{2}{3}$  probability of answering correctly. And Peter dutifully performs the calculations, then marks the answer, giving him a  $\frac{5}{6}$  probability of a correct answer. What is the probability that the question is answered correctly, but not via cheating?

$S_{\text{gold}}$  :

$$\text{Prob}(\text{Alan}) = \frac{1}{5}$$

$$\text{Prob}(\text{Roger}) \text{ without cheating} = \frac{2}{3} - 1 = \frac{1}{3}$$

$$\text{Prob}(\text{Peter}) = \frac{5}{6}$$

$$\text{Total Probability} = \frac{1}{5} \cdot \frac{1}{3} \cdot \frac{5}{6} = \boxed{\frac{1}{18}}$$

$S_{\text{model}}$  :

Alan has a  $\frac{1}{5}$  chance of getting the answer correct.

Roger has a  $\frac{2}{3}$  chance of getting the answer correct.

Peter has a  $\frac{5}{6}$  chance of getting the answer correct.

The probability that the question is answered correctly is  $\frac{1}{5} + \frac{2}{3} + \frac{5}{6} = \frac{13}{12}$ .

The probability that the question is answered correctly, but

$$\text{not via cheating is } 1 - \left(\frac{1}{5} + \frac{2}{3} + \frac{5}{6}\right) = 1 - \frac{13}{12} = \boxed{\frac{-1}{12}}$$

$$A_{\text{gold}}: \frac{1}{18}$$

$$A_{\text{model}}: \frac{-1}{12}$$

$$R_5 = 0$$



# Performance of DaSLaM

Decomposer: LLaMA 13B  
Solver: GPT-3.5

Dataset	Method						
	CoT	L2M	PHP	DSP	GPT3.5 Decomposer	DaSLaM-NF	DaSLaM
PnC	16.4	16.0	10.2	16.2	16.0	20.0	<b>21.4</b>
NT	14.4	11.0	9.8	20.3	14.2	18.4	<b>26.1</b>
ALG	27.6	22.4	24.0	15.3	32.1	31.6	<b>33.4</b>
I-ALG	16.4	16.8	10.0	17.0	18.4	20.8	<b>24.8</b>
Calc.	14.0	14.58	14.28	18.8	12.0	15.1	<b>18.2</b>
P-ALG	32.3	28.0	26.5	28.0	35.5	38.0	<b>44.0</b>
Geom.	14.2	12.5	14.0	5.2	<b>22.0</b>	19.04	21.4
AQuA	41.6	44.7	44.4	44.0	45.4	53.2	<b>54.5</b>

- DaSLaM outperforms existing methods of problem decomposition in single LM

COT: Chain-of-thoughts  
L2M: Least-to-most prompting  
PHP: Progressive Hint Prompting  
DSP: Demonstrate-Search-Predict

# Performance of DaSLaM

Decomposer: LLaMA 13B  
Solver: GPT-3.5

Dataset	Method						
	CoT	L2M	PHP	DSP	GPT3.5 Decomposer	DaSLaM-NF	DaSLaM
PnC	16.4	16.0	10.2	16.2	16.0	20.0	<b>21.4</b>
NT	14.4	11.0	9.8	20.3	14.2	18.4	<b>26.1</b>
ALG	27.6	22.4	24.0	15.3	32.1	31.6	<b>33.4</b>
I-ALG	16.4	16.8	10.0	17.0	18.4	20.8	<b>24.8</b>
Calc.	14.0	14.58	14.28	18.8	12.0	15.1	<b>18.2</b>
P-ALG	32.3	28.0	26.5	28.0	35.5	38.0	<b>44.0</b>
Geom.	14.2	12.5	14.0	5.2	<b>22.0</b>	19.04	21.4
AQuA	41.6	44.7	44.4	44.0	45.4	53.2	<b>54.5</b>

- A Finetuned decomposer is largely superior compared to an orders of magnitude larger LLM (GPT 3.5) prompted to act as a decomposer.
- Prompting docomposer does not help!!

COT: Chain-of-thoughts  
L2M: Least-to-most prompting  
PHP: Progressive Hint Prompting  
DSP: Demonstrate-Search-Predict

# Performance of DaSLaM

Decomposer: LLaMA 13B  
Solver: GPT-3.5

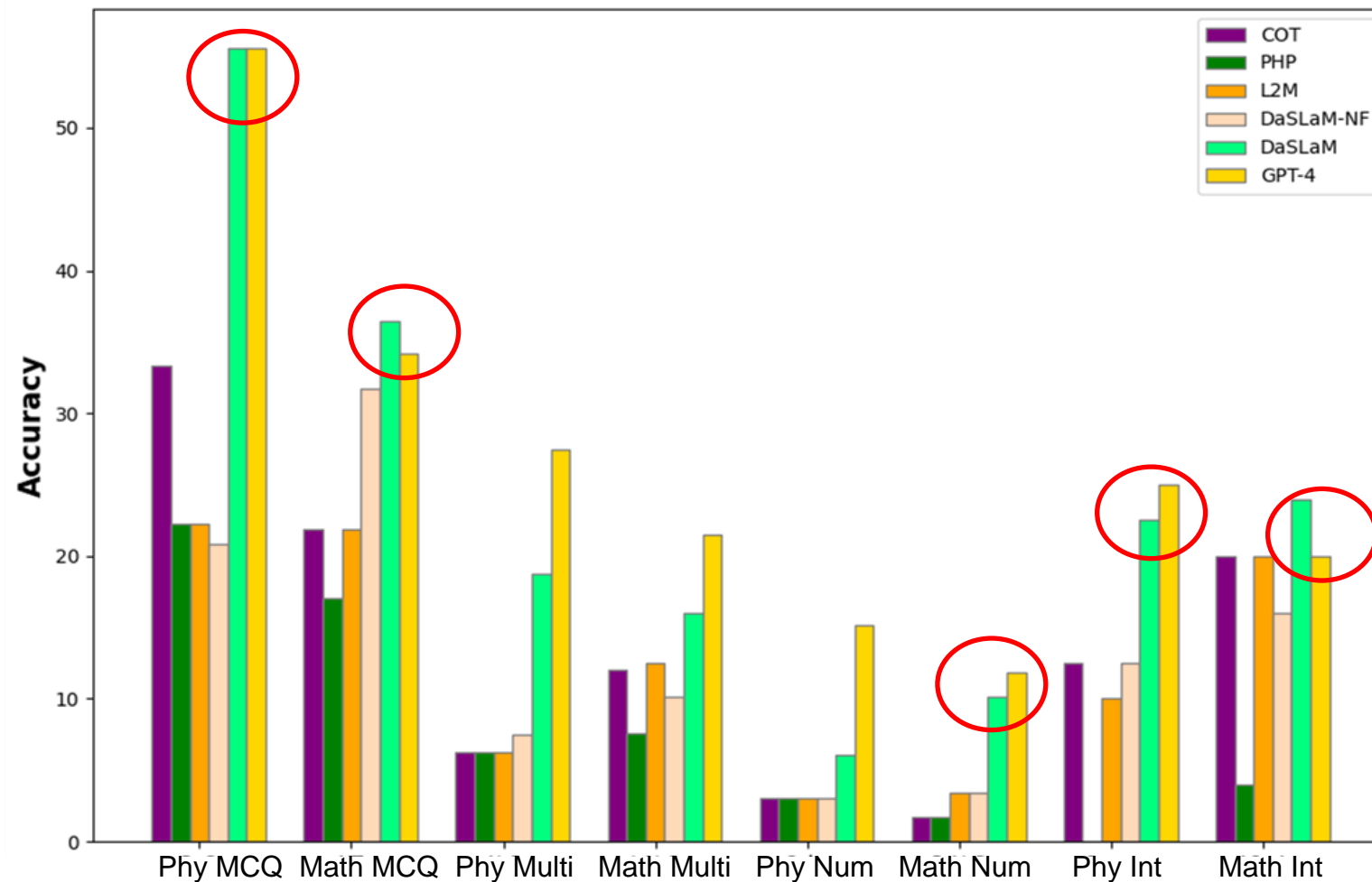
Dataset	Method						DaSLaM-NF	DaSLaM
	CoT	L2M	PHP	DSP	GPT3.5 Decomposer			
PnC	16.4	16.0	10.2	16.2	16.0		20.0	<b>21.4</b>
NT	14.4	11.0	9.8	20.3	14.2		18.4	<b>26.1</b>
ALG	27.6	22.4	24.0	15.3	32.1		31.6	<b>33.4</b>
I-ALG	16.4	16.8	10.0	17.0	18.4		20.8	<b>24.8</b>
Calc.	14.0	14.58	14.28	18.8	12.0		15.1	<b>18.2</b>
P-ALG	32.3	28.0	26.5	28.0	35.5		38.0	<b>44.0</b>
Geom.	14.2	12.5	14.0	5.2	<b>22.0</b>		19.04	21.4
AQuA	41.6	44.7	44.4	44.0	45.4		53.2	<b>54.5</b>

- Feedback from solver is important

COT: Chain-of-thoughts  
L2M: Least-to-most prompting  
PHP: Progressive Hint Prompting  
DSP: Demonstrate-Search-Predict

# Performance of DaSLaM -- JEEBench

DaSLaM makes GPT-3.5 comparable to GPT-4



# Take Aways

- **Heterogeneous specialized modules** improve model efficiency and robustness of LM-based planning and solution of complex word problems.
- LLMs should not just act as a “glue” between input and specialized modules – there should be an “**interplay**”.
- Planner LM
  - need not be very large
  - is relatively easily trained/tuned
  - can be agnostic to tools and solver used
- A **deterministic reward** may be enough (RLHF is always costly)!

# The Future of Tool-Augmented LLMs

- Making APIs accessible for model use
- Providing (multi-step) supervision and scale to 100s and 1000s of APIs (not one per task like Toolformer)
- Pre-training and Instruction-finetuning tool-augmented LLMs
- Improving reasoning and problem decomposition
- Compensating for API errors and preventing error cascades