

Stream of Search (SoS): Learning to Search in Language



Oral at COLM 2024

Kanishk Gandhi*
Stanford University

Denise Lee
Stanford University

Gabriel Grand
MIT

Muxin Liu
Harvey Mudd

Winson Cheng
Stanford University

Archit Sharma
Stanford University

Noah D. Goodman
Stanford University

BG & Motivation

Try solve this problem:

Combine [2, 5, 6, 8] to get 24 using arithmetic operations

Claude 3.5 Sonnet

1. To combine the numbers [2, 5, 6, 8] to get 24, we need to find the correct arithmetic operations. Let's think through this step-by-step:

First, let's consider multiplication, as it's often the quickest way to get larger numbers.

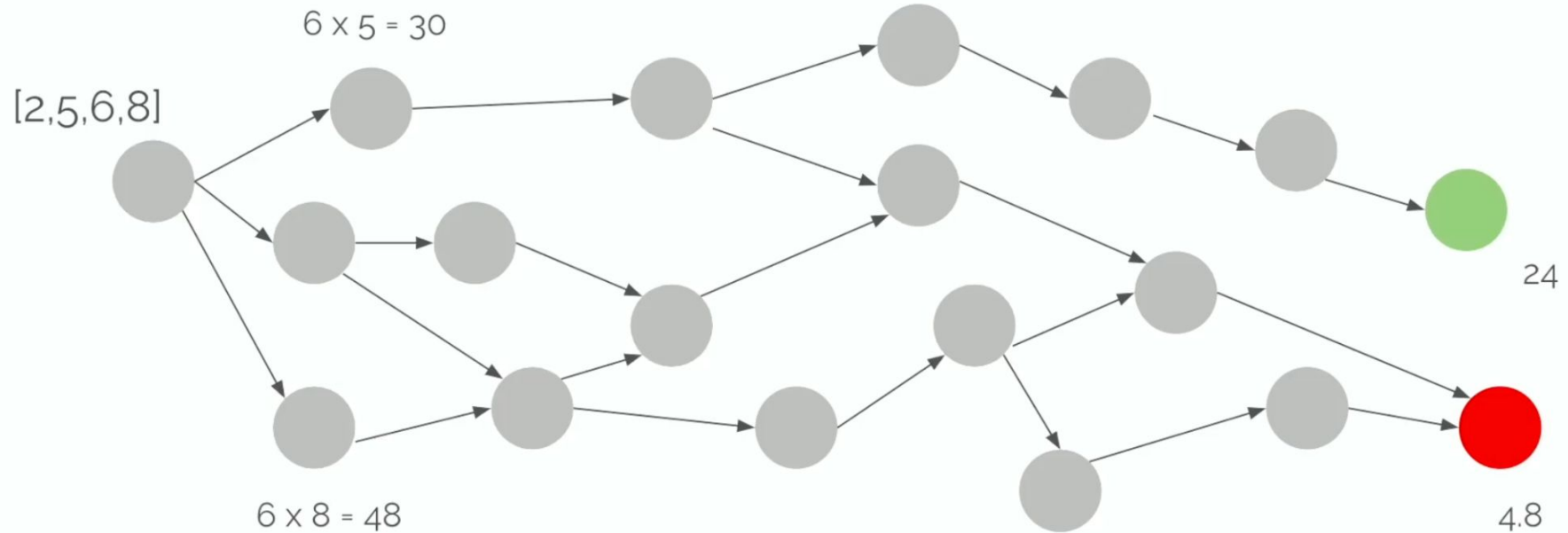
$6 \times 8 = 48$, which is a good start as it's a multiple of 24.

2. Now we need to use 2 and 5 to divide 48 by 2.

We can do this by multiplying 2 and 5 to get 10, then dividing 48 by 10. So the solution is

3. $(6 \times 8) / (2 \times 5) = 24 \rightarrow 48 / 10 \neq 24$

Snowballing of errors and the problem of looking ahead



How do human **search** and **backtrack**?

Combine [2, 5, 6, 8]
to get 24!

Human think-out-loud Transcription

8 times 2 is 16 plus 32 minus 5 that's 43 **not going to work**

6 plus 2 8 plus 5 13 **no that's not going to work**

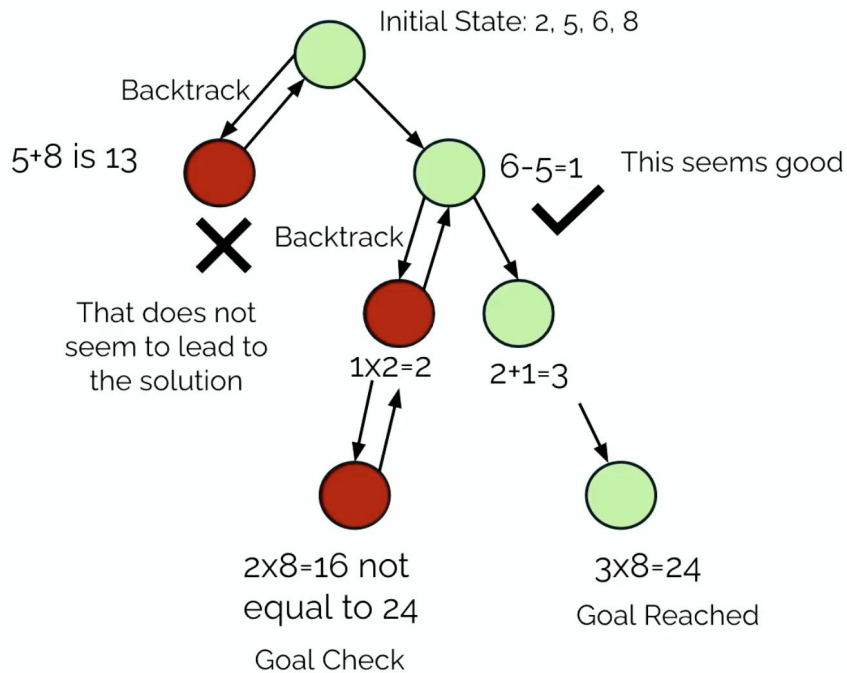
6 minus 2 6 minus 2 Plus five **no, that's not gonna work**

....

Maybe it's eight minus two eight minus two minus 2
8 minus 2 3 times 6 **no** 8 minus 2 which would be 3,
plus 5...

30 minus 2. **Oh, wait a minute. 6 times 5 30 minus 8 plus 2 is 24.**

Hypothesis: Language models struggle to search as they never see the process of search during training



Human search and backtrack

vs.

Language Model

What language models see:

Initial State: 2, 5, 6, 8

Let's Think Step by Step:

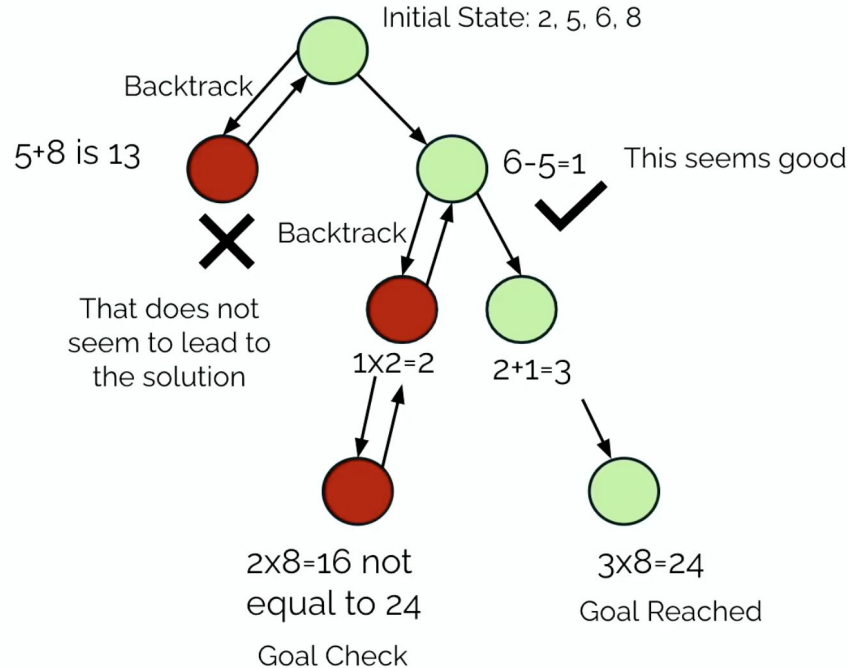
Step 1: $6-5 = 1$

Step 2: $2+1 = 3$

Step 3: $3 \times 8 = 24$

LLMs being unable to search and backtrack might not be the an architecture problem but a **data problem**.

We can linearize the process of search to get a Stream of Search



Initial State: 2, 5, 6, 8

Let's Think Step by Step:

Step 1: Let's try $5+8=13$.

This does not seem promising. Going back to the start and trying again.

Step 1: $6-5=1$. This seems good.

Step 2: $1 \times 2 = 2$. This seems good.

Step 3: $2 \times 8 = 16$. This is not the target. Going back to step 2 and trying again.

Step 2: $2+1=3$

Step 3: $3 \times 8 = 24$. This is the target. Goal Reached!

We can teach language models to search and solve complex problems by showing them **Streams of Search** - flattened strings that represent search traces with exploration, verification and backtracking

Problem Setup

Task Description: Countdown

Target Number: a number $[10, 100]$

Inputs: 4 integers in $[1, \text{target}]$

Task: Combine the inputs using addition, subtraction multiplication and division to get the target.



How do we represent search in language?

A Vocabulary for Search

States:

- a. Current State
- b. Goal State

Actions:

- a. State Expansion Function
- b. Exploration Choice
- c. Pruning
- d. Goal Check / Verification
- e. Backtracking

Heuristic:

Function that approximates the distance of the current state from the goal state

“We choose to keep the heuristic functions, values of states and the pruning strategy implicit.”

How do we represent search in language?

A Vocabulary for Search

Stream of Search Solution

```
Current State: 18:[74, 24, 36, 44], Operations: []
Exploring Operation: 74-44=30, Resulting Numbers: [24, 36, 30]
Generated Node #0,0: 18:[24, 36, 30] Operation: 74-44=30
Exploring Operation: 44-36=8, Resulting Numbers: [74, 24, 8]
Generated Node #0,1: 18:[74, 24, 8] Operation: 44-36=8
Exploring Operation: 44-24=20, Resulting Numbers: [74, 36, 20]
Generated Node #0,2: 18:[74, 36, 20] Operation: 44-24=20
Exploring Operation: 74-24=50, Resulting Numbers: [36, 44, 50]
Generated Node #0,3: 18:[36, 44, 50] Operation: 74-24=50
Exploring Operation: 74-36=38, Resulting Numbers: [24, 44, 38]
Generated Node #0,4: 18:[24, 44, 38] Operation: 74-36=38
Moving to Node #0,0
Current State: 18:[24, 36, 30], Operations: ['74-44=30']
Exploring Operation: 36-24=12, Resulting Numbers: [30, 12]
Generated Node #0,0,0: 18:[30, 12] Operation: 36-24=12
Exploring Operation: 36-30=6, Resulting Numbers: [24, 6]
Generated Node #0,0,1: 18:[24, 6] Operation: 36-30=6
Exploring Operation: 24+30=54, Resulting Numbers: [36, 54]
Generated Node #0,0,2: 18:[36, 54] Operation: 24+30=54
Exploring Operation: 30-24=6, Resulting Numbers: [36, 6]
Generated Node #0,0,3: 18:[36, 6] Operation: 30-24=6
Exploring Operation: 24+36=60, Resulting Numbers: [30, 60]
Generated Node #0,0,4: 18:[30, 60] Operation: 24+36=60
Moving to Node #0,4
Current State: 18:[24, 44, 38], Operations: ['74-36=38']
Exploring Operation: 44-38=6, Resulting Numbers: [24, 6]
Generated Node #0,4,0: 18:[24, 6] Operation: 44-38=6
Exploring Operation: 44-24=20, Resulting Numbers: [38, 20]
Generated Node #0,4,1: 18:[38, 20] Operation: 44-24=20
Exploring Operation: 24+38=62, Resulting Numbers: [44, 62]
Generated Node #0,4,2: 18:[44, 62] Operation: 24+38=62
Exploring Operation: 24+44=68, Resulting Numbers: [38, 68]
Generated Node #0,4,3: 18:[38, 68] Operation: 24+44=68
Exploring Operation: 38-24=14, Resulting Numbers: [44, 14]
Generated Node #0,4,4: 18:[44, 14] Operation: 38-24=14
Moving to Node #0,1
Current State: 18:[74, 24, 8], Operations: ['44-36=8']
Exploring Operation: 74-8=66, Resulting Numbers: [24, 66]
Generated Node #0,1,0: 18:[24, 66] Operation: 74-8=66
Exploring Operation: 24-8=16, Resulting Numbers: [74, 16]
Generated Node #0,1,1: 18:[74, 16] Operation: 24-8=16
Exploring Operation: 24+8=32, Resulting Numbers: [74, 32]
Generated Node #0,1,2: 18:[74, 32] Operation: 24+8=32
Exploring Operation: 74+8=82, Resulting Numbers: [24, 82]
Generated Node #0,1,3: 18:[24, 82] Operation: 74+8=82
Exploring Operation: 74-24=50, Resulting Numbers: [8, 50]
Generated Node #0,1,4: 18:[8, 50] Operation: 74-24=50
Moving to Node #0,2
Current State: 18:[74, 36, 30], Operations: ['44-24=20']
```

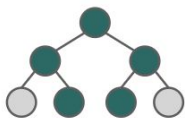
“We choose to keep the heuristic functions, values of states and the pruning strategy implicit.”

Creating a Dataset with SoS

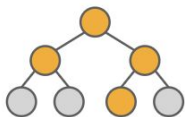
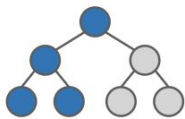
(a) Search problem

Target: 50
Numbers:
39, 66, 33, 13

Solution:
 $39 + 13 = 52$
 $66 / 33 = 2$
 $52 - 2 = 50$



Diverse
Search
Strategies



(b) Streams of Search

```
Current State: 50:[39, 66, 33, 13], Operations: []  
Exploring Operation: 66-13=53, Resulting Numbers: [39, 33, 53]  
Generated Node #0,0: 50:[39, 33, 53] Operation: 66-13=53  
Moving to Node #0,0  
Current State: 50:[39, 33, 53], Operations: ['66-13=53']  
Exploring Operation: 39+33=72, Resulting Numbers: [53, 72]  
Generated Node #0,0,0: 50:[53, 72] Operation: 39+33=72  
Moving to Node #0,0,0  
Current State: 50:[53, 72], Operations: ['66-13=53', '39+33=72']  
Exploring Operation: 72-53=19, Resulting Numbers: [19]  
19,50 unequal: No Solution  
Moving to Node #0,0  
Current State: 50:[39, 33, 53], Operations: ['66-13=53']  
Exploring Operation: 53-33=20, Resulting Numbers: [39, 20]  
...  
Moving to Node #0,2  
Current State: 50:[66, 33, 52], Operations: ['39+13=52']  
Exploring Operation: 66/33=2, Resulting Numbers: [52, 2]  
Generated Node #0,2,2: 50:[52, 2] Operation: 66/33=2  
Moving to Node #0,2,2  
Current State: 50:[52, 2], Operations: ['39+13=52', '66/33=2']  
Exploring Operation: 52-2=50, Resulting Numbers: [50]  
50,50 equal: Goal Reached
```

Start state

Backtracking

Search Strategy

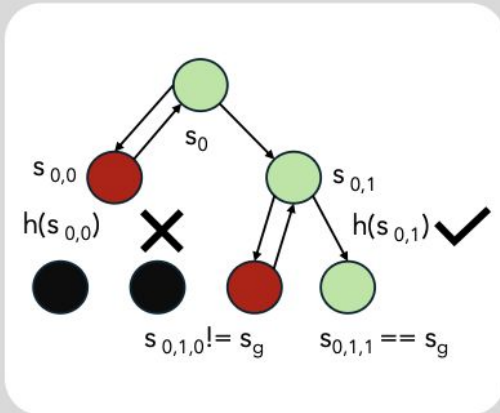
Goal state

Sample with Diverse Search
Strategies like BFS and DFS with
different parameters and heuristics

Training Data

Target Range: [10, 100] 500, 000 problems

1. Train model on **Optimal Path** (baseline)
2. Train model on **Stream of Search** dataset



Stream of Search

$s_c = s_0$ → Current State
 $s_{0,0} = SE(s_0)$ → State Expansion
Moving to s_0 → Moving between states
 $s_c = s_0$
 $s_{0,1} = SE(s_0)$ → Implicitly evaluate w/ heuristic and prune
Moving to $s_{0,1}$ → Implicit strategy for exploring
 $s_c = s_{0,1}$
 $s_{0,1,0} = SE(s_0)$ → Goal Check
 $s_{0,1,0} \neq s_g$ → Backtracking
Moving to $s_{0,1}$ → Backtracking
 $s_c = s_{0,1}$
 $s_{0,1,1} = SE(s_0)$ → Goal State
 $s_{0,1,1} == s_g$ → Goal Reached!
Goal Reached!

Optimal Path

$s_c = s_0$
 $s_{0,1} = SE(s_0)$
Moving to $s_{0,1}$
 $s_c = s_{0,1}$
 $s_{0,1,1} = SE(s_0)$
 $s_{0,1,1} == s_g$
Goal Reached!

Only correct steps included

285,000 correct solutions

500,000 correct solutions

Training Setup

Train

Train a causal language model with

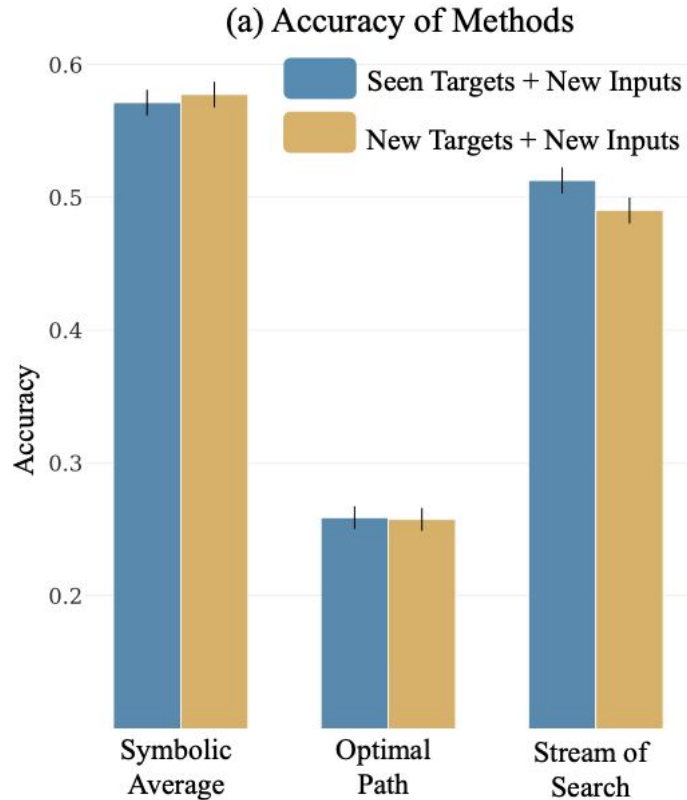
1. Train model on **Optimal Path**
2. Train model on **Stream of Search**

A **250M GPT-Neo** model trained from scratch

Evaluation

- 1) Seen targets with new sets of input
24(seen) \rightarrow [3, 10, 13, 4] (new)
- 2) New targets with new sets of inputs
13(new) \rightarrow [4, 3, 1, 1] (new)

SoS Model outperforms a model trained on Optimal Solution

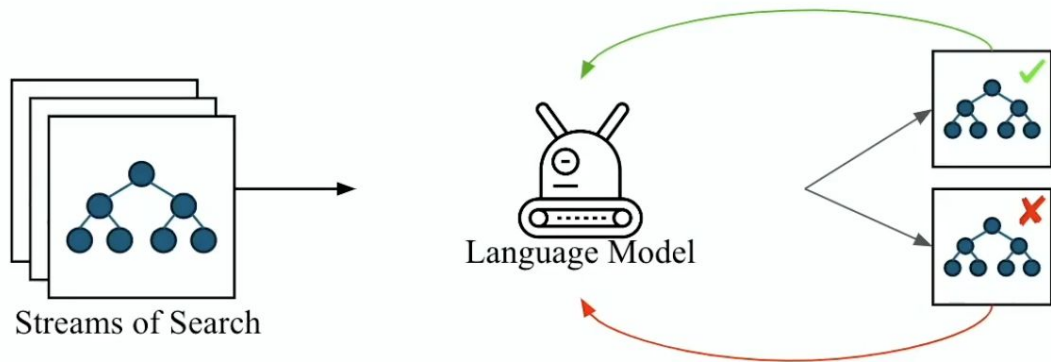


Symbolic Average: BFS/DFS; strategy used to construct the dataset.

Optimal Path: Trained with solutions containing only the correct steps.

Stream Of Search: Trained with solutions containing only the all steps (including backtracking).

Improve the SoS model by giving it feedback based on outcomes

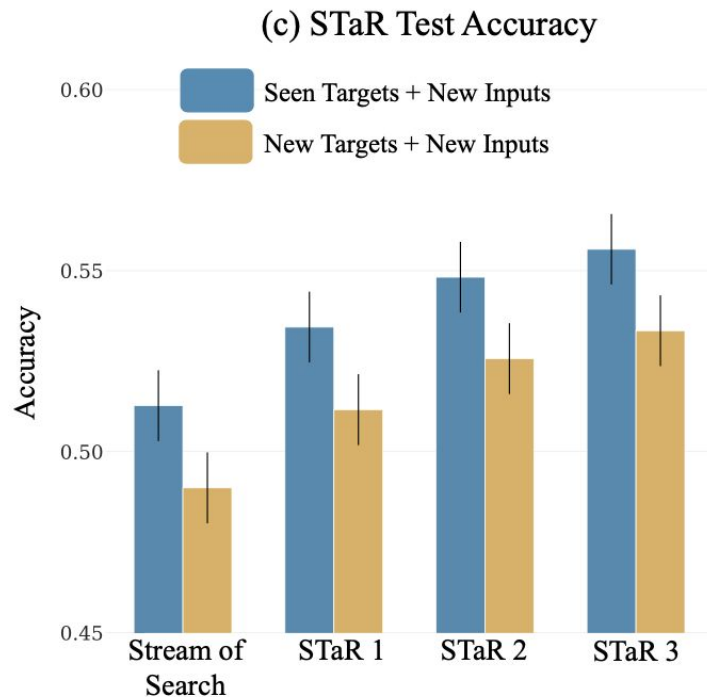


With

- 1) Actor-Critic:
Advantage-Induced Policy
Alignment (APA)
- 2) Expert Iteration with STaR
(Self-Taught Reasoner)

Policy Improvement with STaR

1. Generate 100,000 correct solutions for problems in the training set with the model from the previous iteration or the base-SoS model
2. Finetune the base-SoS model on this dataset
3. Repeat until we observe a convergence in validation accuracy



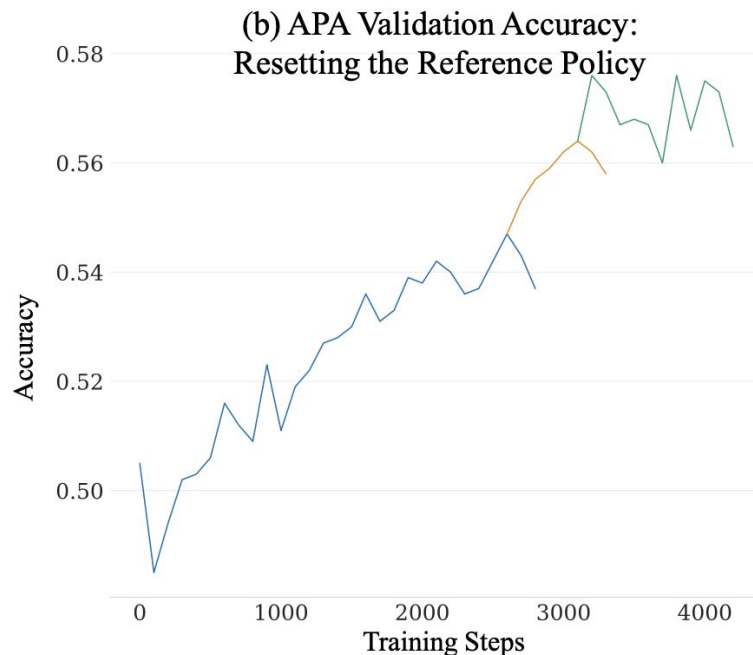
Policy Improvement Actor Critic with APA

Create a copy of the language model to serve as a value network

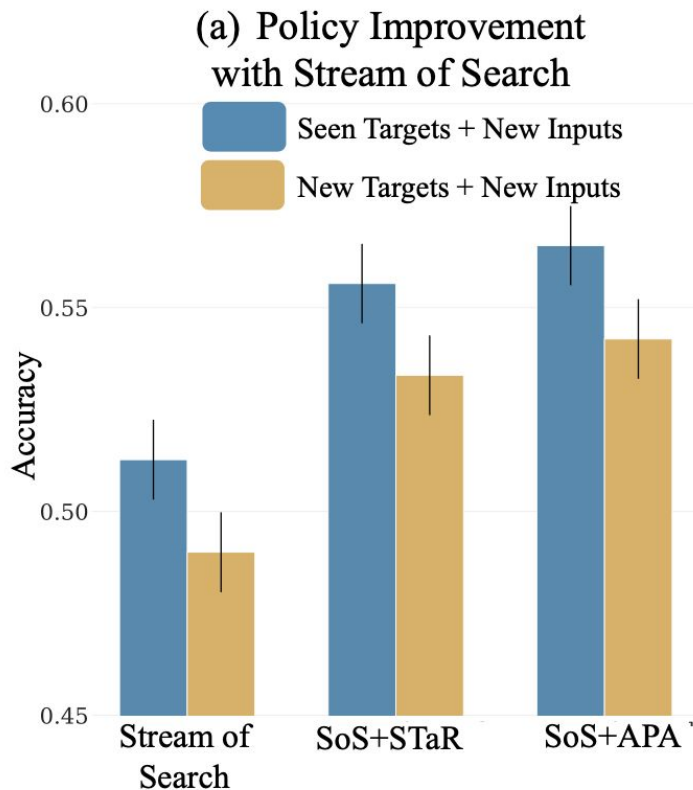
Define a straightforward reward function that takes into account the correctness and length of the generated trajectory

APA uses a reference policy to prevent policy from drifting too far from the initial

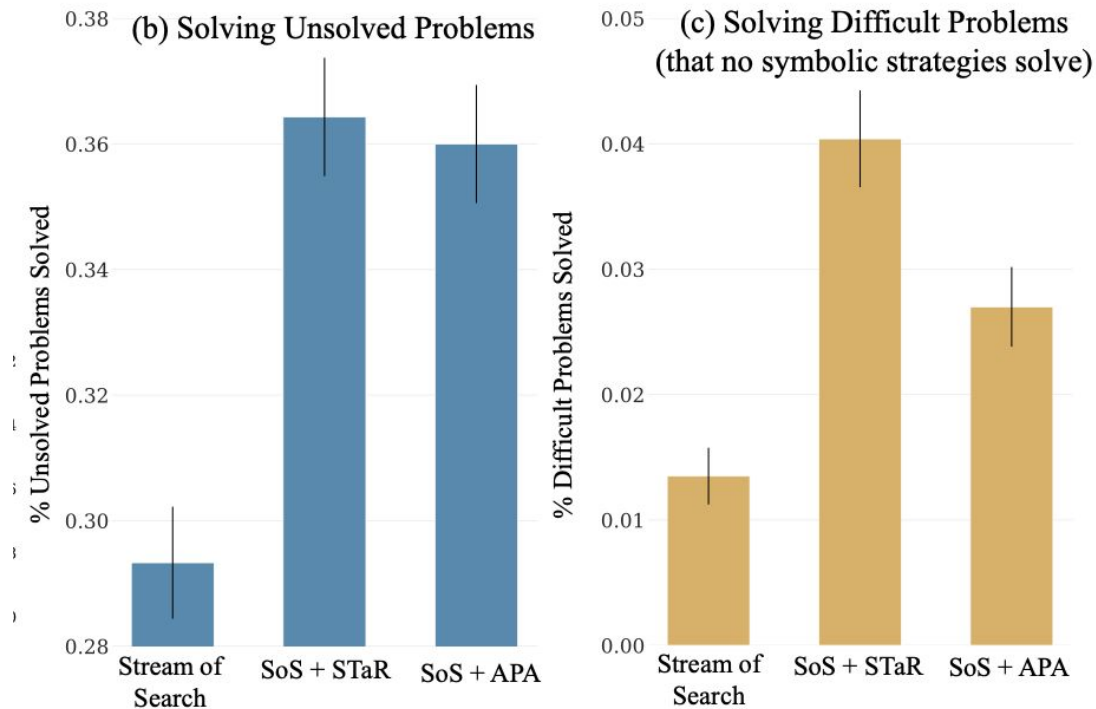
Update the reference policy whenever the validation reward converges result in further policy improvement.



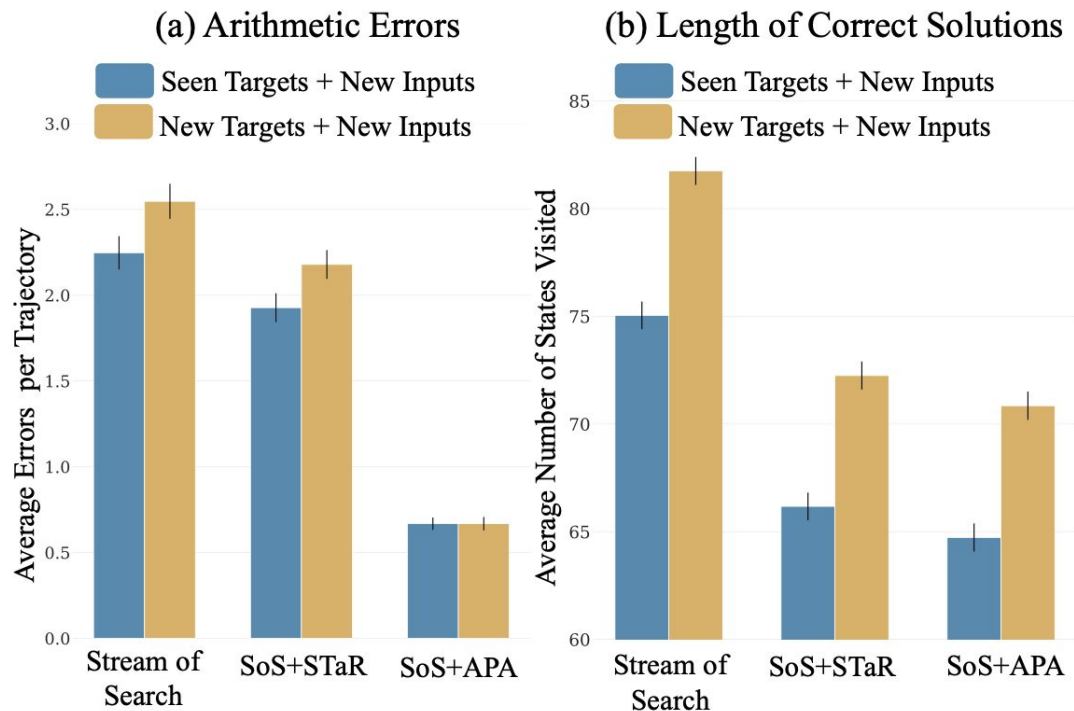
STaR and APA show similar improvements



SoS + STaR and APA models solved previously-unsolved problems & difficult problems



Policy improvement leads to fewer arithmetic errors and more efficient solutions



SoS addresses **key criticisms** of language model for search and problem-solving

- Backtracking mitigates snowballing errors
- Exploration of alternative paths and verification overcome failures in lookahead tasks
- Find that exposing models to problem-solving processes, including exploration and backtracking is crucial

Some open questions and directions

How can we **efficiently generate SoS data**, as we extend SoS to pretrained LLMs?

How do we efficiently **combine generation and verification** in text?

How do we **improve exploration**? Can backtracking and verification be emergent?



Thanks!