

Understanding Architectural Constraints on LLM Reasoning Abilities

Michael Hahn

NeurIPS Workshop Foundations of LLM Reasoning



Saarland Informatics
Campus



UNIVERSITÄT
DES
SAARLANDES

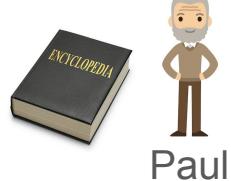
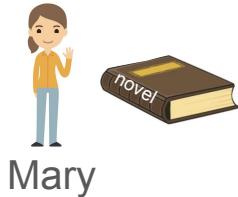
Can LLMs Track States?

Can LLMs Track States?

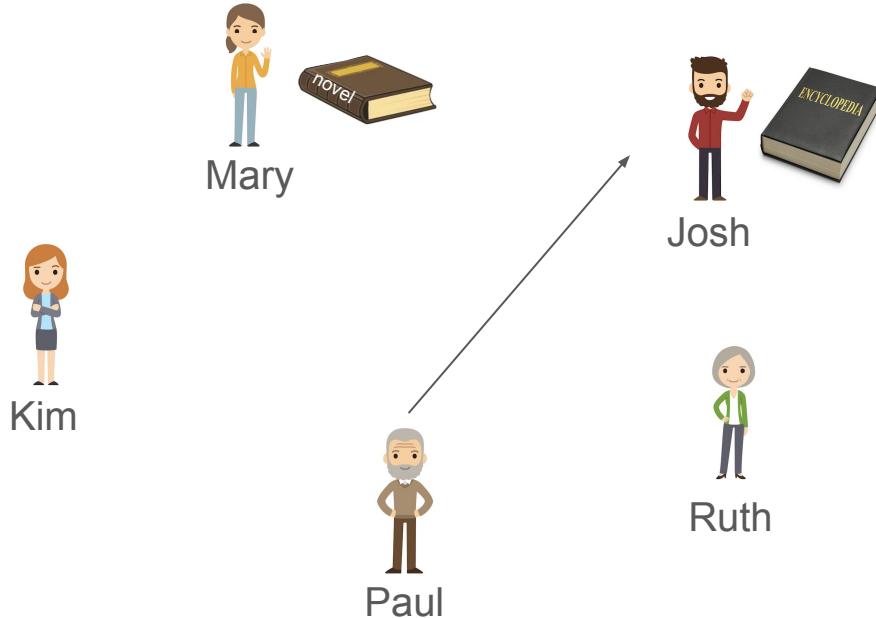


Can LLMs Track States?

Paul had an encyclopedia.
Mary had a novel.

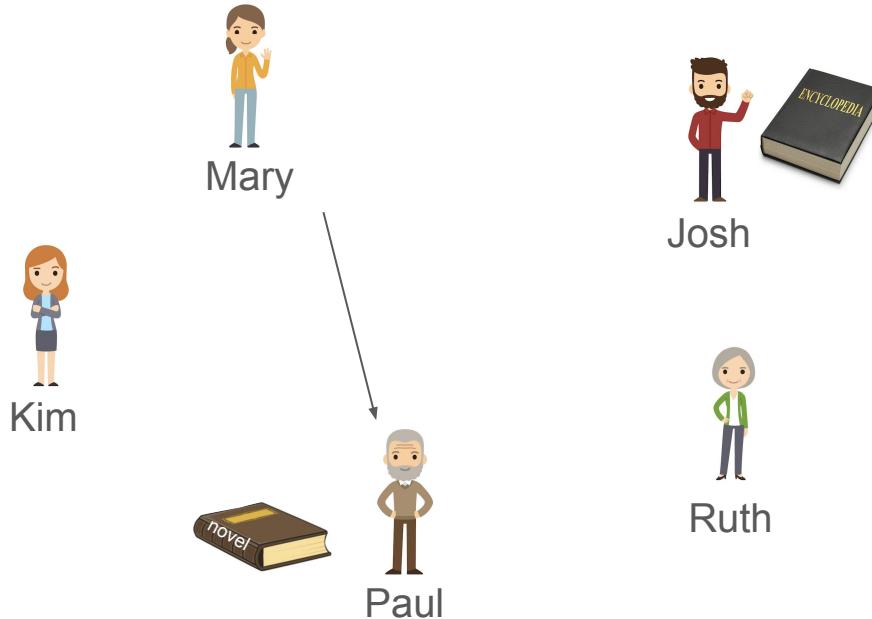


Can LLMs Track States?



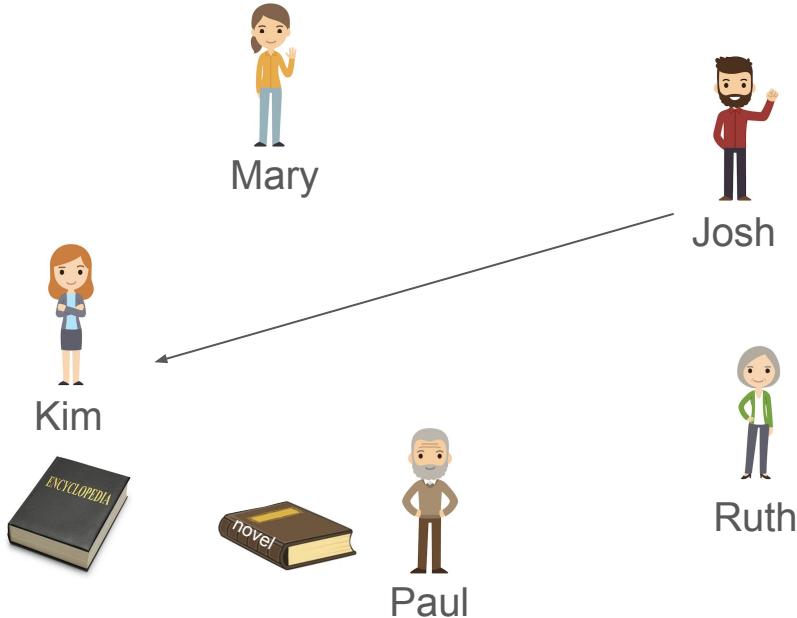
Paul had an encyclopedia.
Mary had a novel.
Paul gave his book to Josh.

Can LLMs Track States?



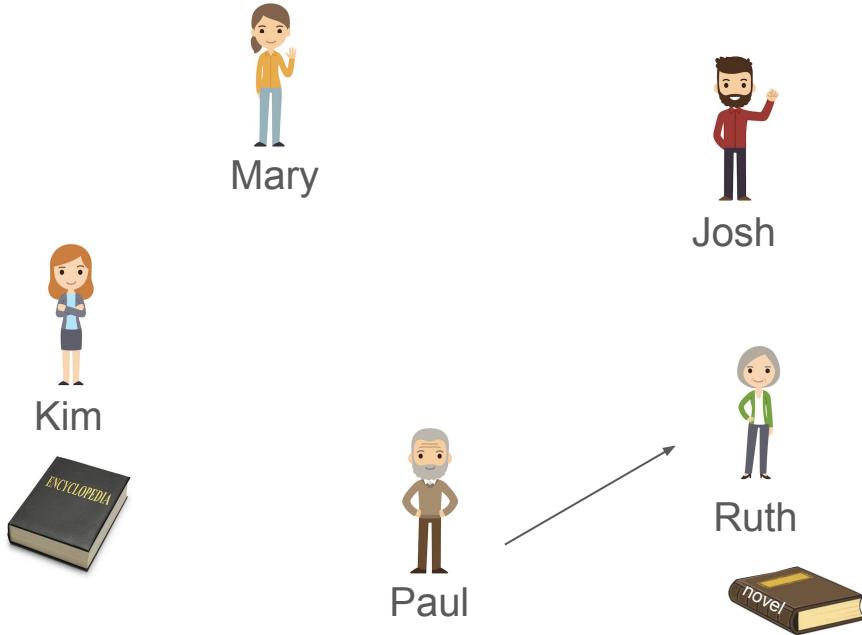
Paul had an encyclopedia.
Mary had a novel.
Paul gave his book to Josh.
Mary gave her book to Paul.

Can LLMs Track States?



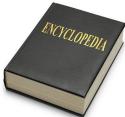
Paul had an encyclopedia.
Mary had a novel.
Paul gave his book to Josh.
Mary gave her book to Paul.
Josh gave his book to Kim.

Can LLMs Track States?



Paul had an encyclopedia.
Mary had a novel.
Paul gave his book to Josh.
Mary gave her book to Paul.
Josh gave his book to Kim.
Paul gave his book to Ruth.

Can LLMs Track States?



Paul had an encyclopedia.
Mary had a novel.
Paul gave his book to Josh.
Mary gave her book to Paul.
Josh gave his book to Kim.
Paul gave his book to Ruth.
Who has the encyclopedia?

Can LLMs Track States?



Paul had an encyclopedia.
Mary had a novel.
Paul gave his book to Josh.
Mary gave her book to Paul.
Josh gave his book to Kim.
Paul gave his book to Ruth.
Who has the encyclopedia?

Answer: Kim has the encyclopedia.

Can LLMs Track States?



Paul had an encyclopedia.

Mary had a novel.

Paul gave his book to Josh.

Mary gave her book to Paul.

Josh gave his book to Kim.

Paul gave his book to Ruth.

Who has the encyclopedia?

Answer: Kim has the encyclopedia.



Ruth has the encyclopedia.

ChatGPT-5 Instant, Aug 31, 2025.

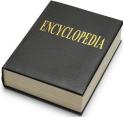
Can LLMs Track States?



Mary



Kim



1



Paul



Josh



Ruth



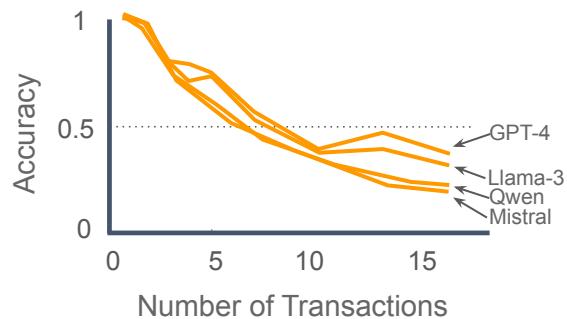
Models: gpt-4o-2024-08-06,
meta-llama/Llama-3.3-70B-Instruct-Turbo,
mistralai/Mistral-7B-Instruct-v0.3,
Qwen/Qwen2.5-72B-Instruct-Turbo

Paul had an encyclopedia.
Mary had a novel.
Paul gave his book to Josh.
Mary gave her book to Paul.
Josh gave his book to Kim.
Paul gave his book to Ruth.
Who has the encyclopedia?

Answer: Kim has the encyclopedia.



Ruth has the encyclopedia.



straight answer, no CoT

Paul had an encyclopedia.

Mary had a novel.

Paul gave his book to Josh.

Mary gave her book to Paul.

Josh gave his book to Kim.

Paul gave his book to Ruth.

Who has the encyclopedia?

Please think step by step.

Got it — let's carefully trace the movements step by step:

1. Initial ownership:

- Paul → *encyclopedia*
- Mary → *novel*

2. Paul gave his book (*encyclopedia*) to Josh:

- Josh → *encyclopedia*
- Paul → nothing

3. Mary gave her book (*novel*) to Paul:

- Paul → *novel*
- Mary → nothing

4. Josh gave his book (*encyclopedia*) to Kim:

- Kim → *encyclopedia*
- Josh → nothing

5. Paul gave his book (*novel*) to Ruth:

- Ruth → *novel*
- Paul → nothing

Final ownership:

- Kim → *encyclopedia*
- Ruth → *novel*
- Paul, Mary, Josh → nothing

correct!

Answer: Kim has the *encyclopedia*.

Paul had an *encyclopedia*.

Mary had a *novel*.

Paul gave his book to Josh.

Mary gave her book to Paul.

Josh gave his book to Kim.

Paul gave his book to Ruth.

Who has the *encyclopedia*?

Please think step by step.

Got it — let's carefully trace the movements step by step:

1. **Initial ownership:**

- Paul → *encyclopedia*
- Mary → *novel*

2. **Paul gave his book (encyclopedia) to Josh:**

- Josh → *encyclopedia*
- Paul → nothing

3. **Mary gave her book (novel) to Paul:**

- Paul → *novel*
- Mary → nothing

4. **Josh gave his book (encyclopedia) to Kim:**

- Kim → *encyclopedia*
- Josh → nothing

5. **Paul gave his book (novel) to Ruth:**

- Ruth → *novel*
- Paul → nothing

Final ownership:

- Kim → *encyclopedia*
- Ruth → *novel*
- Paul, Mary, Josh → nothing

correct!

Answer: Kim has the *encyclopedia*.

Paul had an *encyclopedia*.

Mary had a *novel*.

Paul gave his book to Josh.

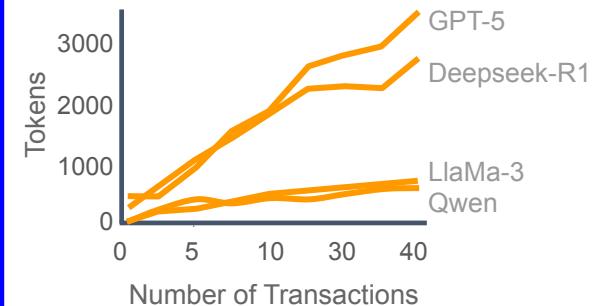
Mary gave her book to Paul.

Josh gave his book to Kim.

Paul gave his book to Ruth.

Who has the *encyclopedia*?

Please think step by step.



Models:

gpt-5-mini-2025-08-07,
meta-llama/Llama-3.3-70B-Instruct-Turbo,
deepseek-ai/DeepSeek-R1-Distill-Llama-70B,
Qwen/Qwen2.5-72B-Instruct-Turbo

Basic Algorithmic
Abilities

1-step

High-level Reasoning
Abilities

CoT

Basic Algorithmic Abilities

1-step

High-level Reasoning Abilities

CoT

Basic Algorithmic Abilities

1-step

High-level Reasoning Abilities

CoT

Which abilities can
transformers learn?

Basic Algorithmic Abilities

1-step

Which abilities can transformers learn?

High-level Reasoning Abilities

CoT

How costly is reasoning?

Basic Algorithmic Abilities

1-step

Which abilities can
transformers learn?

High-level Reasoning Abilities

CoT

How costly is
reasoning?

A FORMAL FRAMEWORK FOR UNDERSTANDING LENGTH GENERALIZATION IN TRANSFORMERS

Xinting Huang^{1*} Andy Yang^{2*} Satwik Bhattacharya³ Yash Sarrof¹
Andreas Krebs⁴ Hattie Zhou⁵ Preetum Nakkiran⁶ Michael Hahn^{1†}
¹Saarland University ²University of Notre Dame ³University of Oxford
⁴University of Tübingen ⁵Mila, Université de Montréal ⁶Apple



Born a Transformer – Always a Transformer? On the Effect of Pretraining on Architectural Abilities

Mayank Jobanputra^{1*}, Yana Veitsman^{1*}, Yash Sarrof¹, Aleksandra Bakalova¹,
Vera Demberg¹, Ellie Pavlick², Michael Hahn¹

¹Saarland University ²Brown University
{mayank,mhahn}@lst.uni-saarland.de



ICLR 2025

NeurIPS 2025

| | | | |
|---|---|---|---|
| 0 | 1 | 1 | 0 |
|---|---|---|---|

PARITY

even



| | | | |
|---|---|---|---|
| 0 | 1 | 1 | 0 |
|---|---|---|---|

PARITY

odd



| | | | |
|---|---|---|---|
| 0 | 1 | 1 | 0 |
|---|---|---|---|

PARITY

even



| | | | |
|---|---|---|---|
| 0 | 1 | 1 | 0 |
|---|---|---|---|

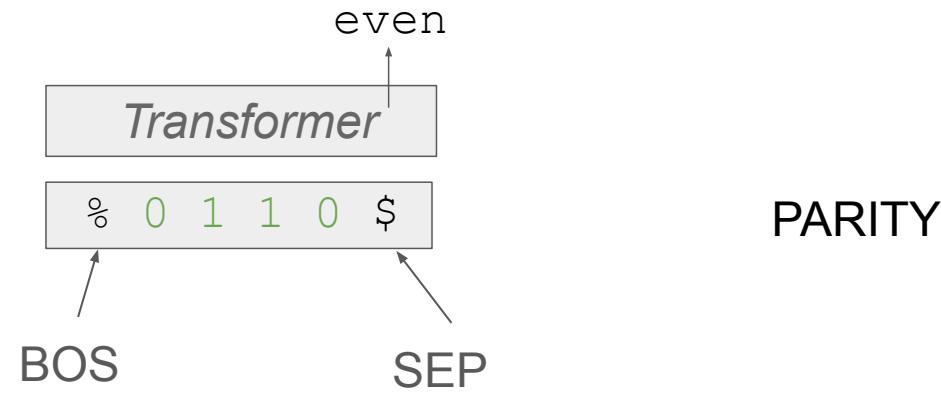
PARITY

even



| | | | |
|---|---|---|---|
| 0 | 1 | 1 | 0 |
|---|---|---|---|

PARITY



Transformer

% 0 1 1 0 \$ even @

PARITY

BOS

SEP

EOS

Transformer

% 0 1 1 0 \$ even @

PARITY

If we train transformer at
lengths $\leq N$, ...

Transformer

‰ 0 1 1 0 1 1 1 0 1 \$

If we train transformer at
lengths $\leq N$, ...

does it perform
correctly at length $2N$?

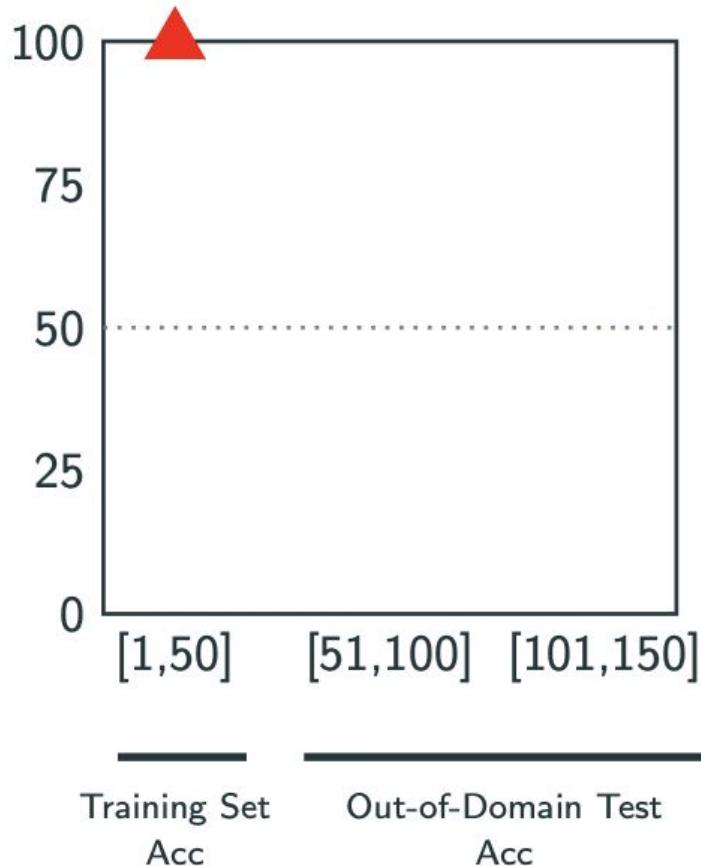
Transformer

```
% 0 1 1 0 1 1 1 0 1 $ even @
```

If we train transformer at
lengths $\leq N$, ...

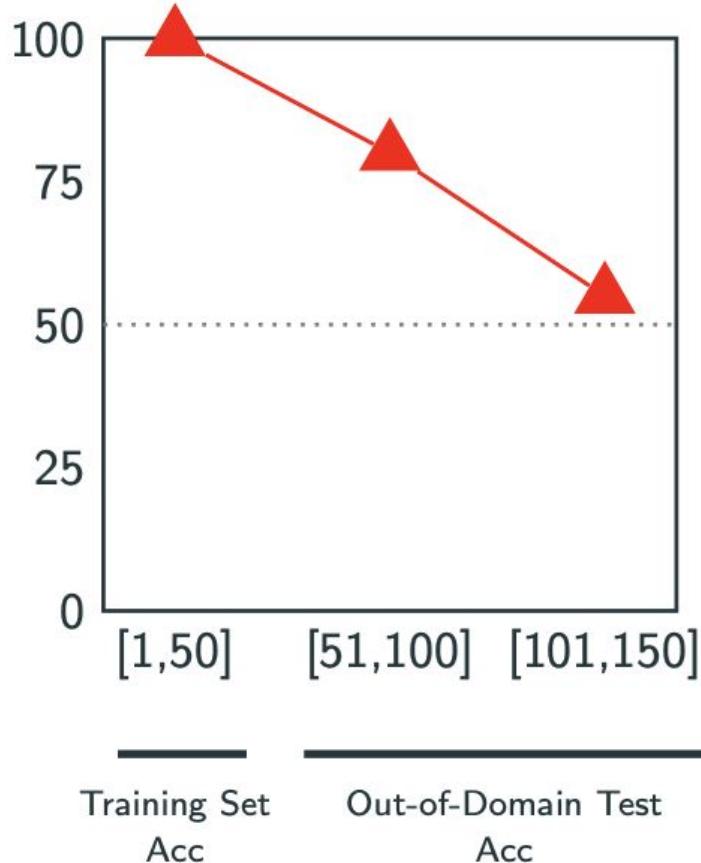
does it perform
correctly at length $2N$?

Not for
this task!



Related results: Bhattamishra et al 2020; Anil et al 2022; Zhou et al 2024.

Not for
this task!

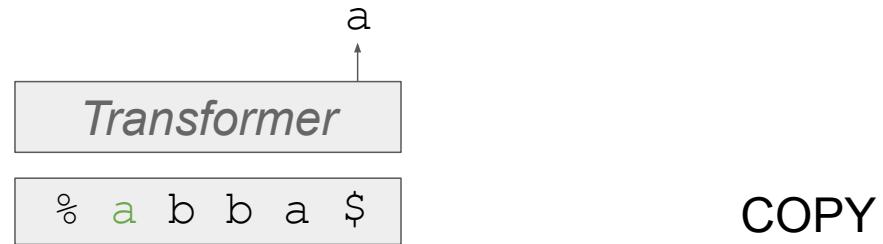


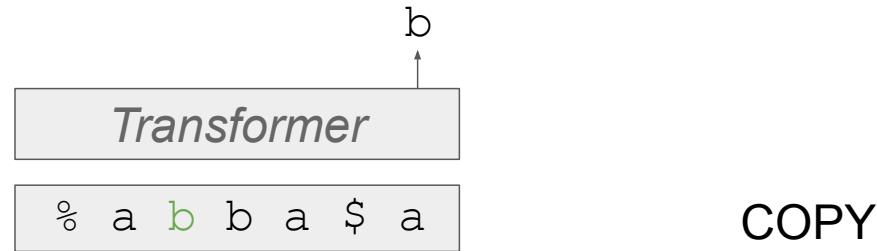
Related results: Bhattamishra et al 2020; Anil et al 2022; Zhou et al 2024.

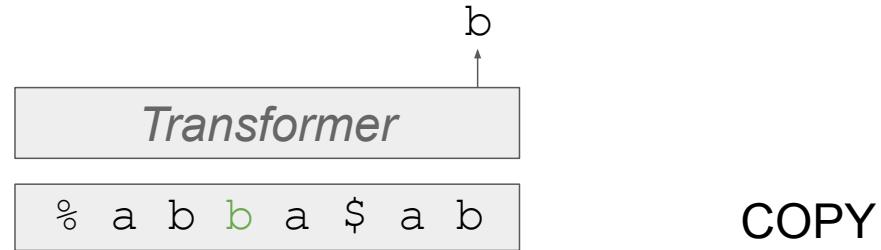
Transformer

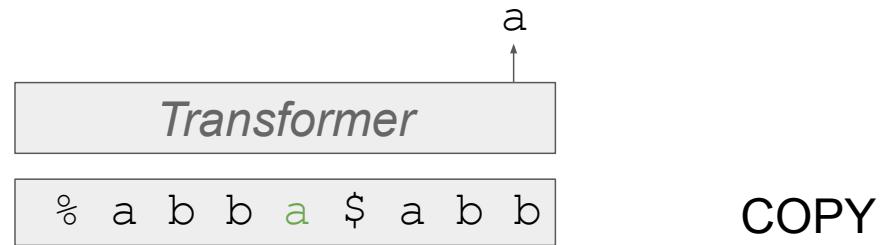
° a b b a \$

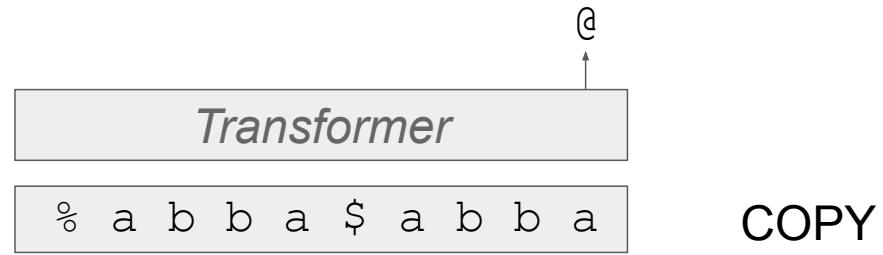
COPY











Transformer

%" a b b a \$ a b b a @

COPY

Transformer

%" a b b a \$ a b b a @

COPY

Transformer

% a b b a \$ a b b a @

COPY

If we train transformer at
lengths $\leq N$, ...

Transformer

% b a b b a b b b a \$

If we train transformer at
lengths $\leq N$, ...

does it perform
correctly at length $2N$?

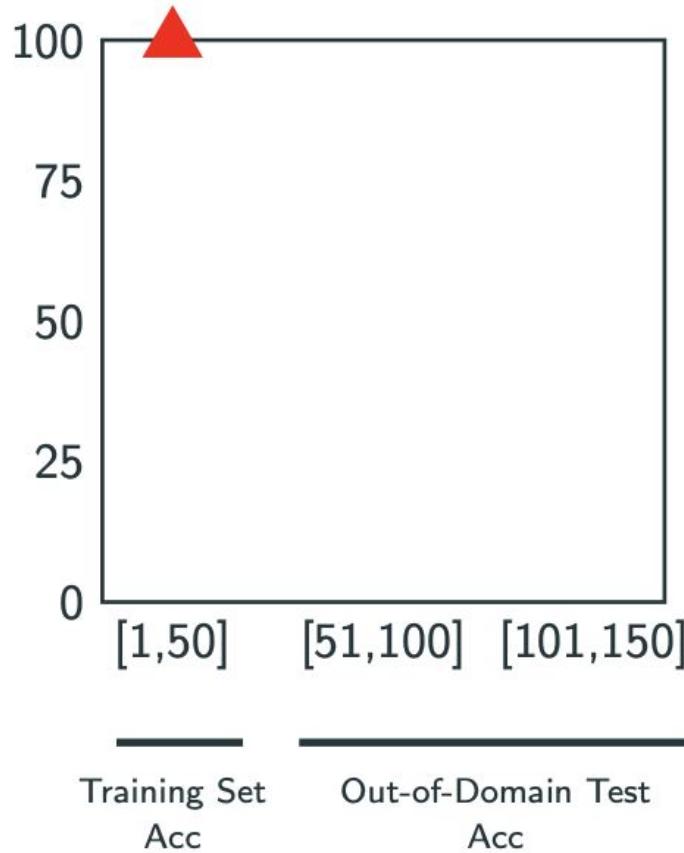
Transformer

%; b a b b a b b b a \$ b a b b a b b b a @

If we train transformer at
lengths $\leq N$, ...

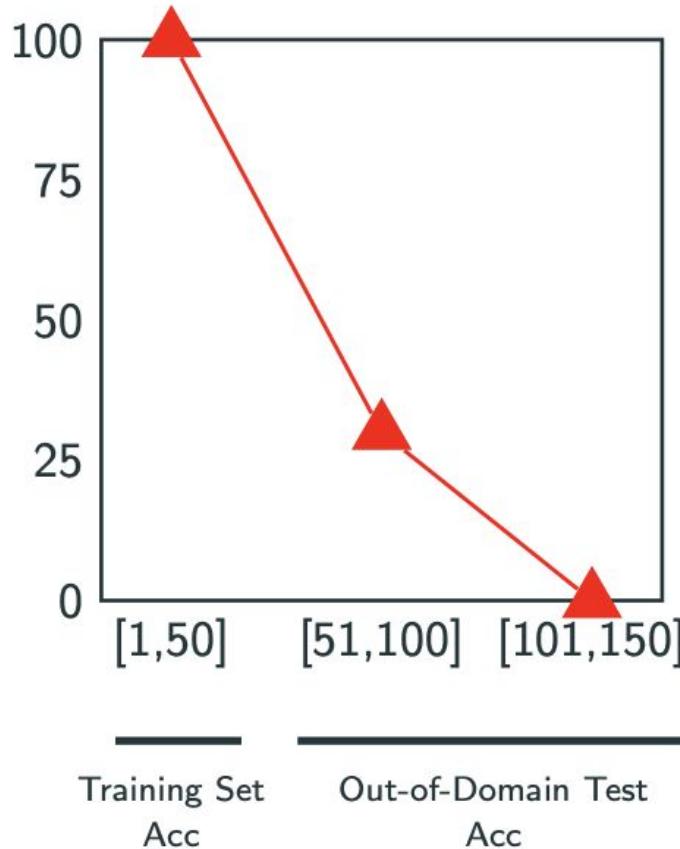
does it perform
correctly at length $2N$?

Not for
this task!



Related results: Zhou et al 2024 ICLR; Jelassi et al 2024 ICML; Kazemnejad et al 2023 NeurIPS.⁷

Not for
this task!



Related results: Zhou et al 2024 ICLR; Jelassi et al 2024 ICML; Kazemnejad et al 2023 NeurIPS.⁷

Transformer

o u r s z \$ u r s z @

COPY UNIQUE

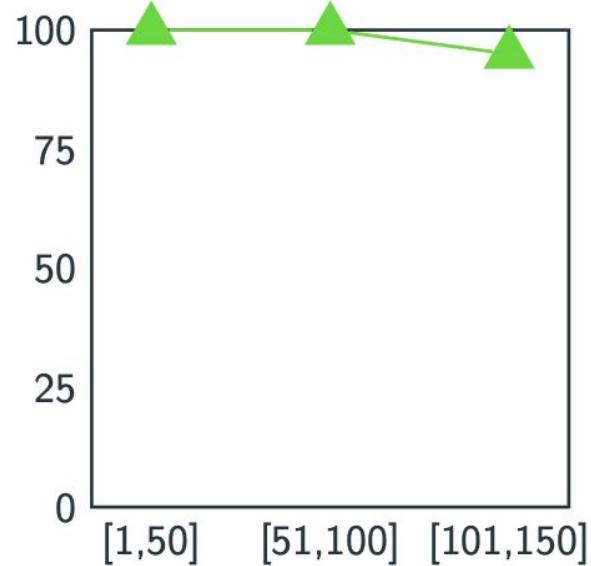
but for many
other tasks, it
works!

Transformer

° u r s z \$ u r s z @

COPY UNIQUE

but for many
other tasks, it
works!



Related results: Zhou et al 2024
ICLR; Jelassi et al 2024 ICML.8

On which tasks do transformers length-generalize?

WHAT ALGORITHMS CAN TRANSFORMERS LEARN? A STUDY IN LENGTH GENERALIZATION

Hattie Zhou^{*1,2}, Arwen Bradley¹, Eta Littwin¹, Noam Razin^{*1,3}, Omid Saremi¹,
Josh Susskind¹, Samy Bengio¹, and Preetum Nakkiran¹

WHAT ALGORITHMS CAN TRANSFORMERS LEARN?
A STUDY IN LENGTH GENERALIZATION

Hattie Zhou^{*1,2}, Arwen Bradley¹, Eta Littwin¹, Noam Razin^{*1,3}, Omid Saremi¹,
Josh Susskind¹, Samy Bengio¹, and Preetum Nakkiran¹

RASP-L Conjecture (paraphrased):
Transformers length-generalize on problems with
simple programs in the RASP-L language.

WHAT ALGORITHMS CAN TRANSFORMERS LEARN?
A STUDY IN LENGTH GENERALIZATION

Hattie Zhou^{*1,2}, Arwen Bradley¹, Eta Littwin¹, Noam Razin^{*1,3}, Omid Saremi¹,
Josh Susskind¹, Samy Bengio¹, and Preetum Nakkiran¹

RASP-L Conjecture (paraphrased):
Transformers length-generalize on problems with
simple programs in the RASP-L language.

Challenge: RASP-L hasn't been formalized. Expressiveness not understood.

WHAT ALGORITHMS CAN TRANSFORMERS LEARN?
A STUDY IN LENGTH GENERALIZATION

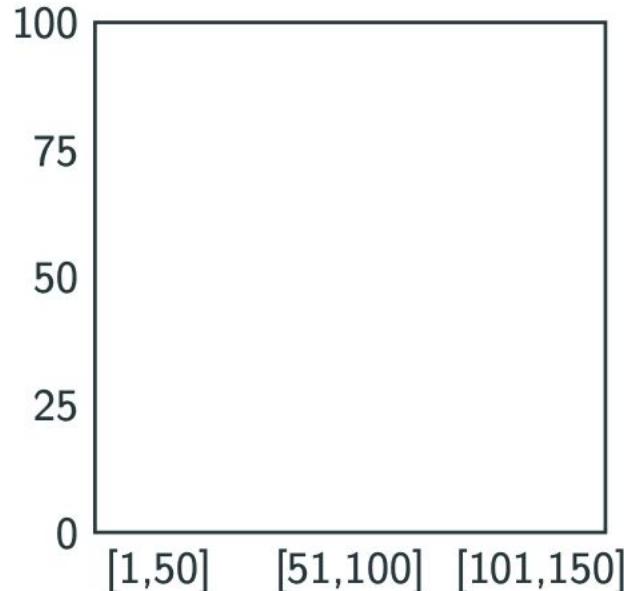
Hattie Zhou^{*1,2}, Arwen Bradley¹, Eta Littwin¹, Noam Razin^{*1,3}, Omid Saremi¹,
Josh Susskind¹, Samy Bengio¹, and Preetum Nakkiran¹

RASP-L Conjecture (paraphrased):
Transformers length-generalize on problems with
simple programs in the RASP-L language.

Challenge: RASP-L hasn't been formalized. Expressiveness not understood.

Our First Contribution: Formalize it based on C-RASP.

Binary Majority

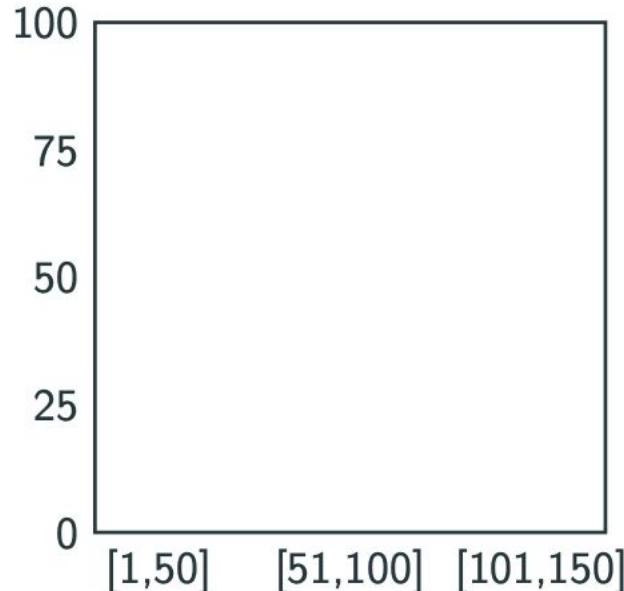


%, 0 0 1 1 0 1 1 1 0 \$

Training Set
Acc

Out-of-Domain Test
Acc

Binary Majority

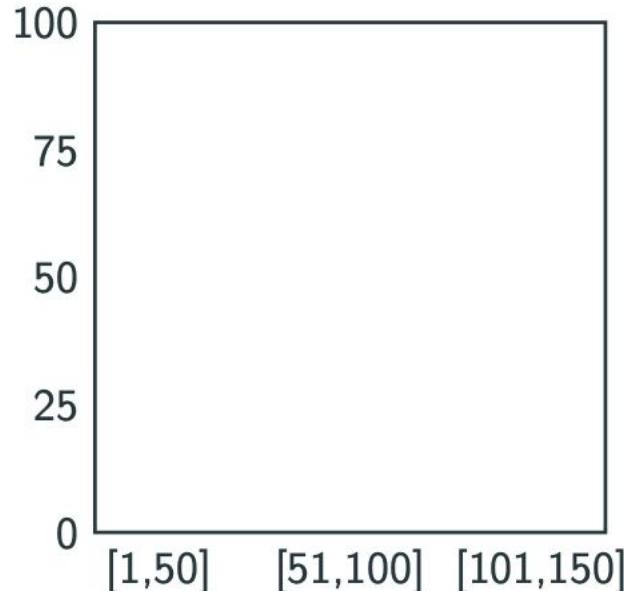


%, 0 0 1 1 0 1 1 1 0 \$

Training Set
Acc

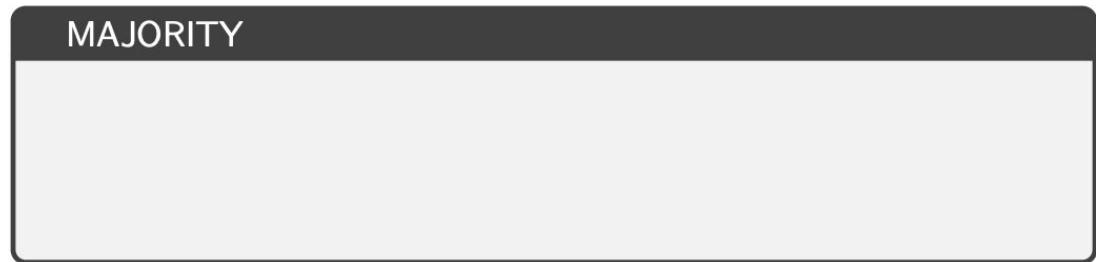
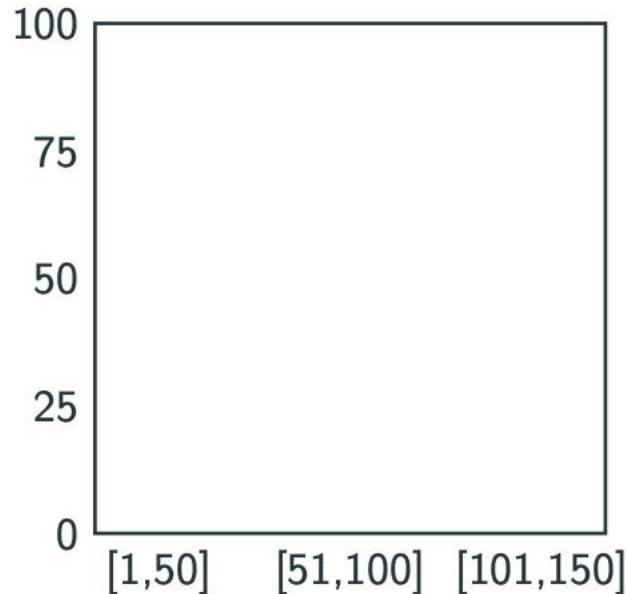
Out-of-Domain Test
Acc

Binary Majority



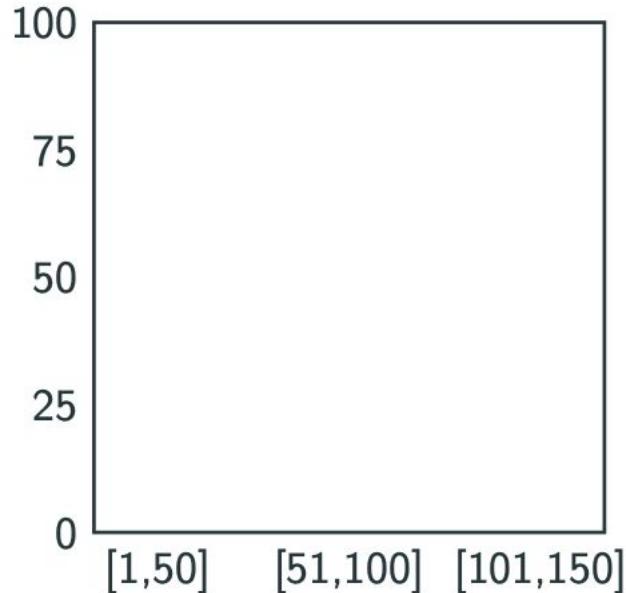
₡ 0 0 1 1 0 1 1 1 0 \$ T @

Binary Majority



₡ 0 0 1 1 0 1 1 1 0 \$

Binary Majority



MAJORITY

$C_1(i) := \# [j \leq i] Q_1(j)$

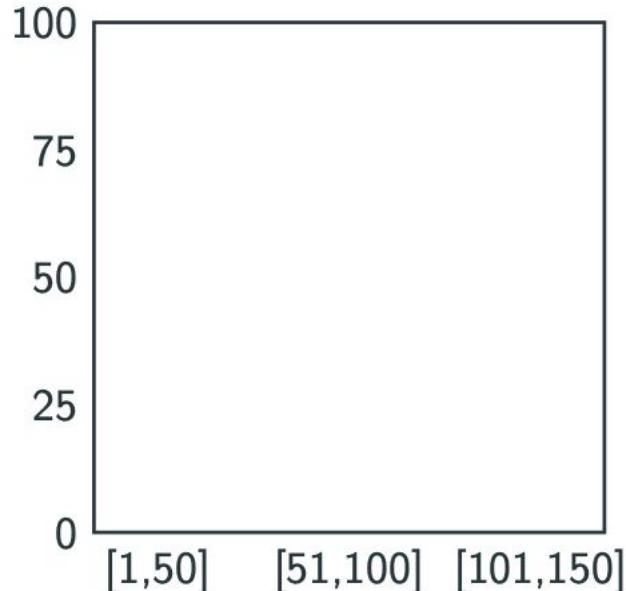
count # of 1's (1)

₡ 0 0 1 1 0 1 1 1 0 \$

Training Set
Acc

Out-of-Domain Test
Acc

Binary Majority



MAJORITY

$$C_1(i) := \# [j \leq i] Q_1(j)$$

count # of 1's (1)

0 0 1 1 0 1 1 1 0 \$

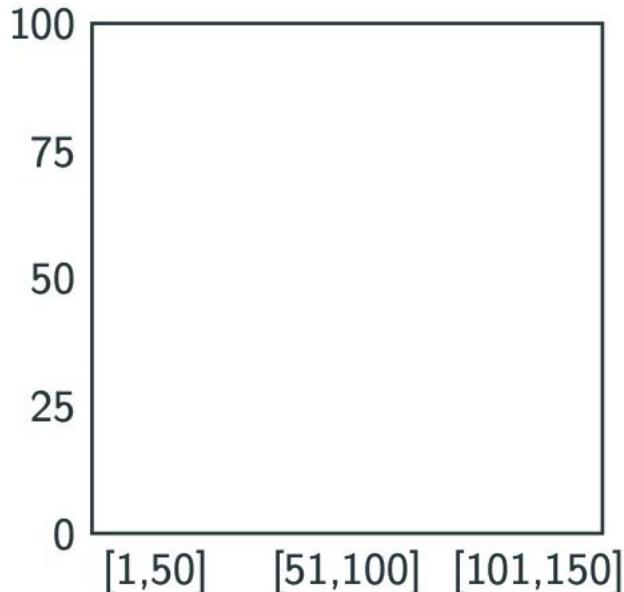
$$C_1(i) \ 0 \ 0$$

running count of 1s

Training Set
Acc

Out-of-Domain Test
Acc

Binary Majority



MAJORITY

$$C_1(i) := \# [j \leq i] Q_1(j)$$

count # of 1's (1)

₡ 0 0 1 1 0 1 1 1 0 \$

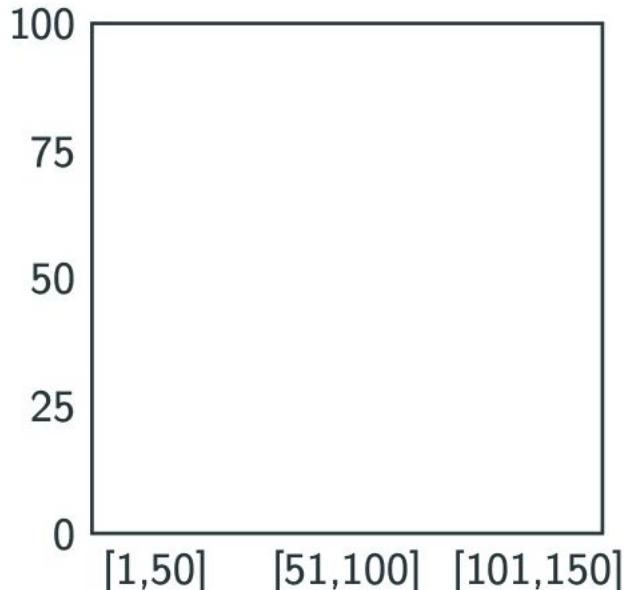
$$C_1(i) \quad 0 \quad 0 \quad 0$$

running count of 1s

Training Set
Acc

Out-of-Domain Test
Acc

Binary Majority



MAJORITY

$$C_1(i) := \# [j \leq i] Q_1(j)$$

count # of 1's (1)

օ 0 0 1 1 0 1 1 1 0 \$

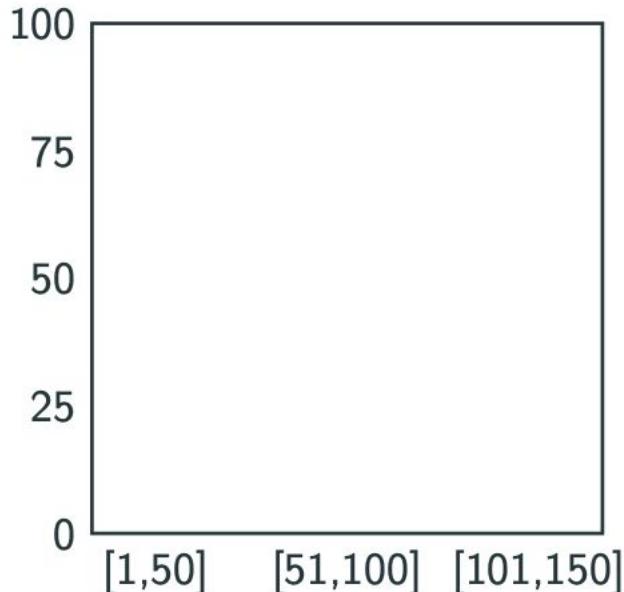
$$C_1(i) \quad 0 \quad 0 \quad 0 \quad 1$$

running count of 1s

Training Set
Acc

Out-of-Domain Test
Acc

Binary Majority



MAJORITY

$$C_1(i) := \# [j \leq i] Q_1(j)$$

count # of 1's (1)

օ 0 0 1 1 0 1 1 1 0 \$

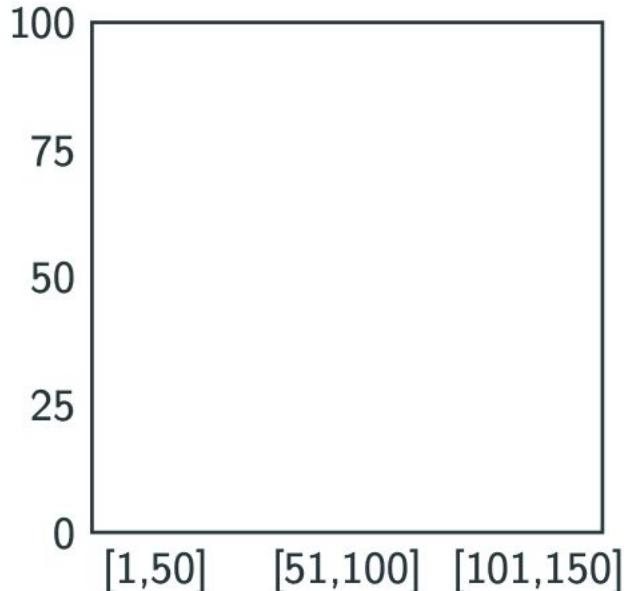
$$C_1(i) \quad 0 \quad 0 \quad 0 \quad 1 \quad 2$$

running count of 1s

Training Set
Acc

Out-of-Domain Test
Acc

Binary Majority



MAJORITY

$$C_1(i) := \# [j \leq i] Q_1(j)$$

count # of 1's (1)

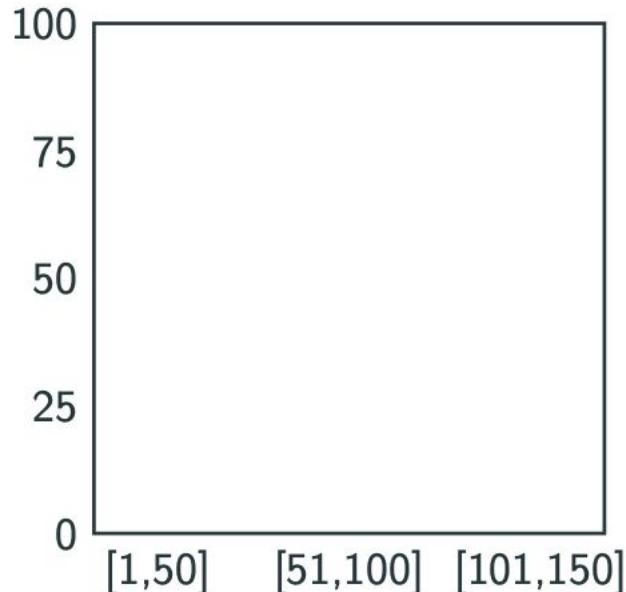
$$\begin{array}{r} \% \ 0 \ 0 \ 1 \ 1 \ \mathbf{0} \ 1 \ 1 \ 1 \ 0 \ \$ \\ \hline C_1(i) \ 0 \ 0 \ 0 \ 1 \ 2 \ 2 \end{array}$$

running count of 1s

Training Set
Acc

Out-of-Domain Test
Acc

Binary Majority



MAJORITY

$$C_1(i) := \# [j \leq i] Q_1(j)$$

count # of 1's (1)

₡ 0 0 1 1 0 **1** 1 1 0 \$

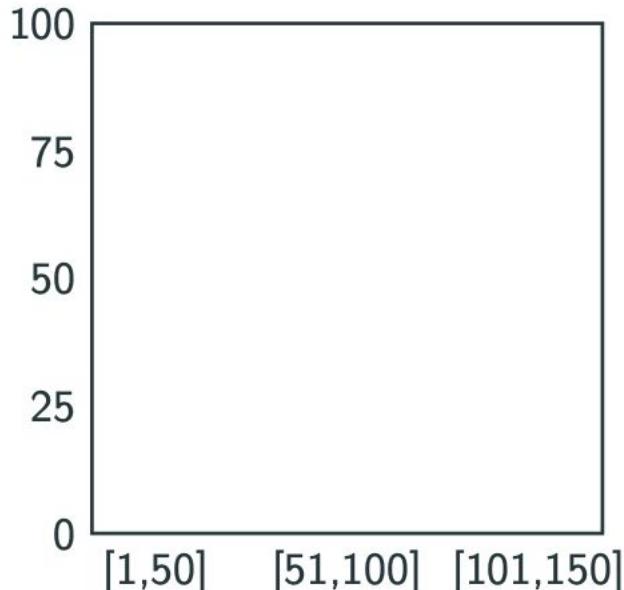
$$C_1(i) \quad 0 \quad 0 \quad 0 \quad 1 \quad 2 \quad 2 \quad 3$$

running count of 1s

Training Set
Acc

Out-of-Domain Test
Acc

Binary Majority



MAJORITY

$$C_1(i) := \# [j \leq i] Q_1(j)$$

count # of 1's (1)

₡ 0 0 1 1 0 1 1 0 \$

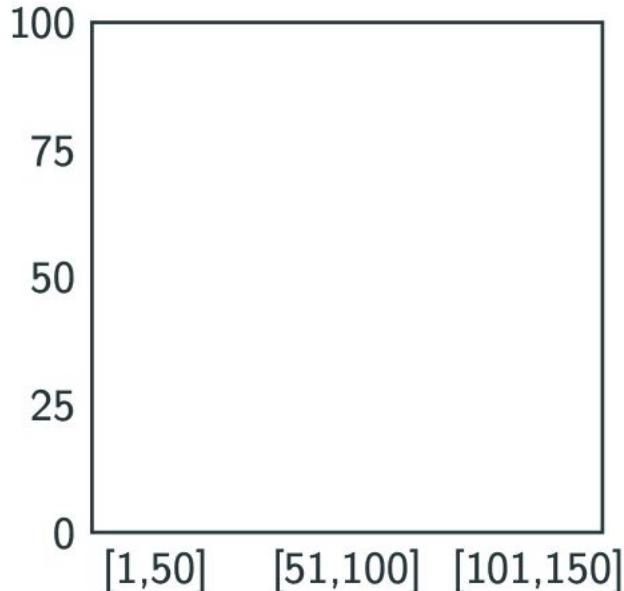
$$C_1(i) \quad 0 \quad 0 \quad 0 \quad 1 \quad 2 \quad 2 \quad 3 \quad 4$$

running count of 1s

Training Set
Acc

Out-of-Domain Test
Acc

Binary Majority



MAJORITY

$$C_1(i) := \# [j \leq i] Q_1(j)$$

count # of 1's (1)

₡ 0 0 1 1 0 1 1 1 0 \$

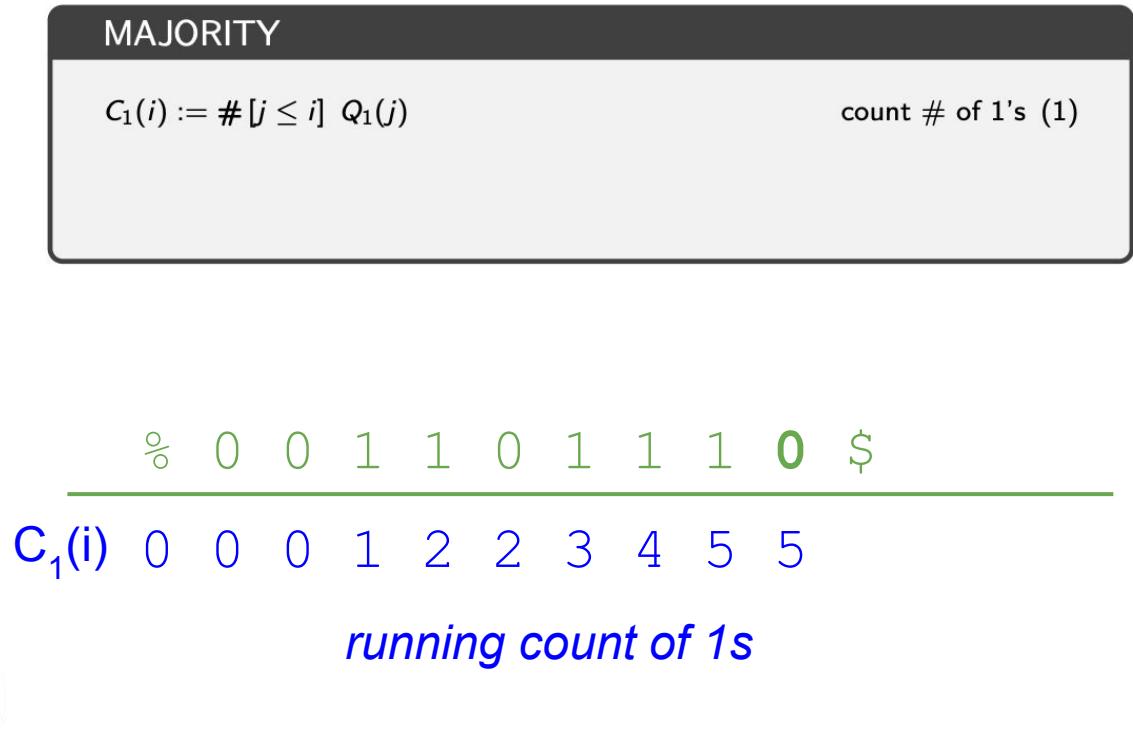
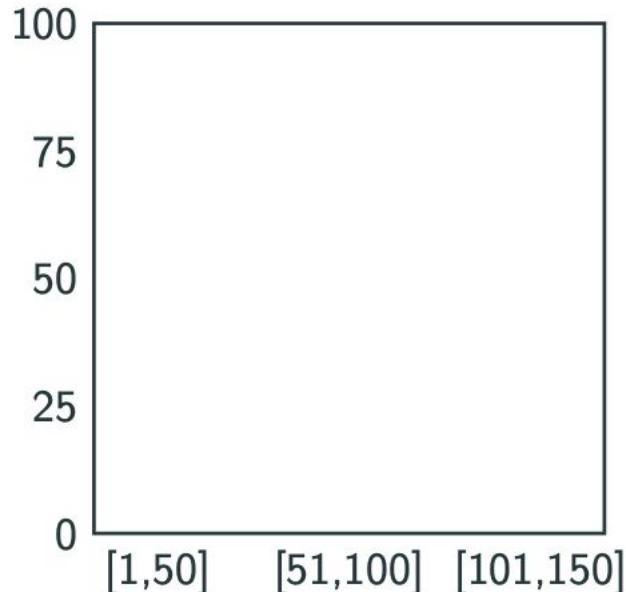
$$C_1(i) \quad 0 \quad 0 \quad 0 \quad 1 \quad 2 \quad 2 \quad 3 \quad 4 \quad 5$$

running count of 1s

Training Set
Acc

Out-of-Domain Test
Acc

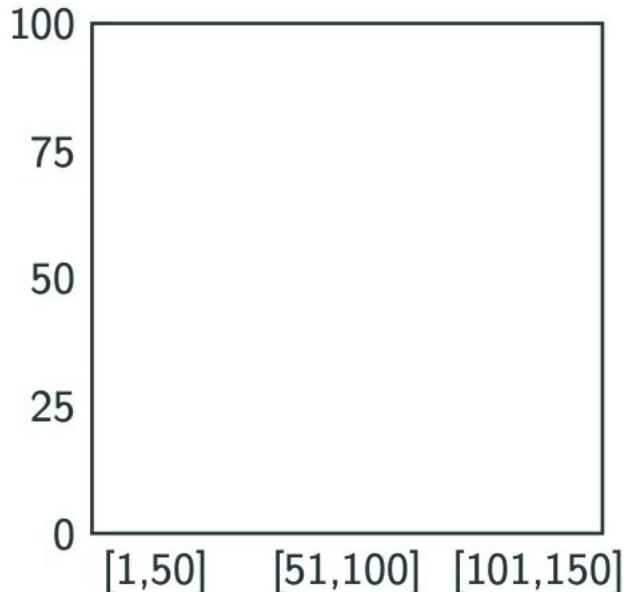
Binary Majority



Training Set
Acc

Out-of-Domain Test
Acc

Binary Majority



MAJORITY

$$C_1(i) := \# [j \leq i] Q_1(j)$$

count # of 1's (1)

0 0 0 1 1 0 1 1 1 0 \$

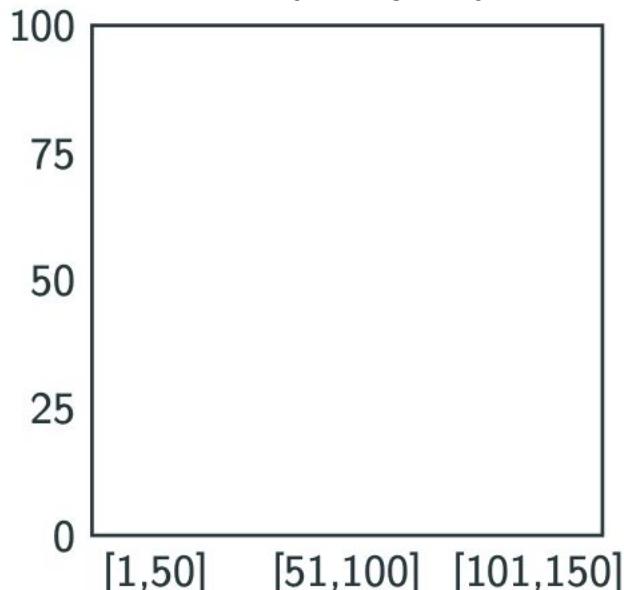
$$C_1(i) \quad 0 \quad 0 \quad 0 \quad 1 \quad 2 \quad 2 \quad 3 \quad 4 \quad 5 \quad 5 \quad 5$$

running count of 1s

Training Set
Acc

Out-of-Domain Test
Acc

Binary Majority



MAJORITY

$$C_1(i) := \# [j \leq i] Q_1(j)$$

count # of 1's (1)

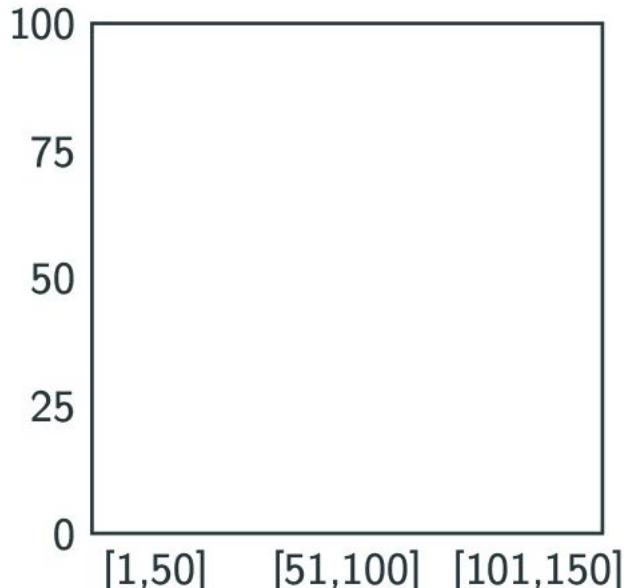
$$C_0(i) := \# [j \leq i] Q_0(j)$$

count # of 0's (2)

$$\begin{array}{r} \% \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad \$ \\ \hline C_1(i) \quad 0 \quad 0 \quad 0 \quad 1 \quad 2 \quad 2 \quad 3 \quad 4 \quad 5 \quad 5 \\ C_0(i) \quad 0 \end{array}$$

running count of 0s

Binary Majority



MAJORITY

$$C_1(i) := \# [j \leq i] Q_1(j)$$

count # of 1's (1)

$$C_0(i) := \# [j \leq i] Q_0(j)$$

count # of 0's (2)

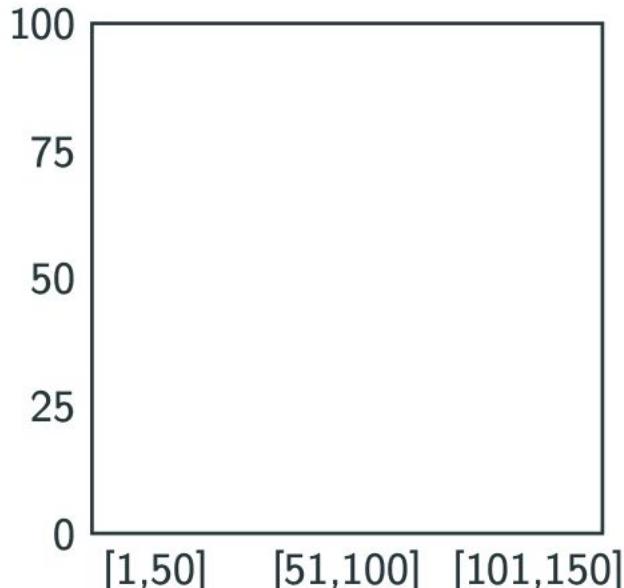
$$\begin{array}{r} \% \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad \$ \\ \hline C_1(i) \quad 0 \quad 0 \quad 0 \quad 1 \quad 2 \quad 2 \quad 3 \quad 4 \quad 5 \quad 5 \\ C_0(i) \quad 0 \quad 1 \end{array}$$

running count of 0s

Training Set
Acc

Out-of-Domain Test
Acc

Binary Majority



MAJORITY

$$C_1(i) := \# [j \leq i] Q_1(j)$$

count # of 1's (1)

$$C_0(i) := \# [j \leq i] Q_0(j)$$

count # of 0's (2)

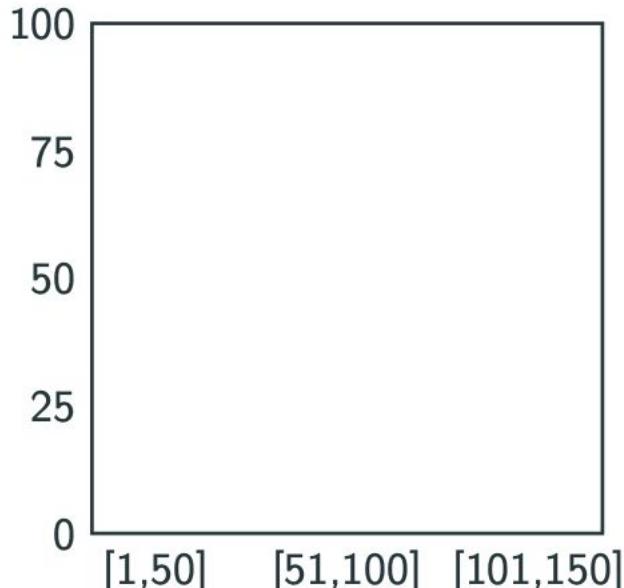
$$\begin{array}{r} \% \quad 0 \quad \mathbf{0} \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad \$ \\ \hline C_1(i) \quad 0 \quad 0 \quad 0 \quad 1 \quad 2 \quad 2 \quad 3 \quad 4 \quad 5 \quad 5 \\ C_0(i) \quad 0 \quad 1 \quad 2 \end{array}$$

running count of 0s

Training Set
Acc

Out-of-Domain Test
Acc

Binary Majority



MAJORITY

$$C_1(i) := \# [j \leq i] Q_1(j)$$

count # of 1's (1)

$$C_0(i) := \# [j \leq i] Q_0(j)$$

count # of 0's (2)

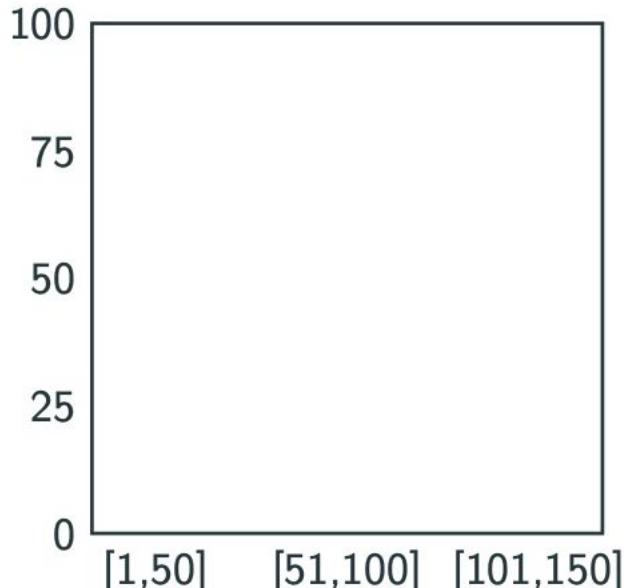
| | | | | | | | | | | |
|----------|---|---|---|---|----------|---|---|---|---|----|
| % | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | \$ |
| $C_1(i)$ | 0 | 0 | 0 | 1 | 2 | 2 | 3 | 4 | 5 | 5 |
| $C_0(i)$ | 0 | 1 | 2 | 2 | 2 | 3 | | | | |

running count of 0s

Training Set
Acc

Out-of-Domain Test
Acc

Binary Majority



MAJORITY

$$C_1(i) := \# [j \leq i] Q_1(j)$$

count # of 1's (1)

$$C_0(i) := \# [j \leq i] Q_0(j)$$

count # of 0's (2)

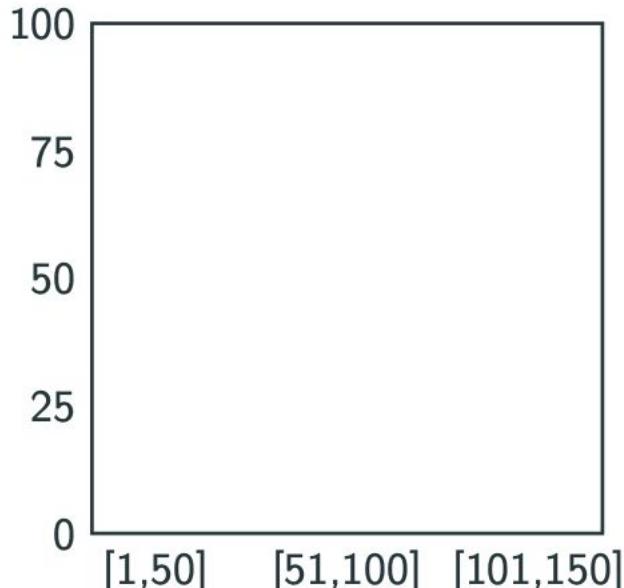
| | | | | | | | | | | | |
|----------|---|---|---|---|---|---|---|---|---|---|----|
| | % | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | \$ |
| $C_1(i)$ | 0 | 0 | 0 | 1 | 2 | 2 | 3 | 4 | 5 | 5 | 5 |
| $C_0(i)$ | 0 | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 4 | |

running count of 0s

Training Set
Acc

Out-of-Domain Test
Acc

Binary Majority



MAJORITY

$$C_1(i) := \# [j \leq i] Q_1(j)$$

count # of 1's (1)

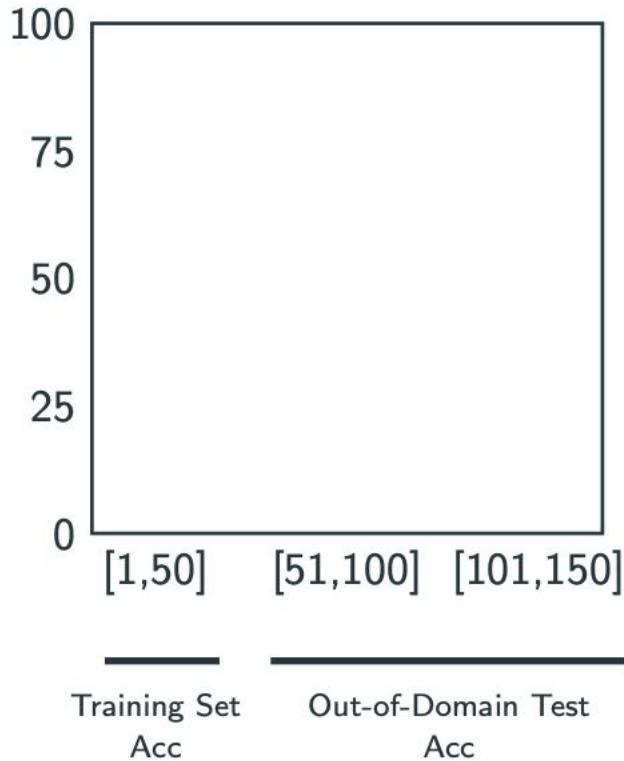
$$C_0(i) := \# [j \leq i] Q_0(j)$$

count # of 0's (2)

| | | | | | | | | | | |
|----------|---|---|---|---|---|---|---|---|---|----|
| % | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | \$ |
| $C_1(i)$ | 0 | 0 | 0 | 1 | 2 | 2 | 3 | 4 | 5 | 5 |
| $C_0(i)$ | 0 | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 4 | 4 |

running count of 0s

Binary Majority



MAJORITY

$$C_1(i) := \# [j \leq i] Q_1(j)$$

count # of 1's (1)

$$C_0(i) := \# [j \leq i] Q_0(j)$$

count # of 0's (2)

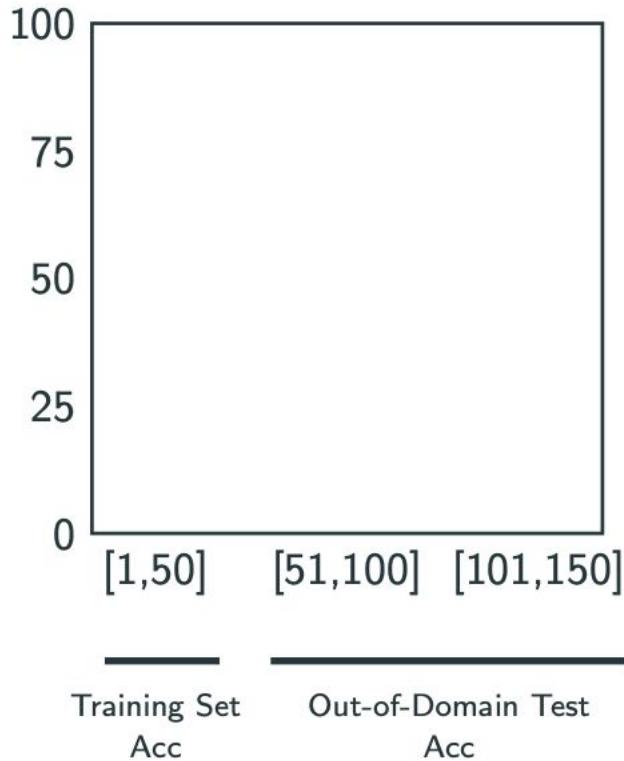
$$M(i) := C_1(i) \geq C_0(i)$$

compare them (3)

| | | | | | | | | | | |
|----------|----------|----------|----------|----------|---|---|---|---|---|----|
| % | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | \$ |
| $C_1(i)$ | 0 | 0 | 0 | 1 | 2 | 2 | 3 | 4 | 5 | 5 |
| $C_0(i)$ | 0 | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 4 | 4 |
| $M(i)$ | F | F | F | F | | | | | | |

compare them

Binary Majority



MAJORITY

$$C_1(i) := \# [j \leq i] Q_1(j)$$

count # of 1's (1)

$$C_0(i) := \# [j \leq i] Q_0(j)$$

count # of 0's (2)

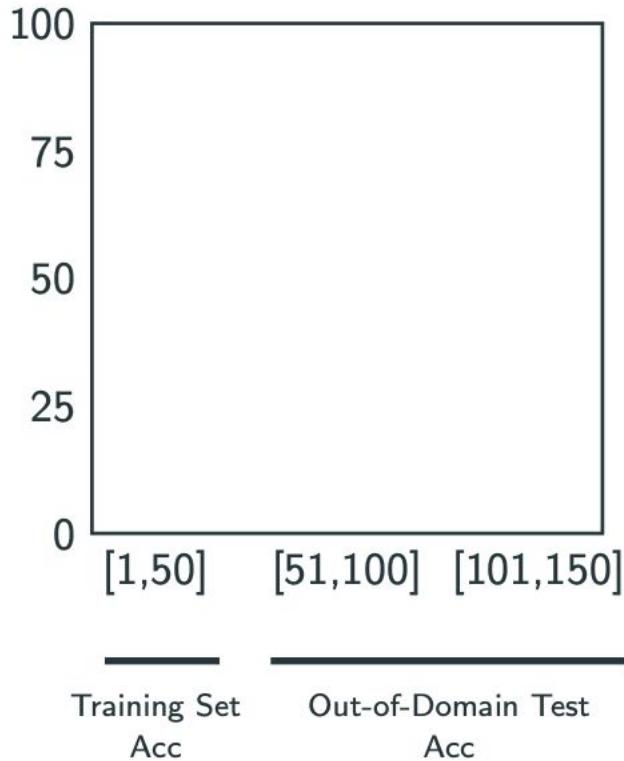
$$M(i) := C_1(i) \geq C_0(i)$$

compare them (3)

| | | | | | | | | | | | |
|----------|---|---|---|---|---|---|---|---|---|---|----|
| | % | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | \$ |
| $C_1(i)$ | 0 | 0 | 0 | 1 | 2 | 2 | 3 | 4 | 5 | 5 | 5 |
| $C_0(i)$ | 0 | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 4 | 4 | 4 |
| $M(i)$ | T | F | F | F | T | F | T | T | T | T | T |

compare them

Binary Majority



MAJORITY

$$C_1(i) := \# [j \leq i] Q_1(j)$$

count # of 1's (1)

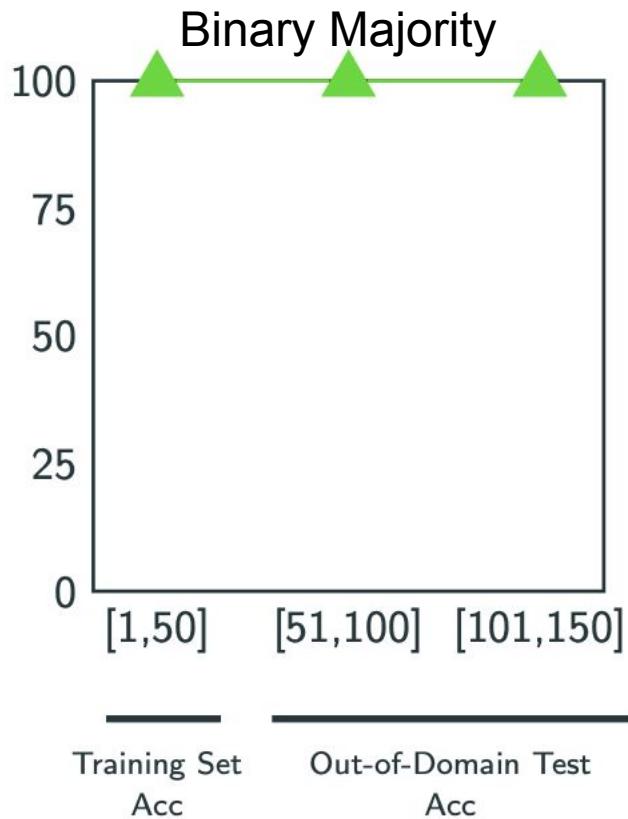
$$C_0(i) := \# [j \leq i] Q_0(j)$$

count # of 0's (2)

$$M(i) := C_1(i) \geq C_0(i)$$

compare them (3)

| | % | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | \$ | T | @ |
|----------|---|---|---|---|---|---|---|---|---|---|----|----------|---|
| $C_1(i)$ | 0 | 0 | 0 | 1 | 2 | 2 | 3 | 4 | 5 | 5 | 5 | | |
| $C_0(i)$ | 0 | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 4 | 4 | | |
| $M(i)$ | T | F | F | F | T | F | T | T | T | T | T | T | |



MAJORITY

$$C_1(i) := \# [j \leq i] Q_1(j)$$

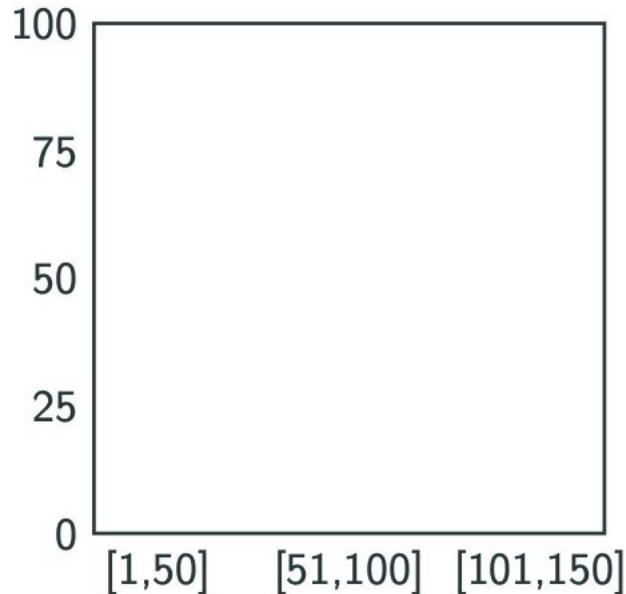
$$C_0(i) := \# [j \leq i] Q_0(j)$$

$$M(i) := C_1(i) \geq C_0(i)$$

count # of 1's (1)
 count # of 0's (2)
 compare them (3)

| | % | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | \$ | T | @ |
|----------|---|---|---|---|---|---|---|---|---|---|----|---|---|
| $C_1(i)$ | 0 | 0 | 0 | 1 | 2 | 2 | 3 | 4 | 5 | 5 | 5 | | |
| $C_0(i)$ | 0 | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 4 | 4 | | |
| $M(i)$ | T | F | F | F | T | F | T | T | T | T | T | | |

Unique Copy



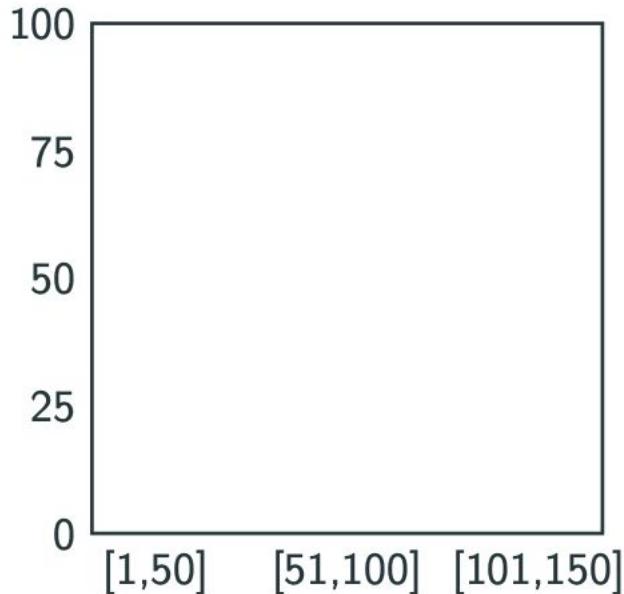
%, u, r, s, z, \$, u, ...

collect
immediately
preceding
token

Training Set
Acc

Out-of-Domain Test
Acc

Unique Copy



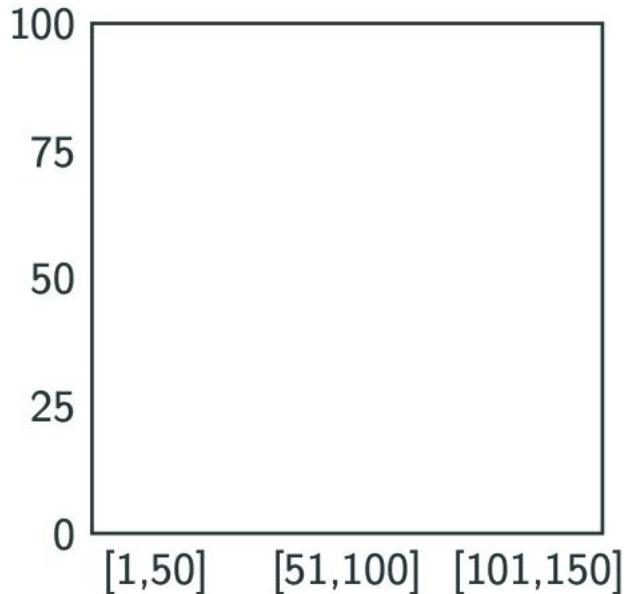
| | % | u | r | s | z | \$ | u | ... |
|----------|---|---|---|---|---|----|---|-----|
| prev = u | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

collect
immediately
preceding
token

Training Set
Acc

Out-of-Domain Test
Acc

Unique Copy



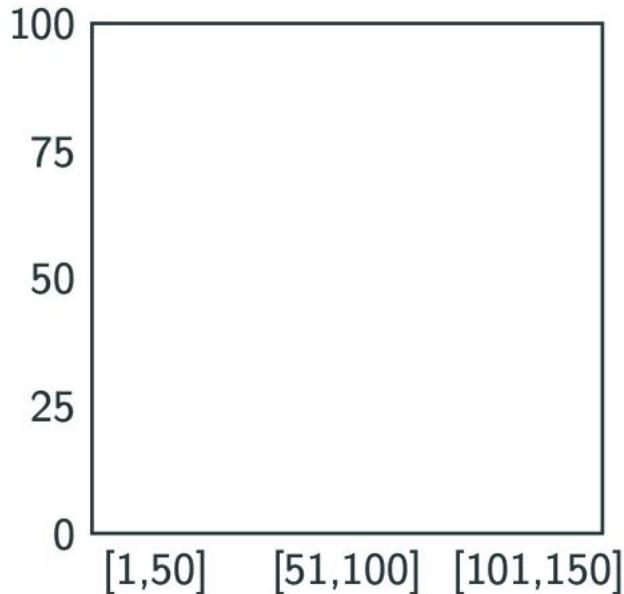
Training Set
Acc

Out-of-Domain Test
Acc

| | % | u | r | s | z | \$ | u | ... |
|----------|---|---|---|---|---|----|---|-----|
| prev = u | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| prev = r | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

collect
immediately
preceding
token

Unique Copy



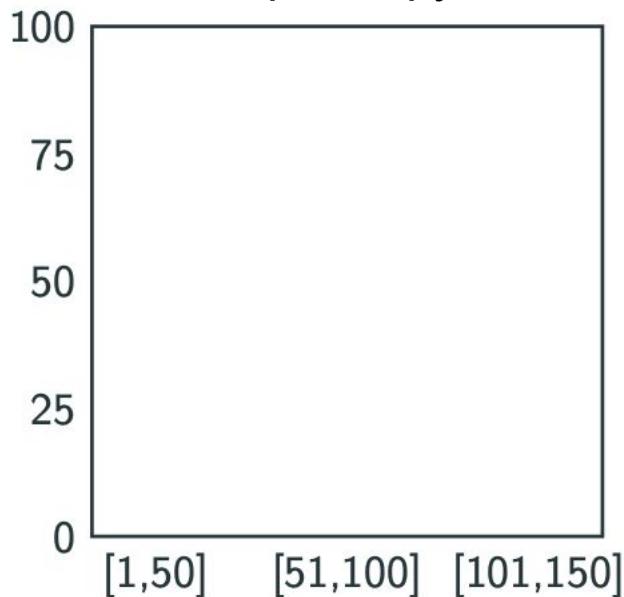
Training Set
Acc

Out-of-Domain Test
Acc

| | % | u | r | s | z | \$ | u | ... |
|----------|---|---|---|---|---|----|---|-----|
| prev = u | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| prev = r | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| prev = s | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

collect
immediately
preceding
token

Unique Copy



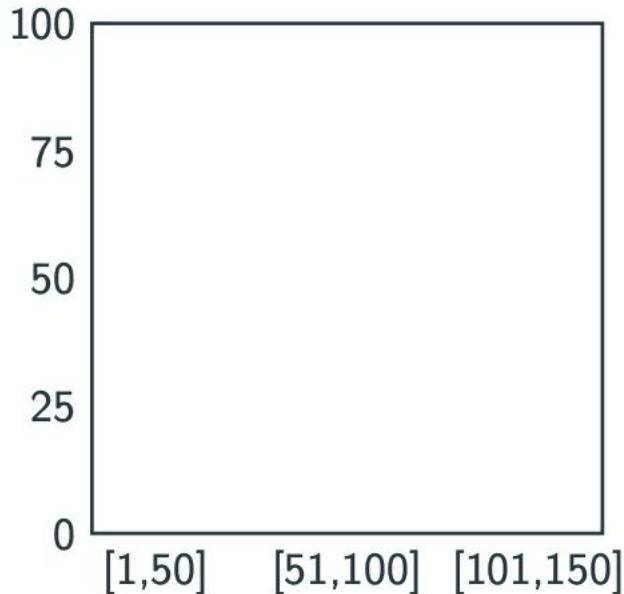
Training Set
Acc

Out-of-Domain Test
Acc

| | % | u | r | s | z | \$ | u | ... |
|----------|---|---|---|---|---|----|---|-----|
| prev = u | 0 | 0 | 1 | 0 | 0 | 0 | 0 | |
| prev = r | 0 | 0 | 0 | 1 | 0 | 0 | 0 | |
| prev = s | 0 | 0 | 0 | 0 | 1 | 0 | 0 | |
| prev = z | 0 | 0 | 0 | 0 | 0 | 1 | 0 | |

collect
immediately
preceding
token

Unique Copy



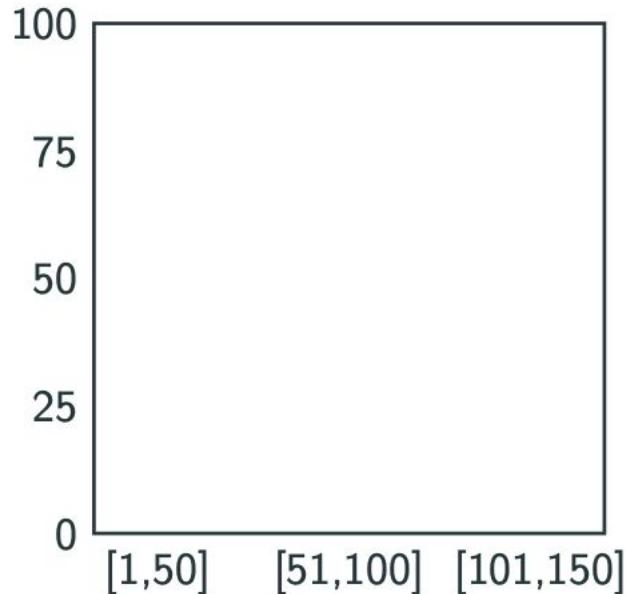
| | % | u | r | s | z | \$ | u | ... |
|----------|---|---|---|---|---|----|---|-----|
| prev = u | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| prev = r | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| prev = s | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| prev = z | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| ur | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

collect
bigram
statistics

Training Set
Acc

Out-of-Domain Test
Acc

Unique Copy



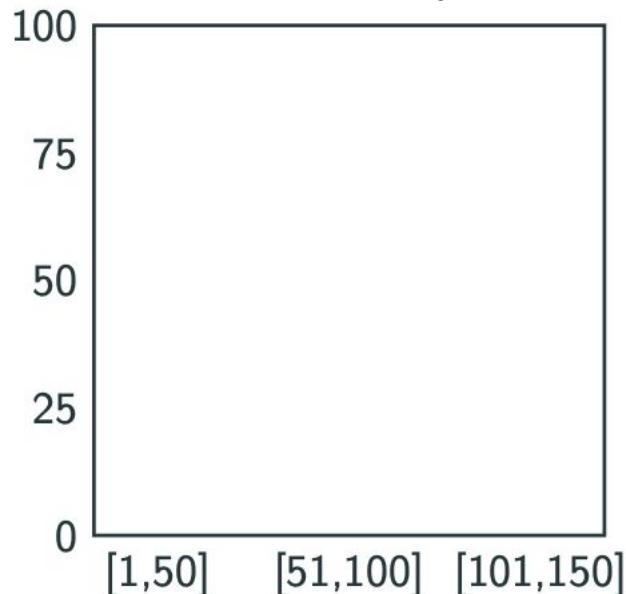
| | % | u | r | s | z | \$ | u | ... |
|----------|---|---|---|---|---|----|---|-----|
| prev = u | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| prev = r | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| prev = s | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| prev = z | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| ur | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| rs | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

collect
bigram
statistics

Training Set
Acc

Out-of-Domain Test
Acc

Unique Copy



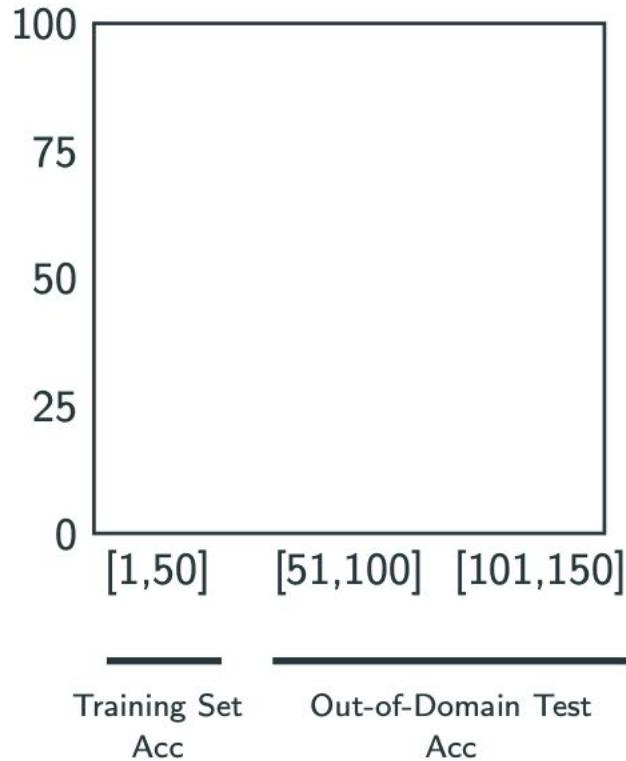
| | % | u | r | s | z | \$ | u | ... |
|----------|---|---|---|---|---|----|---|-----|
| prev = u | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| prev = r | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| prev = s | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| prev = z | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| ur | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| rs | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| sz | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

collect
bigram
statistics

Training Set
Acc

Out-of-Domain Test
Acc

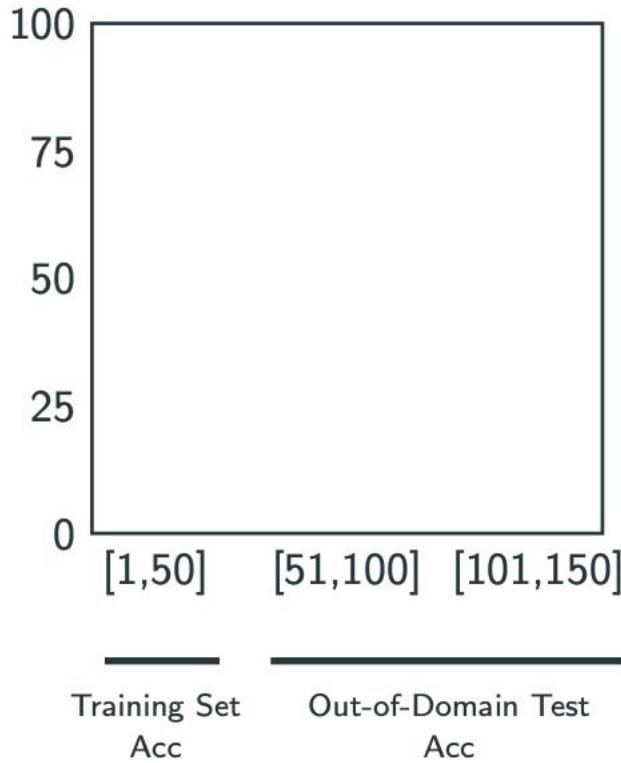
Unique Copy



| | % | u | r | s | z | \$ | u | ... |
|----------|---|---|---|---|---|----|---|-----|
| prev = u | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| prev = r | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| prev = s | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| prev = z | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| ur | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| rs | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| sz | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| us | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

collect
bigram
statistics

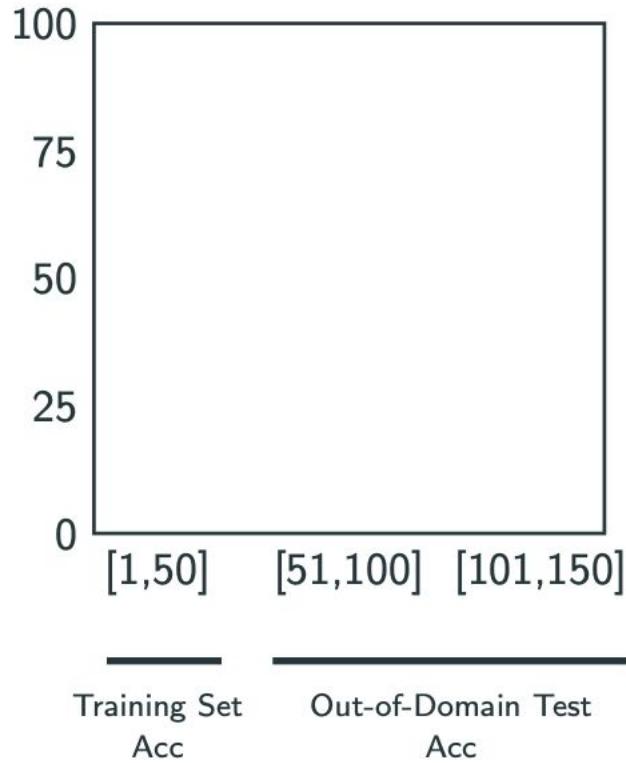
Unique Copy



| | % | u | r | s | z | \$ | u | ... |
|----------|-----|-----|-----|-----|-----|-----|-----|-----|
| prev = u | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| prev = r | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| prev = s | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| prev = z | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| ur | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| rs | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| sz | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| us | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |

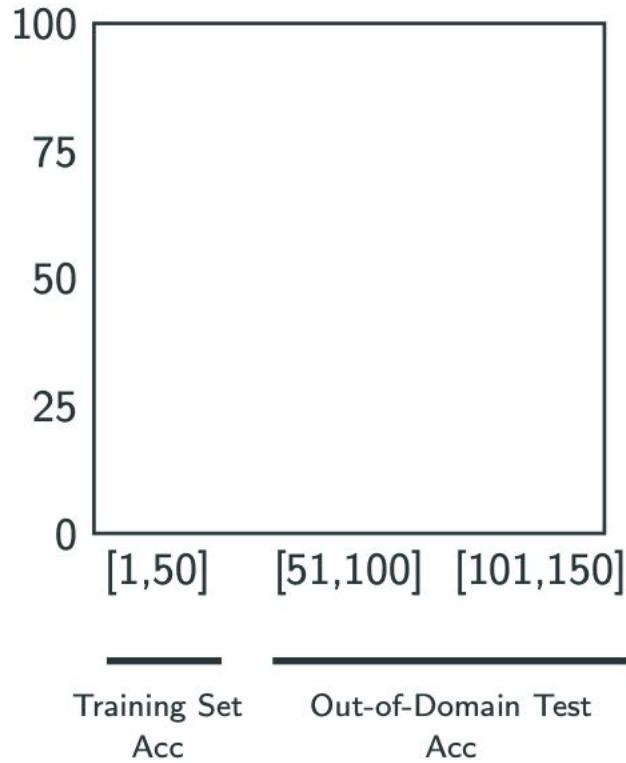
collect
bigram
statistics

Unique Copy



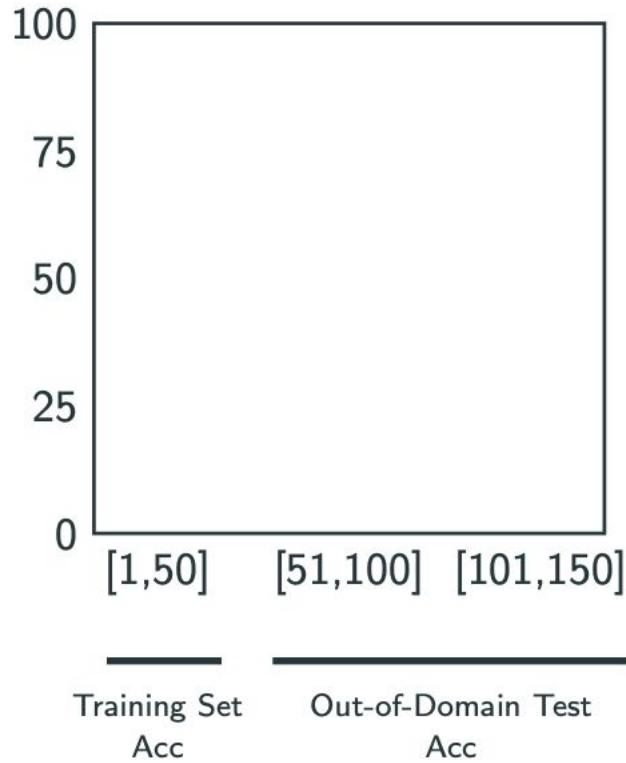
| | % | u | r | s | z | \$ | u | ... |
|-----------|-----|-----|-----|-----|-----|-----|----------|-----|
| prev = u | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| prev = r | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| prev = s | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| prev = z | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| ur | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| rs | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| sz | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| us | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |

Unique Copy



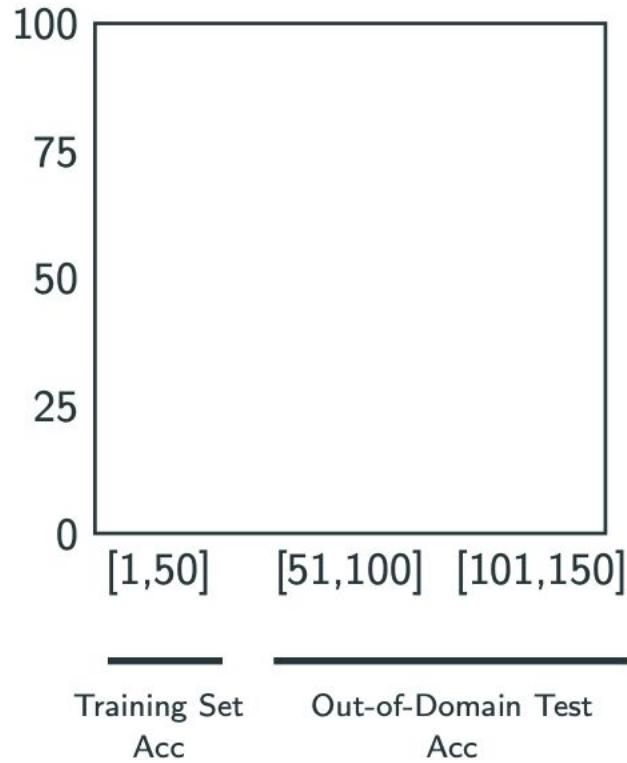
| | % | u | r | s | z | \$ | u | ... |
|-----------|-----|---|---|---|---|----|----------|-----|
| prev = u | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| prev = r | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| prev = s | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| prev = z | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| ur | 0 | 0 | 1 | 1 | 1 | 1 | 1 | |
| rs | 0 | 0 | 0 | 1 | 1 | 1 | 1 | |
| sz | 0 | 0 | 0 | 0 | 1 | 1 | 1 | |
| us | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ... | ... | | | | | | ... | |

Unique Copy



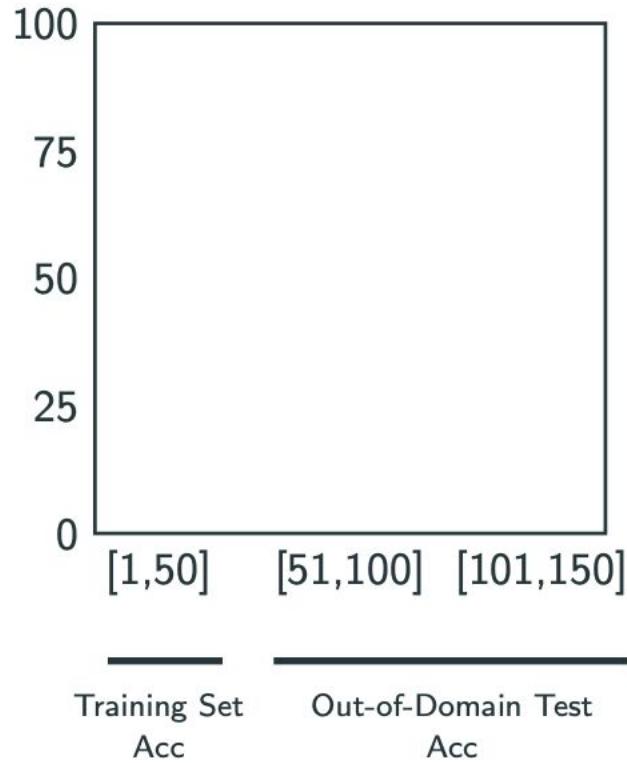
| | % | u | r | s | z | \$ | u | r |
|-----------|-----|---|---|---|---|----|----------|----------|
| prev = u | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| prev = r | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| prev = s | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| prev = z | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| ur | 0 | 0 | 1 | 1 | 1 | 1 | 1 | |
| rs | 0 | 0 | 0 | 1 | 1 | 1 | 1 | |
| sz | 0 | 0 | 0 | 0 | 1 | 1 | 1 | |
| us | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ... | ... | | | | | | | |

Unique Copy



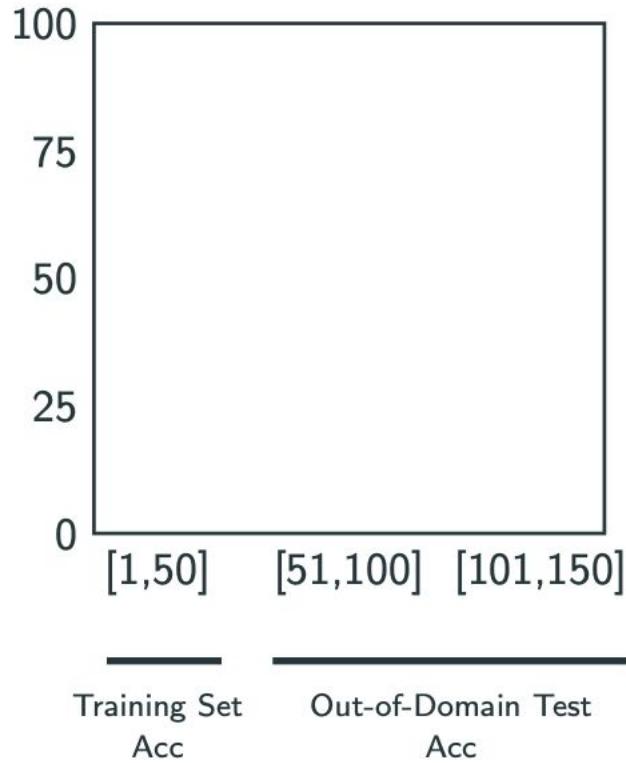
| | % | u | r | s | z | \$ | u | r |
|----------|-----|---|---|---|---|----|---|---|
| prev = u | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| prev = r | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| prev = s | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| prev = z | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| ur | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| rs | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| sz | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| us | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ... | ... | | | | | | | |

Unique Copy



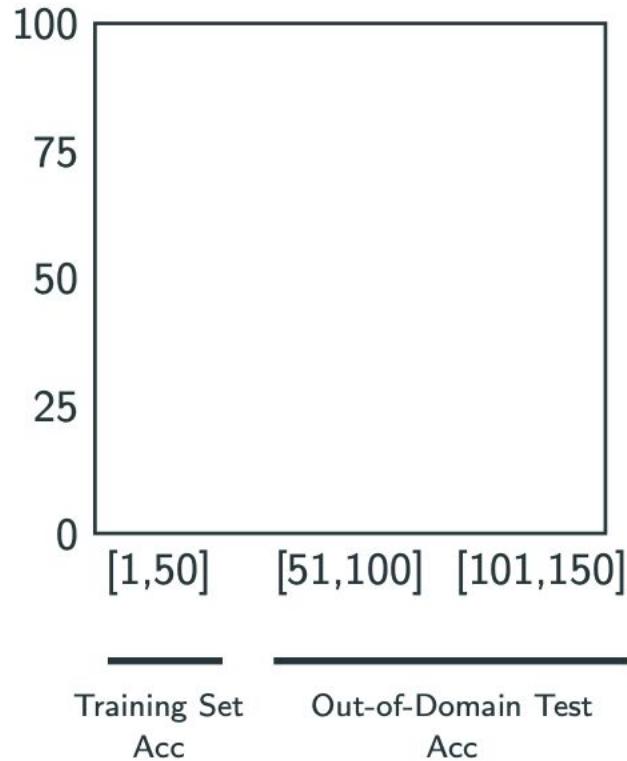
| | % | u | r | s | z | \$ | u | r |
|-----------|-----|-----|-----|-----|-----|-----|-----|----------|
| prev = u | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| prev = r | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| prev = s | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| prev = z | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| ur | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| rs | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| sz | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| us | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |

Unique Copy



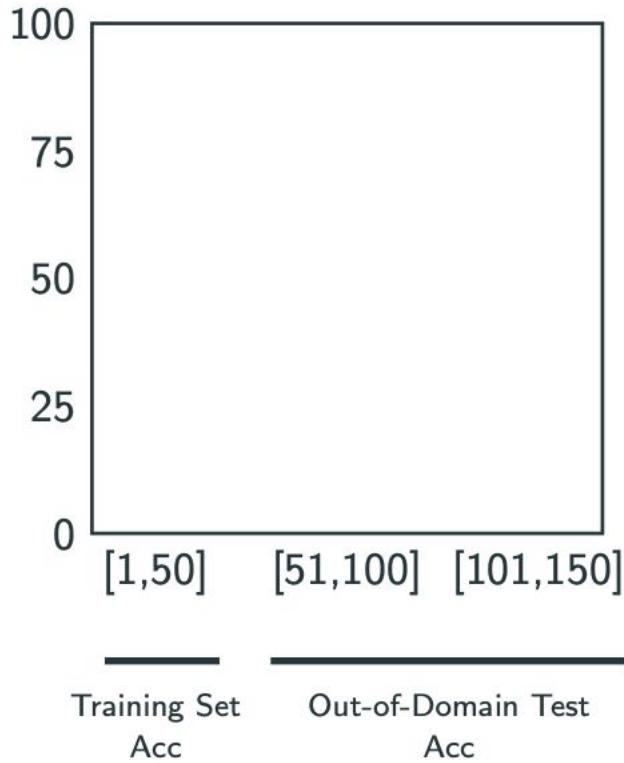
| | % | u | r | s | z | \$ | u | r | s |
|-----------|-----|---|---|---|---|----|---|----------|----------|
| prev = u | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| prev = r | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| prev = s | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| prev = z | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| ur | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| rs | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | |
| sz | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| us | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ... | ... | | | | | | | | |

Unique Copy



| | % | u | r | s | z | \$ | u | r | s | z | \$ |
|----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| prev = u | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| prev = r | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| prev = s | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| prev = z | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| ur | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| rs | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| sz | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| us | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

Unique Copy



° u r s z \$ u r s z @

Unique Copy (Induction Head)

:

$$CP_a(i) := \# [j \leq i, j = i - 1] Q_a(j)$$

$$PRED_a(i) := CP_a(i) \geq 1$$

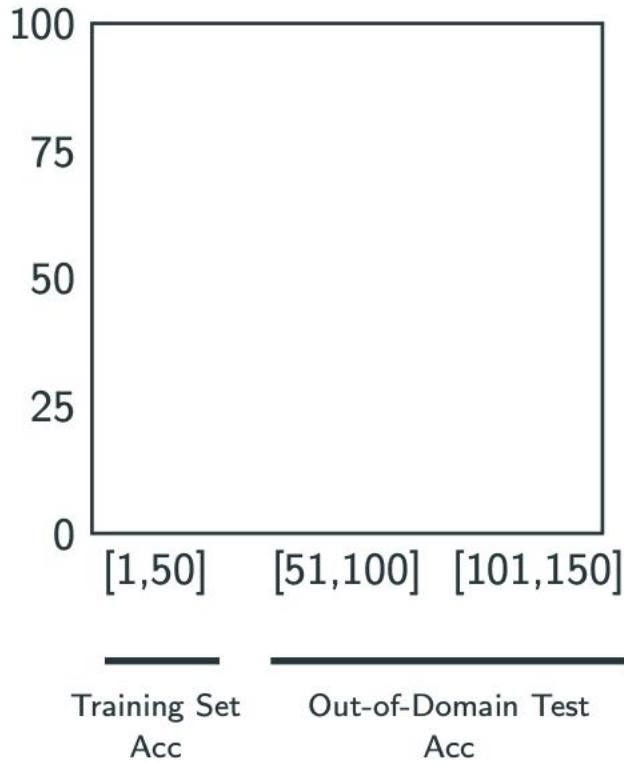
check preceding token

(1)

(2)

(3)

Unique Copy



° u r s z \$ u r s z @

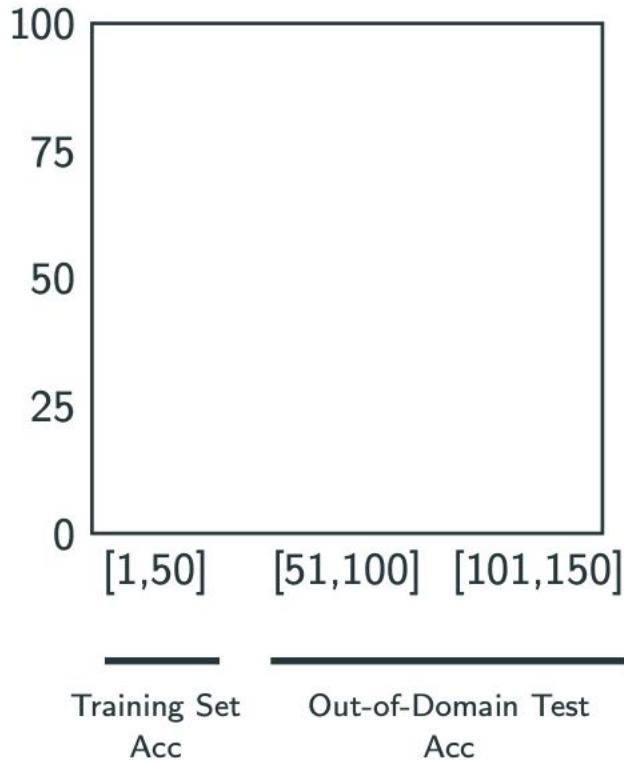
Unique Copy (Induction Head)

⋮ (1)
 $CP_a(i) := \# [j \leq i, j = i - 1] Q_a(j)$ (2)
 $PRED_a(i) := CP_a(i) \geq 1$ (3)

⋮ (4)
 $CBIGRAM_{ab} := \# [j \leq i] Q_b(j) \wedge PRED_a(j)$ (5)
 $EXISTS_{ab} := CBIGRAM_{ab}(i) \geq 1$ (6)

*collect all
bigram
statistics*

Unique Copy



o u r s z \$ u r s z @

Unique Copy (Induction Head)

$$\vdots \quad (1)$$

$$CP_a(i) := \# [j \leq i, j = i - 1] Q_a(j) \quad (2)$$

$$PRED_a(i) := CP_a(i) \geq 1 \quad (3)$$

$$\vdots \quad (4)$$

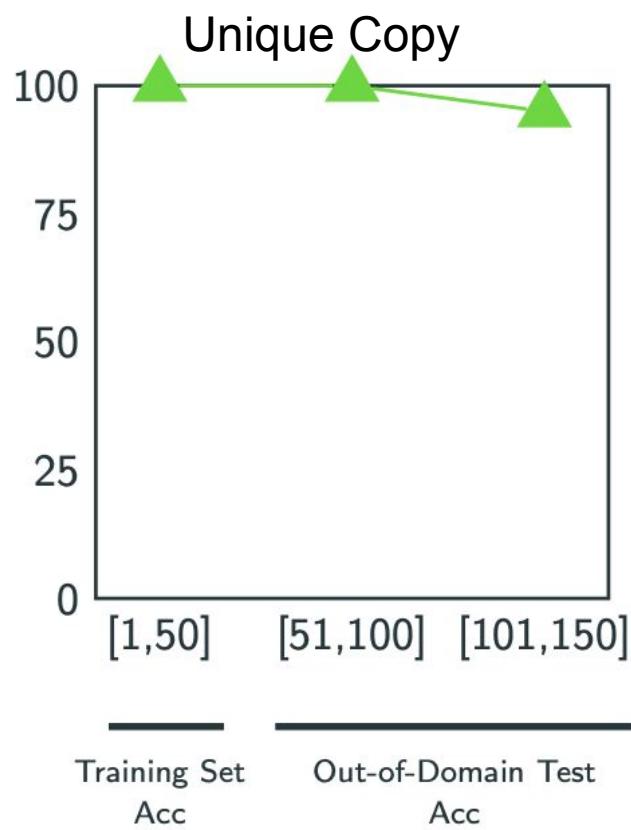
$$CBIGRAM_{ab} := \# [j \leq i] Q_b(j) \wedge PRED_a(j) \quad (5)$$

$$EXISTS_{ab} := CBIGRAM_{ab}(i) \geq 1 \quad (6)$$

$$\vdots \quad (7)$$

$$NEXT_a(i) := \bigvee_{\sigma \in \Sigma} [Q_\sigma(i) \wedge EXISTS_{\sigma a}(i)] \quad (8)$$

*use bigram
statistics to
output next token*



| | % | u | r | s | z | \$ | u | r | s | z | \$ |
|----------|-----|---|---|---|---|----|---|---|---|---|----|
| prev = u | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| prev = r | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| prev = s | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| prev = z | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| ur | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| rs | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| sz | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| us | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ... | ... | | | | | | | | | | |

C-RASP[*periodic, local*]

Boolean-Valued Operations

Count-Valued Operations

C-RASP[*periodic, local*]

Boolean-Valued Operations

Initial

$P(i) := Q_\sigma(i)$
for $\sigma \in \Sigma$

check for symbols

C-RASP[*periodic, local*]

Boolean-Valued Operations

Initial $P(i) := Q_\sigma(i)$
for $\sigma \in \Sigma$

Boolean $P(i) := \neg P_1(i)$
 $P(i) := P_1(i) \wedge P_2(i)$ propositional logic

Constant $P(i) := \top$

C-RASP[*periodic, local*]

Boolean-Valued Operations

| | |
|-------------------|---|
| Initial | $P(i) := Q_\sigma(i)$ <i>for</i> $\sigma \in \Sigma$ |
| Boolean | $P(i) := \neg P_1(i)$ $P(i) := P_1(i) \wedge P_2(i)$ |
| Constant | $P(i) := \top$ |
| Positional | $P(i) := \phi(i)$ <i>for</i> $\phi \in \Phi$ |

checking position

C-RASP[*periodic, local*]

Boolean-Valued Operations

| | |
|-------------------|---|
| Initial | $P(i) := Q_\sigma(i)$ <i>for</i> $\sigma \in \Sigma$ |
| Boolean | $P(i) := \neg P_1(i)$ $P(i) := P_1(i) \wedge P_2(i)$ |
| Constant | $P(i) := \top$ |
| Positional | $P(i) := \phi(i)$ <i>for</i> $\phi \in \Phi$ |
| Comparison | $P(i) := C_1(i) \leq C_2(i)$ |

compare
counts

C-RASP[*periodic, local*]

count how many preceding
positions j satisfy a property

Count-Valued Operations

Counting

$$C(i) := \# [j \leq i, \psi(i, j)] \quad P(j)$$

for $\psi \in \Psi \cup \{\top\}$

C-RASP[*periodic, local*]

count arithmetic

Count-Valued Operations

Counting $C(i) := \# [j \leq i, \psi(i, j)] \ P(j)$
for $\psi \in \Psi \cup \{\top\}$

Conditional $C(i) := P(i) ? C_1(i) : C_2(i)$

Addition $C(i) := C_1(i) + C_2(i)$

Subtraction $C(i) := C_1(i) - C_2(i)$

Constant $C(i) := 1$

C-RASP[*periodic, local*]

| Boolean-Valued Operations | |
|----------------------------------|---|
| Initial | $P(i) := Q_\sigma(i)$ for $\sigma \in \Sigma$ |
| Boolean | $P(i) := \neg P_1(i)$ $P(i) := P_1(i) \wedge P_2(i)$ |
| Constant | $P(i) := \top$ |
| Positional | $P(i) := \phi(i)$ for $\phi \in \Phi$ |
| Comparison | $P(i) := C_1(i) \leq C_2(i)$ |

| Count-Valued Operations | |
|--------------------------------|---|
| Counting | $C(i) := \# [j \leq i, \psi(i, j)] P(j)$ for $\psi \in \Psi \cup \{\top\}$ |
| Conditional | $C(i) := P(i) ? C_1(i) : C_2(i)$ |
| Addition | $C(i) := C_1(i) + C_2(i)$ |
| Subtraction | $C(i) := C_1(i) - C_2(i)$ |
| Constant | $C(i) := 1$ |

C-RASP[*periodic, local*]

Boolean-Valued Operations

Initial $P(i) := Q_\sigma(i)$
for $\sigma \in \Sigma$

Boolean $P(i) := \neg P_1(i)$
 $P(i) := P_1(i) \wedge P_2(i)$

Constant $P(i) := \top$

Positional $P(i) := \phi(i)$
for $\phi \in \Phi$

Comparison $P(i) := C_1(i) \leq C_2(i)$

Count-Valued Operations

Counting $C(i) := \# [j \leq i, \psi(i, j)] P(j)$
for $\psi \in \Psi \cup \{\top\}$

Conditional $C(i) := P(i) ? C_1(i) : C_2(i)$

Addition $C(i) := C_1(i) + C_2(i)$

Subtraction $C(i) := C_1(i) - C_2(i)$

Constant $C(i) := 1$

Operations from original C-RASP
definitions by Yang&Chiang 2024 (COLM).

Newly added (or modified) positionally-aware operations. Needed to model positional encodings.

Boolean-Valued Operations

| | |
|-------------------|---|
| Initial | $P(i) := Q_\sigma(i)$ for $\sigma \in \Sigma$ |
| Boolean | $P(i) := \neg P_1(i)$ $P(i) := P_1(i) \wedge P_2(i)$ |
| Constant | $P(i) := \top$ |
| Positional | $P(i) := \phi(i)$ for $\phi \in \Phi$ |
| Comparison | $P(i) := C_1(i) \leq C_2(i)$ |

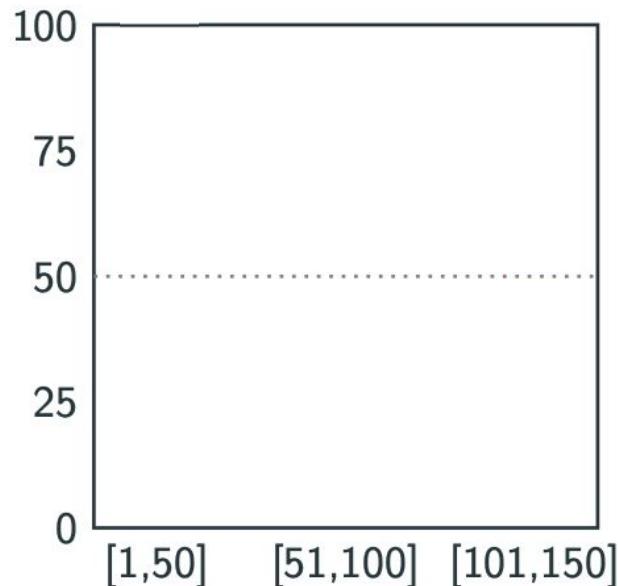
Count-Valued Operations

| | |
|--------------------|---|
| Counting | $C(i) := \# [j \leq i, \psi(i, j)] P(j)$ for $\psi \in \Psi \cup \{\top\}$ |
| Conditional | $C(i) := P(i) ? C_1(i) : C_2(i)$ |
| Addition | $C(i) := C_1(i) + C_2(i)$ |
| Subtraction | $C(i) := C_1(i) - C_2(i)$ |
| Constant | $C(i) := 1$ |

Operations from original C-RASP definitions by Yang&Chiang 2024 (COLM).

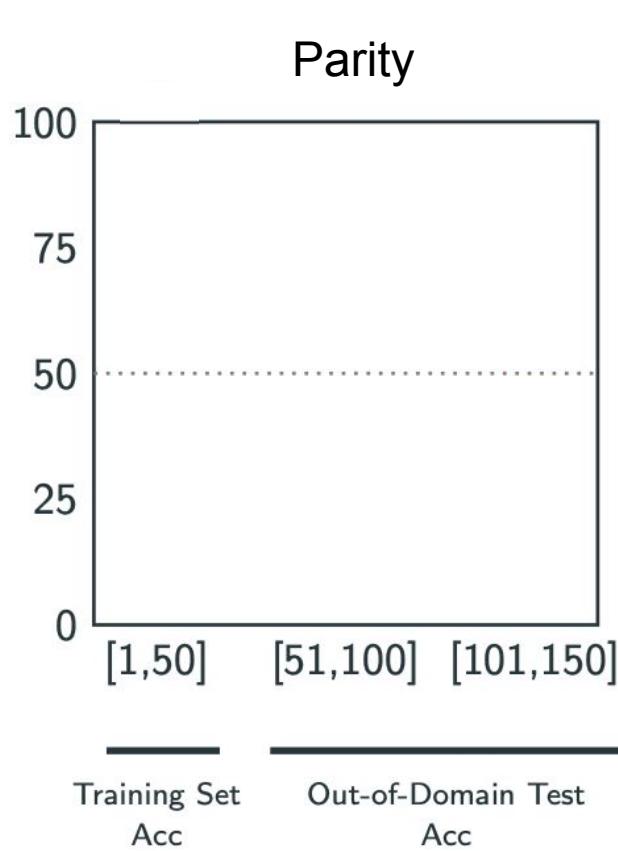
Parity

‰ 0 1 1 0 \$ even @



Training Set
Acc

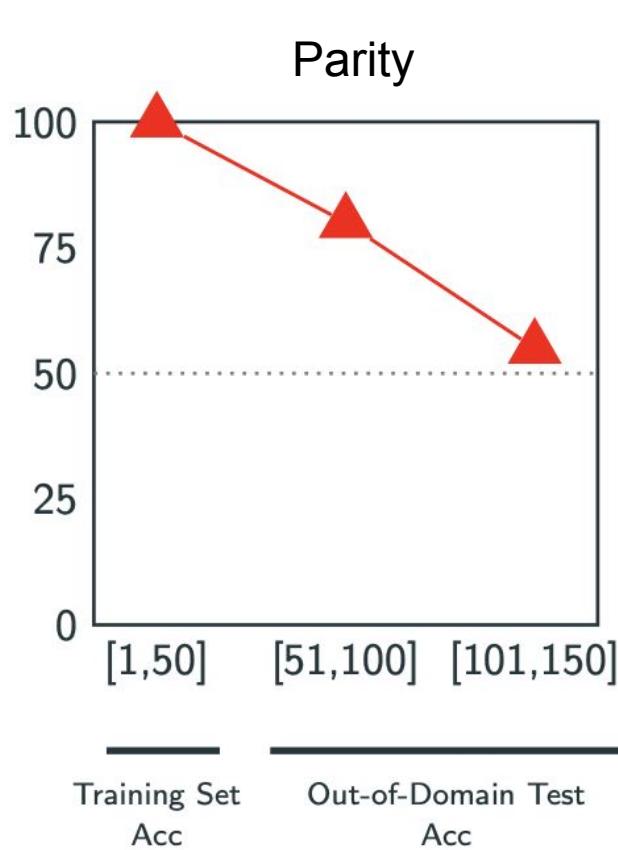
Out-of-Domain Test
Acc



Provably no C-RASP program!

C-RASP[periodic, local] for Parity

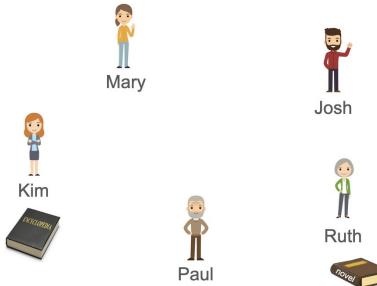
⇒ C-RASP[\emptyset] for $(aa)^*$



Provably no C-RASP program!

C-RASP[periodic, local] for Parity
 \implies C-RASP[\emptyset] for $(aa)^*$

Same holds for more complex state tracking.



Paul had an encyclopedia.
Mary had a novel.
Paul gave his book to Josh.
Mary gave her book to Paul.
Josh gave his book to Kim.
Paul gave his book to Ruth.
Who has the encyclopedia?

Answer: Kim has the encyclopedia.

People exchanging books

:

:

???

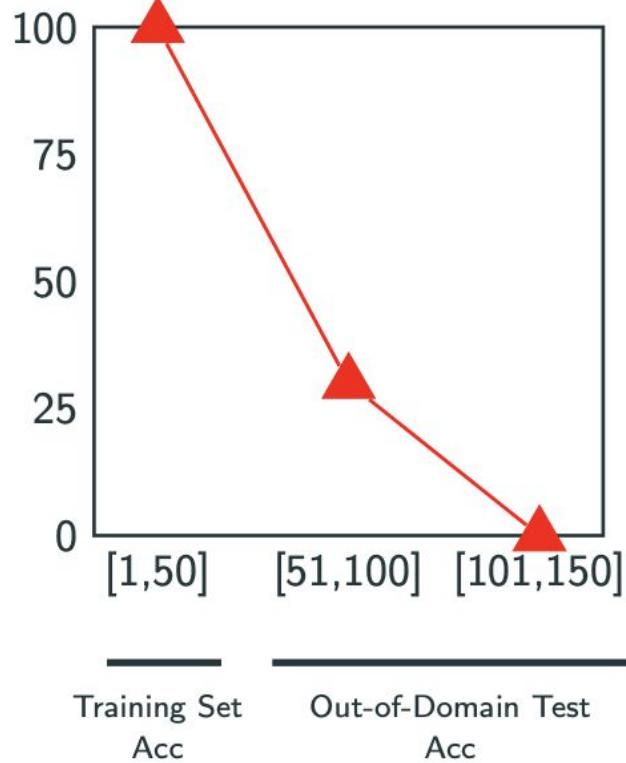
(1)

(2)

(3)

Provably no C-RASP program!

Copy with Repetitions



% a a b a \$ a a b a @

Copy With Repetition

:

:

???

(1)

(2)

(3)

Provably no C-RASP program!

rigorous proof via
communication complexity

Theorem (informal)

Theorem (informal)

Assume f is expressible in C-RASP.

Theorem (informal)

Assume f is expressible in C-RASP.

Choose a transformer T_n

Theorem (informal)

Assume f is expressible in C-RASP.

Choose a transformer T_n

- with context window $2n$

context window of T_n

length $2n$

Theorem (informal)

Assume f is expressible in C-RASP.

Choose a transformer T_n

- with context window $2n$
- that exactly fits f on all inputs up to length n

context window of T_n

length $2n$

length n

*training
data*

Theorem (informal)

Assume f is expressible in C-RASP.

Choose a transformer T_n

- with context window $2n$
- that exactly fits f on all inputs up to length n

subject to a certain regularizer.

context window of T_n

length $2n$

length n

*training
data*

Theorem (informal)

Assume f is expressible in C-RASP.

Choose a transformer T_n

- with context window $2n$
- that exactly fits f on all inputs up to length n

subject to a certain regularizer.

Then T_n matches f at all inputs up to length $2n$,

context window of T_n

length $2n$

length n

*training
data*

length $2n$

*test
examples*

Theorem (informal)

Assume f is expressible in C-RASP.

Choose a transformer T_n

- with context window $2n$
- that exactly fits f on all inputs up to length n

subject to a certain regularizer.

Then T_n matches f at all inputs up to length $2n$.

as long as $n > N_0(f)$.

context window of T_n

length $2n$

length n

*training
data*

length $2n$

*test
examples*



Disclaimer

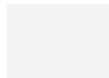
- Idealized learning: all data is available up to length n .
Not SGD



Disclaimer

- Idealized learning: all data is available up to length n .
Not SGD
- Focus on APE / NoPE

Proof Idea



T_1

Proof Idea

T_1



Proof Idea

T_1

T_2

length ≤ 2 length ≤ 4

training
data



test
examples



Proof Idea

T_1

T_2

T_3

length ≤ 3 length ≤ 6

training
data

test
examples



Proof Idea

T_1

T_2

T_3

T_4

length ≤ 4 length ≤ 8

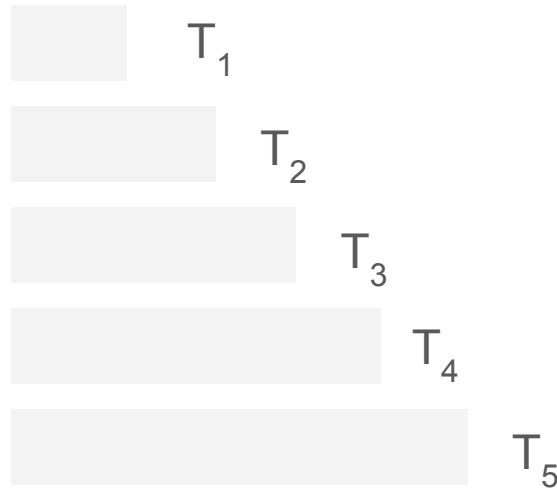
training data



test examples



Proof Idea

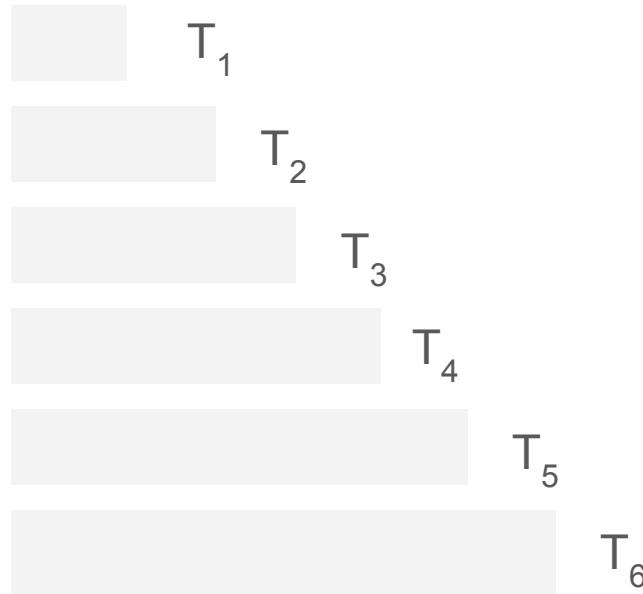


length ≤ 5 length ≤ 10

training data 

test examples 

Proof Idea



length ≤ 6 length ≤ 12

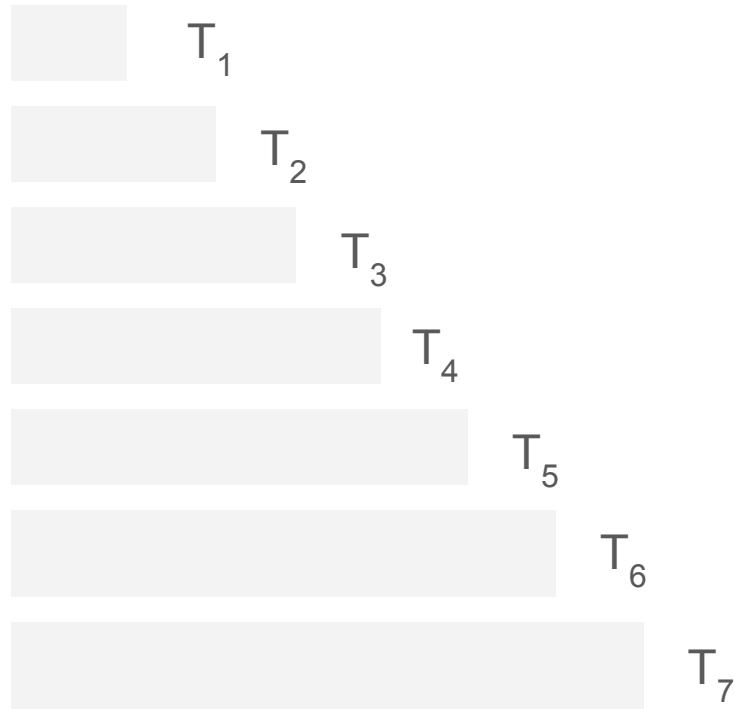
training
data



test
examples



Proof Idea



length ≤ 7 length ≤ 14

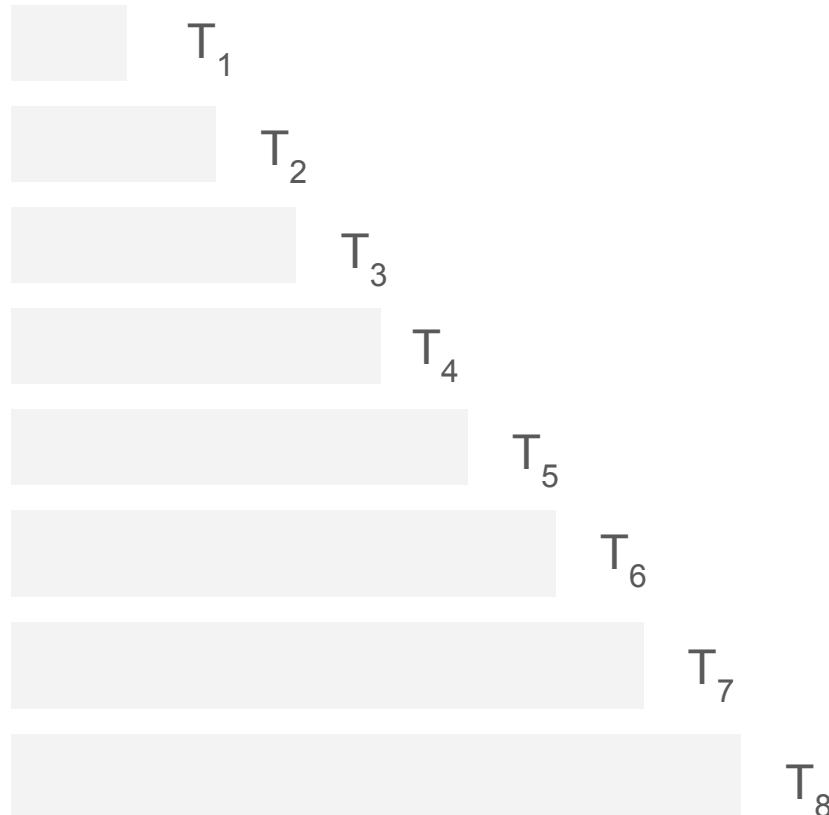
training
data



test
examples



Proof Idea



length ≤ 8 length ≤ 16

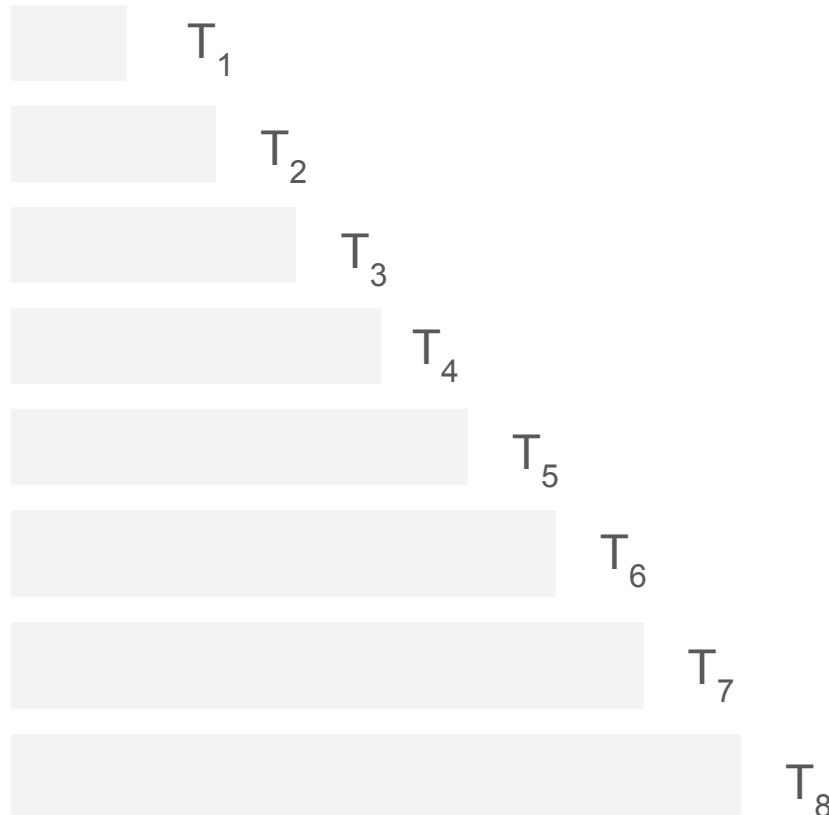
training
data



test
examples



Proof Idea



Length Generalization



training
data

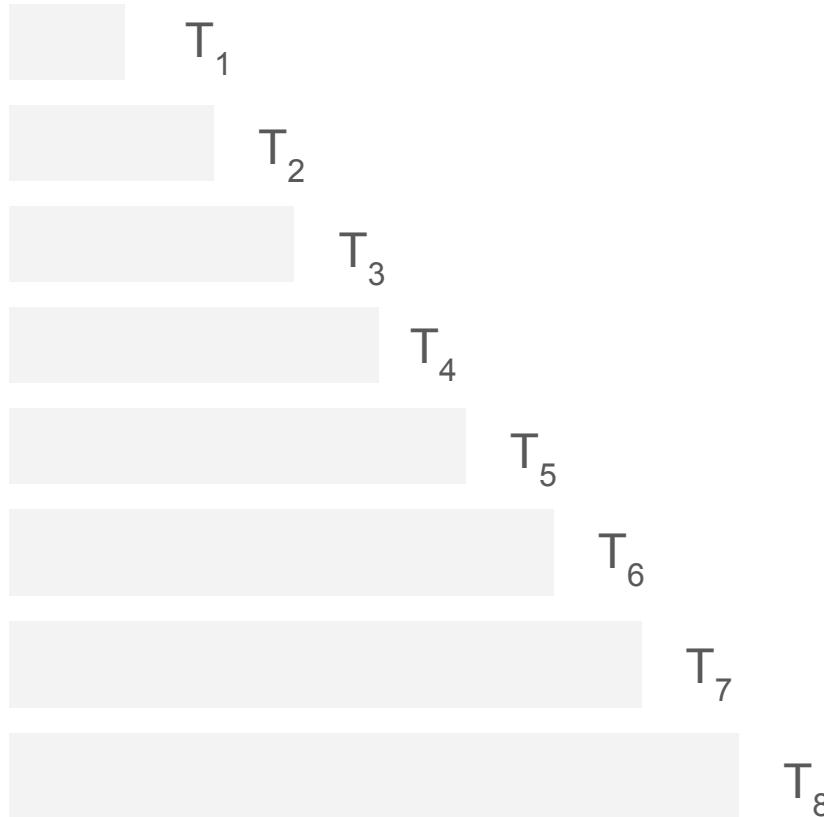


test
examples



Proof Idea

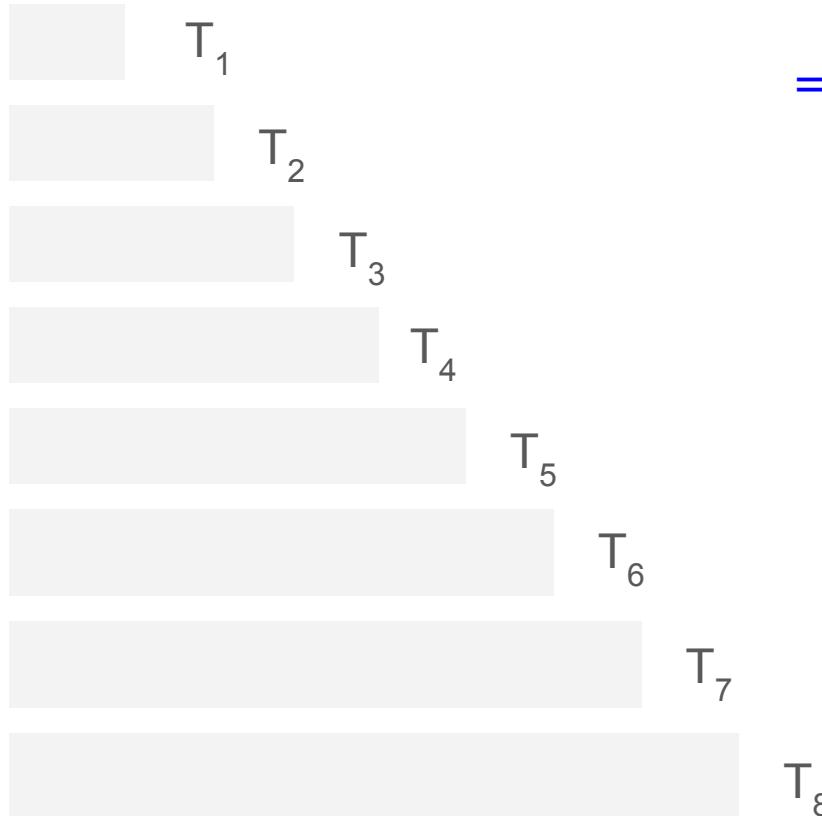
f has C-RASP program



Proof Idea

f has C-RASP program

$\Rightarrow R(T_i)$ is bounded

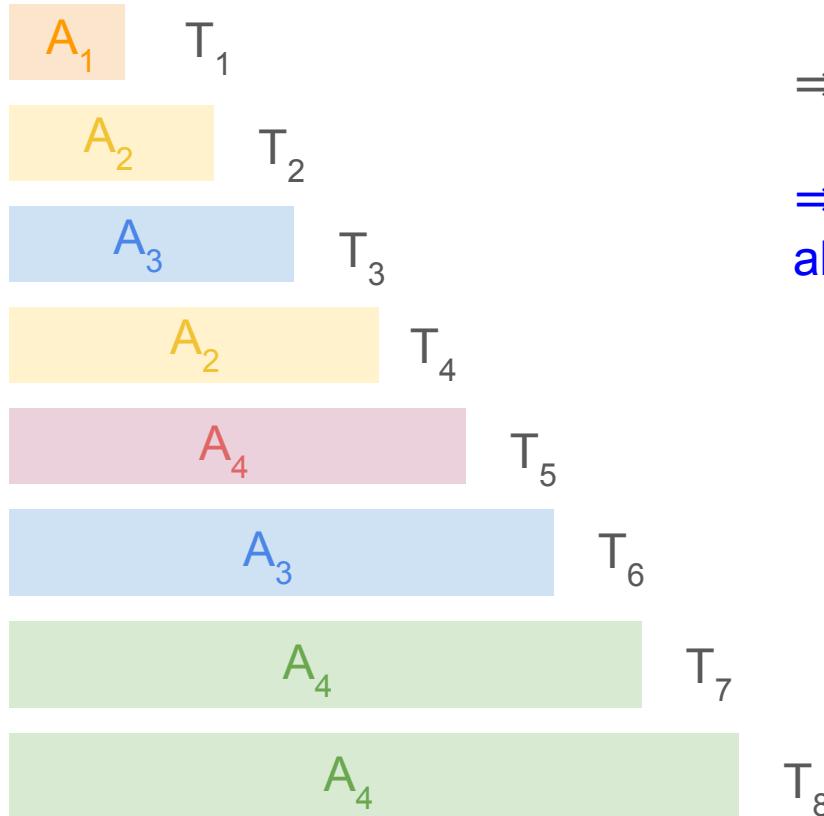


Proof Idea

f has C-RASP program

$\Rightarrow R(T_i)$ is bounded

$\Rightarrow T_i$ only traverse a finite number of distinct algorithms



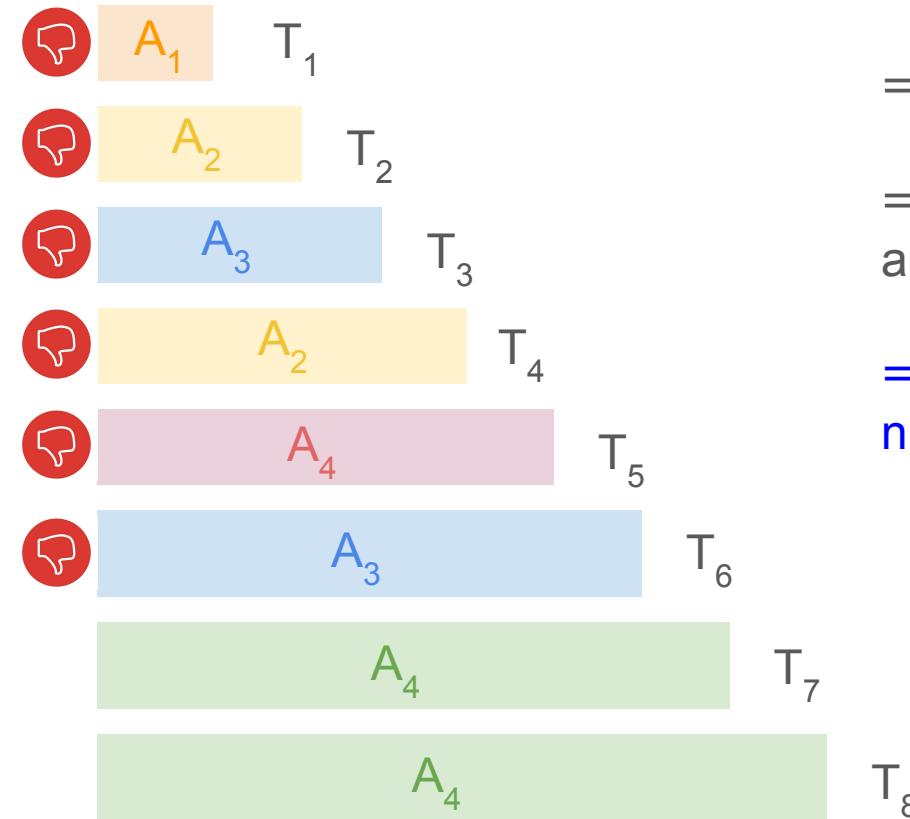
Proof Idea

f has C-RASP program

$\Rightarrow R(T_i)$ is bounded

$\Rightarrow T_i$ only traverse a finite number of distinct algorithms

\Rightarrow each algorithm is either ruled out at some n



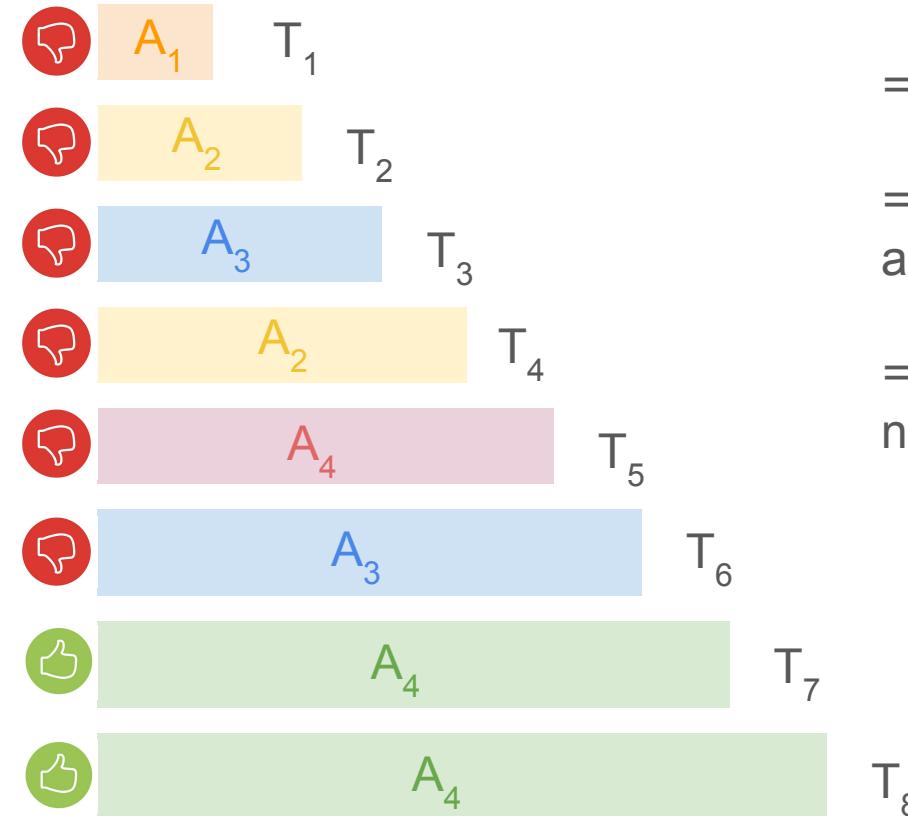
Proof Idea

f has C-RASP program

$\Rightarrow R(T_i)$ is bounded

$\Rightarrow T_i$ only traverse a finite number of distinct algorithms

\Rightarrow each algorithm is either ruled out at some n , or never ruled out



Proof Idea

f has C-RASP program

$\Rightarrow R(T_i)$ is bounded

$\Rightarrow T_i$ only traverse a finite number of distinct algorithms

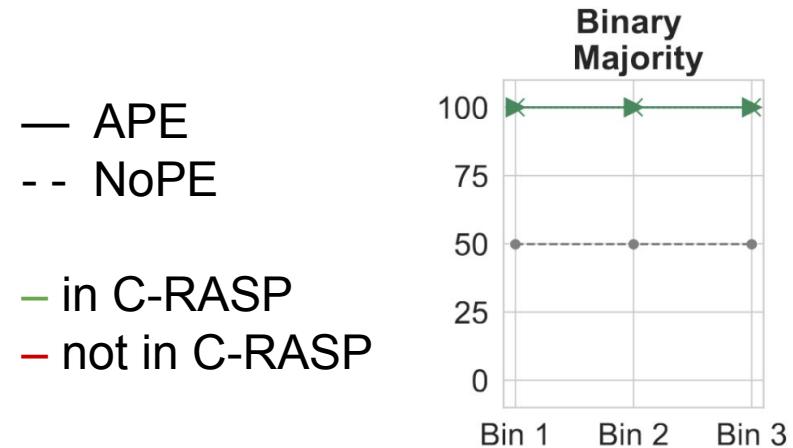
\Rightarrow each algorithm is either ruled out at some n , or never ruled out

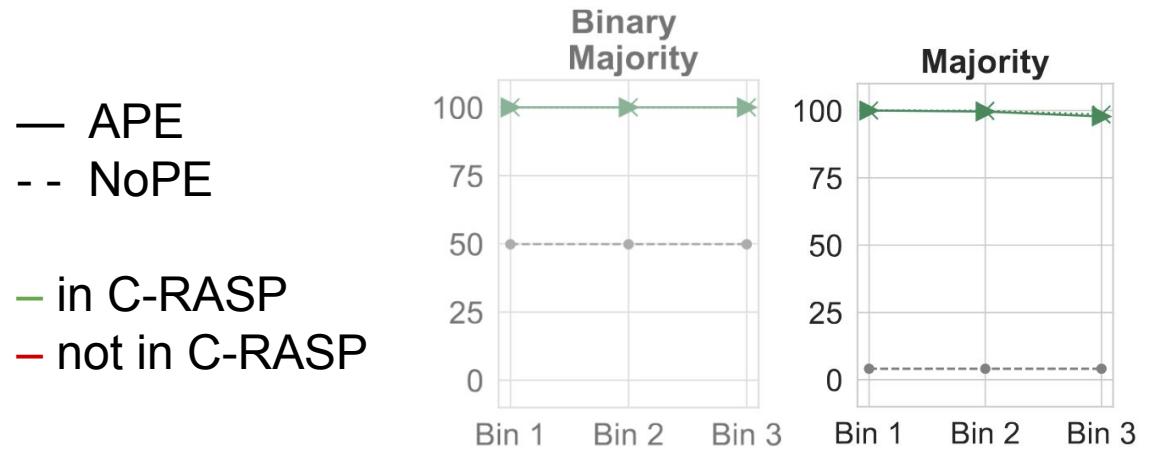
\Rightarrow after some $N_0(f)$, each T_n matches f

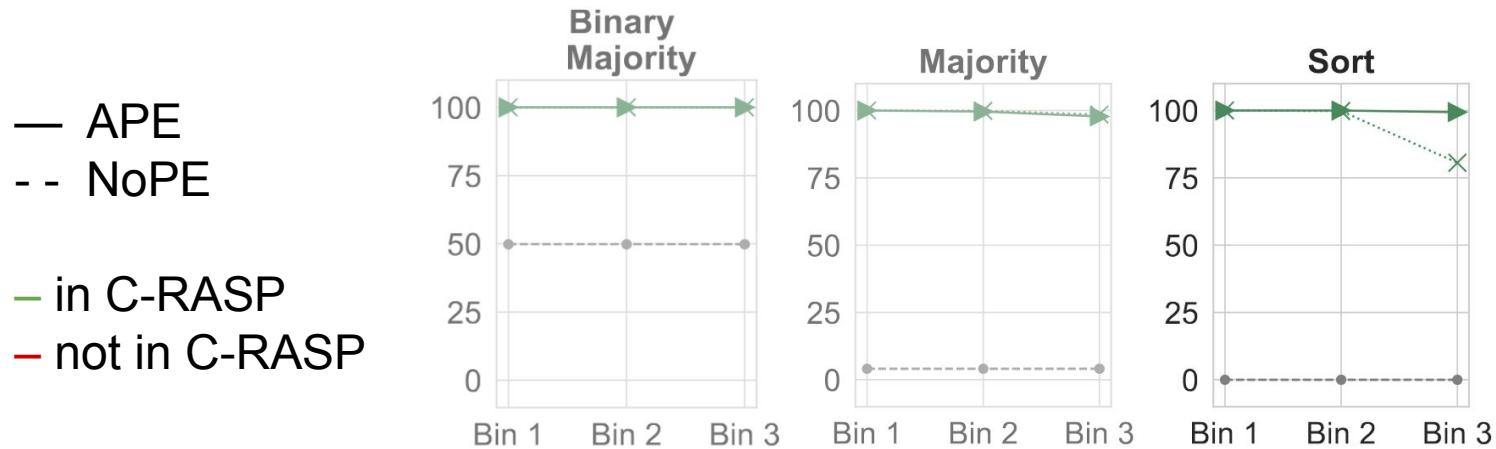
length n length $2n$

training data 

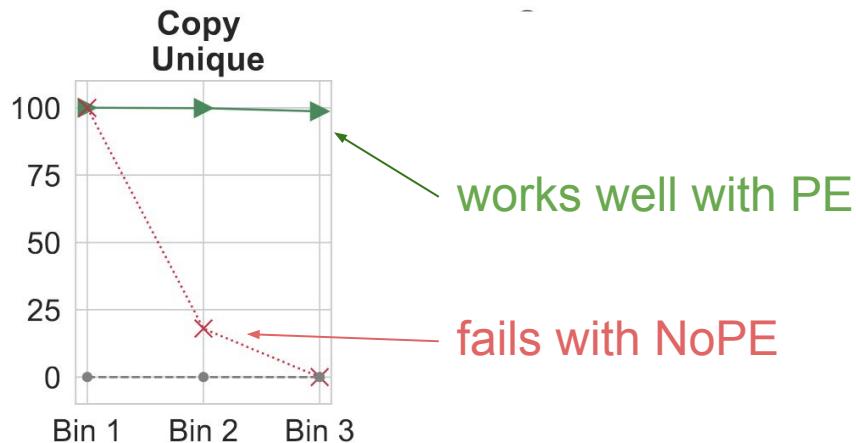
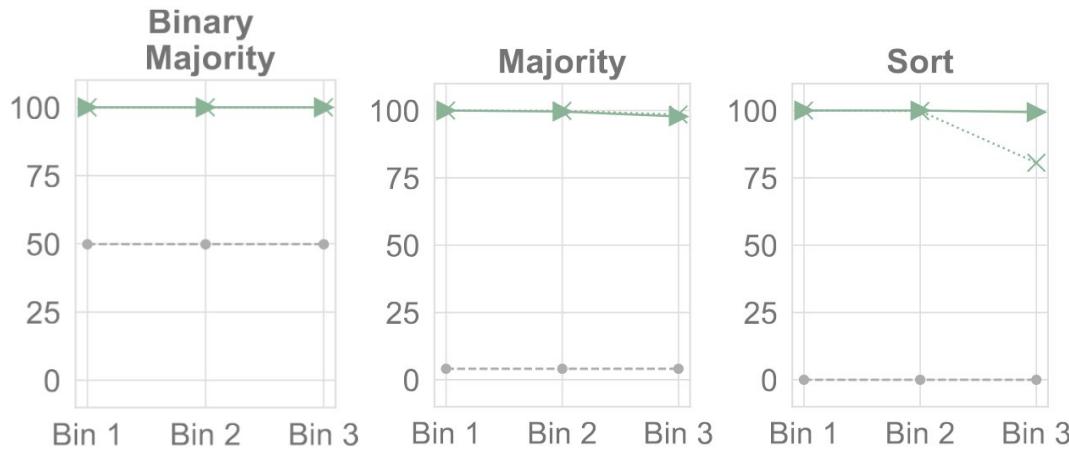
test examples 



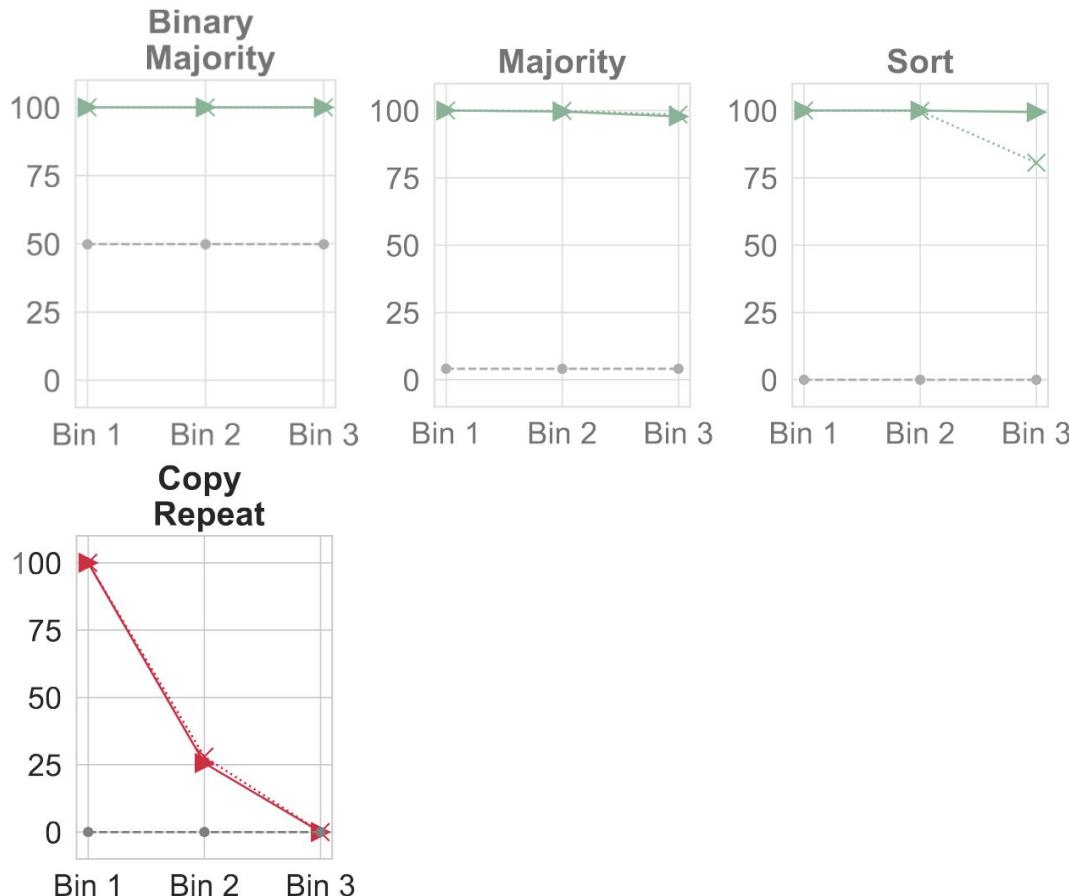




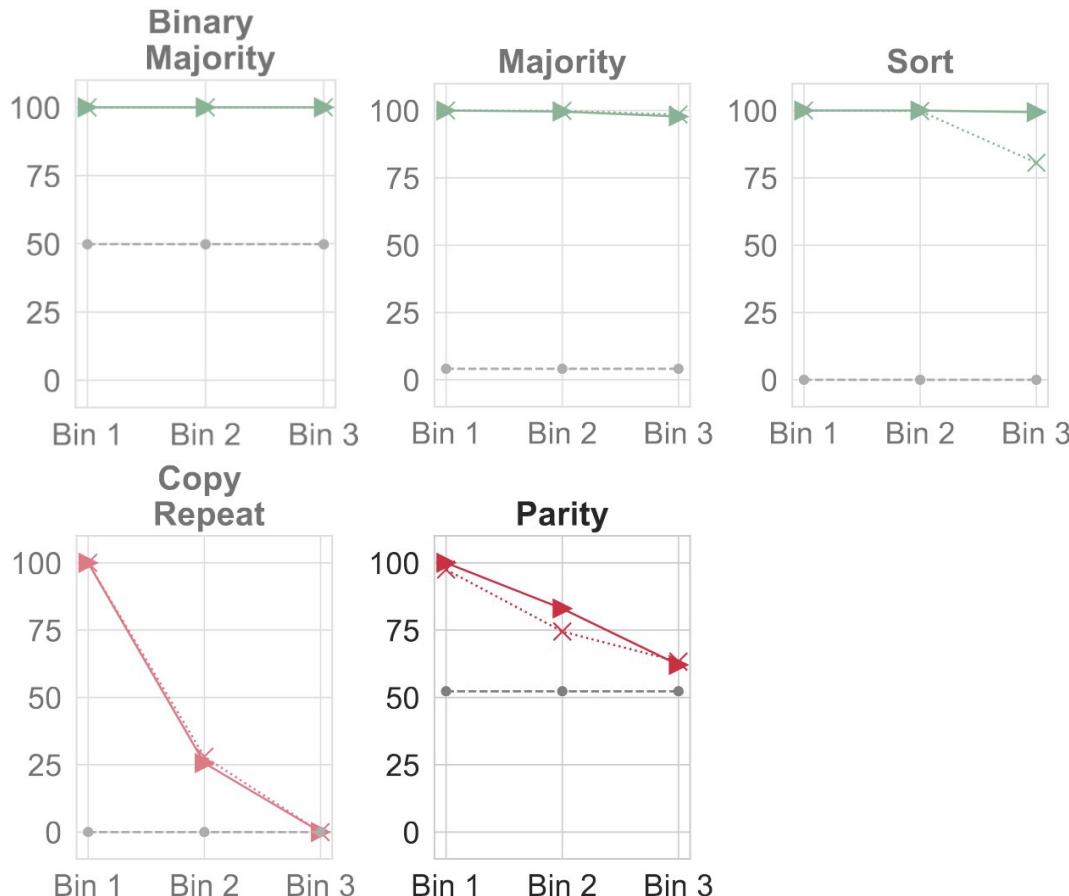
- APE
- NoPE
- in C-RASP
- not in C-RASP

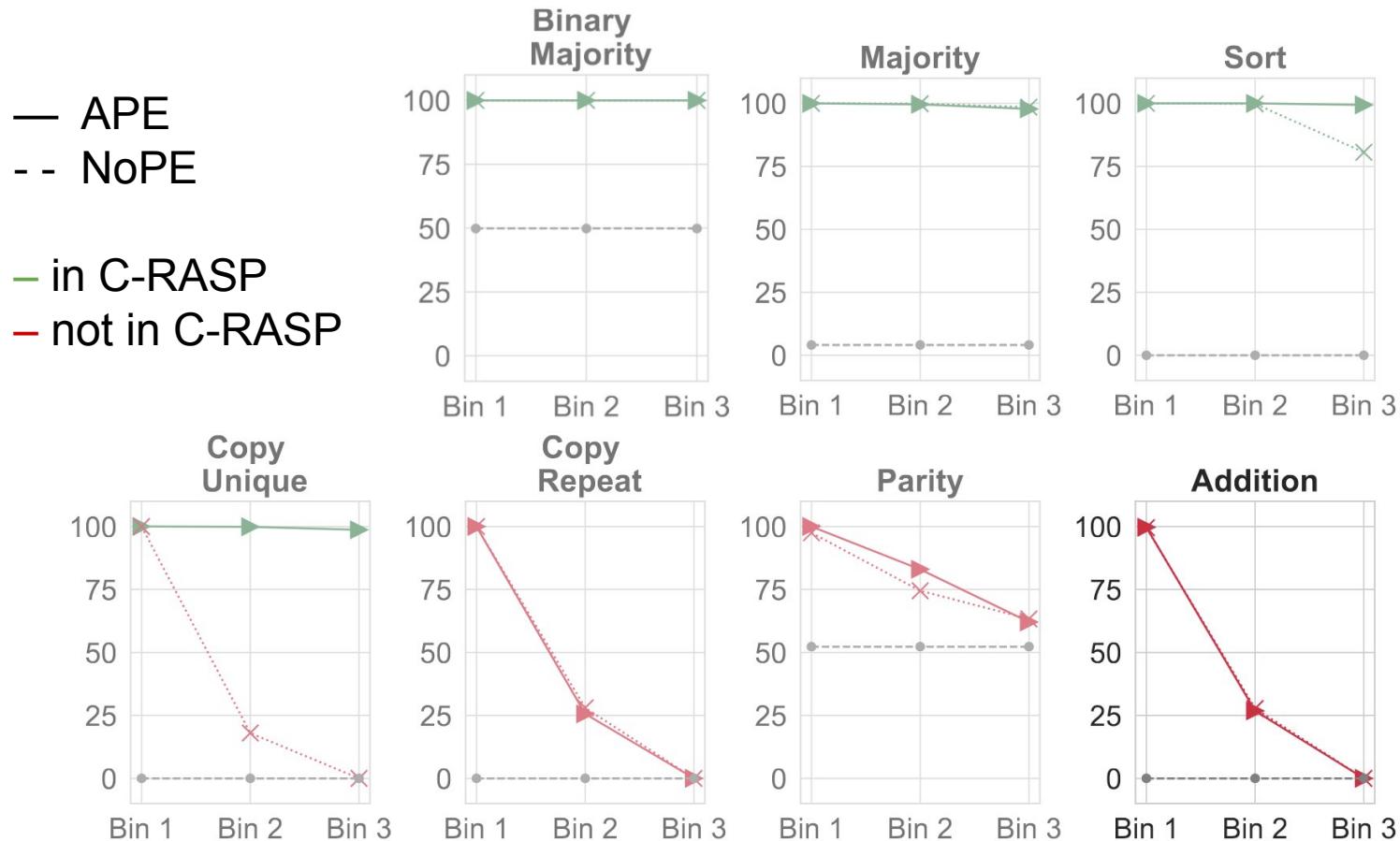


- APE
- NoPE
- in C-RASP
- not in C-RASP

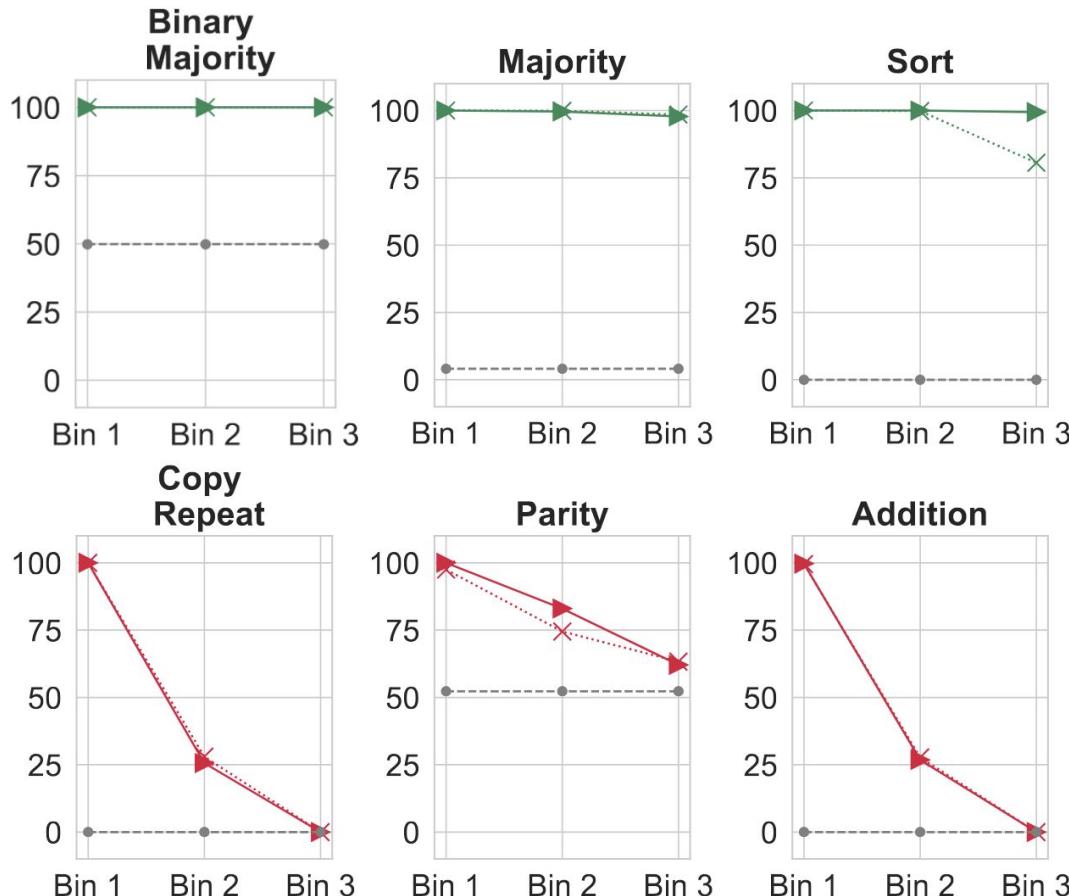


- APE
- NoPE
- in C-RASP
- not in C-RASP



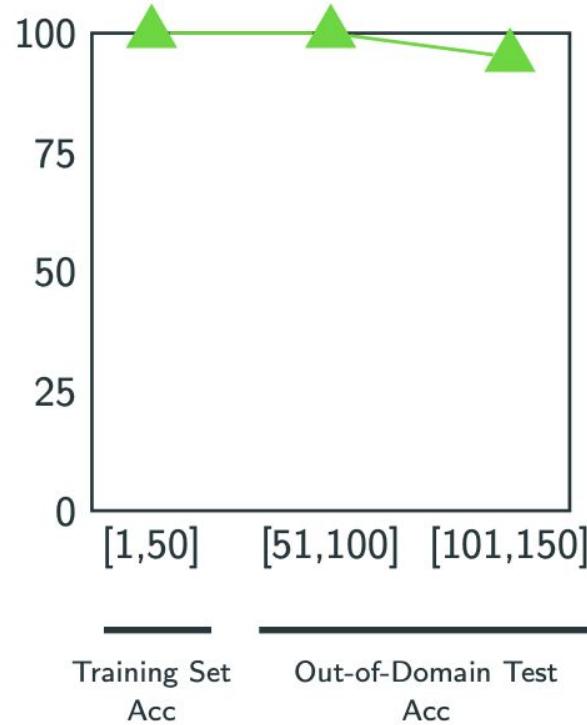


- APE
- NoPE
- in C-RASP
- not in C-RASP

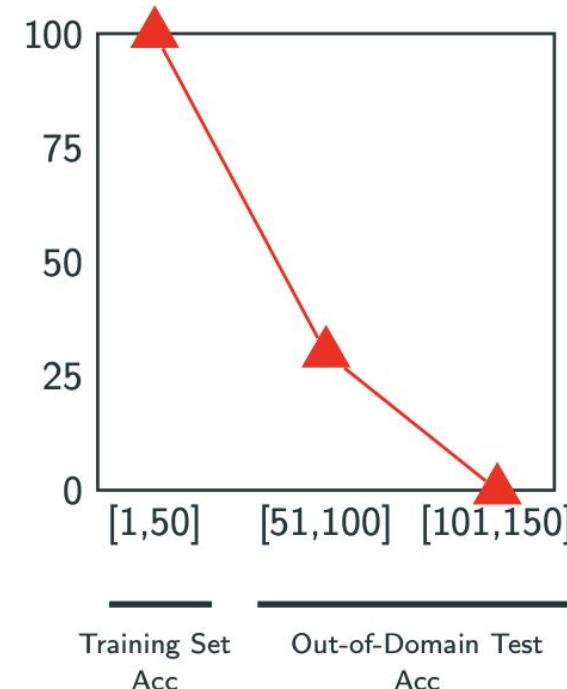


(Non)Unique Copying in LLMs

% u r s z \$ u r s z @



% a b b a \$ a b b a @



(Non)Unique Copying in LLMs

 Lorem-ipsum style copying:

(Non)Unique Copying in LLMs

 Lorem-ipsum style copying:

 Lorem ad ipsum sed do ipsum sed aliqua

(Non)Unique Copying in LLMs

 Lorem-ipsum style copying:

 Lorem ad ipsum sed do ipsum sed aliqua

 Lorem...

(Non)Unique Copying in LLMs

 Lorem-ipsum style copying:

 Lorem ad ipsum sed do ipsum sed aliqua

 Lorem...

(Non)Unique Copying in LLMs

 Lorem-ipsum style copying:

 Lorem ad ipsum sed do ipsum sed aliqua

 Lorem ad...

(Non)Unique Copying in LLMs

 Lorem-ipsum style copying:

 Lorem **ad** ipsum sed do ipsum sed aliqua

 Lorem **ad**...

(Non)Unique Copying in LLMs

 Lorem-ipsum style copying:

 Lorem ad ipsum sed do ipsum sed aliqua

 Lorem ad ipsum ...

(Non)Unique Copying in LLMs

 Lorem-ipsum style copying:

 Lorem ad ipsum sed do ipsum sed aliqua

 Lorem ad ipsum...

(Non)Unique Copying in LLMs

 Lorem-ipsum style copying:

 Lorem ad ipsum sed do ipsum sed aliqua

 Lorem ad ipsum...

(Non)Unique Copying in LLMs

 Lorem-ipsum style copying:

 Lorem ad ipsum sed do ipsum sed aliqua

 Lorem ad ipsum sed...

(Non)Unique Copying in LLMs

 Lorem-ipsum style copying:

 Lorem ad ipsum **sed** do ipsum **sed** aliqua

 Lorem ad ipsum **sed**...

(Non)Unique Copying in LLMs

 Lorem-ipsum style copying:

 Lorem ad ipsum **sed do** ipsum **sed aliqua**

 Lorem ad ipsum **sed...**

(Non)Unique Copying in LLMs

 Lorem-ipsum style copying:

 Lorem ad ipsum **sed do** ipsum **sed aliqua**

 Lorem ad ipsum **sed...**

Prediction: copying errors
should appear when bigram
transitions are ambiguous

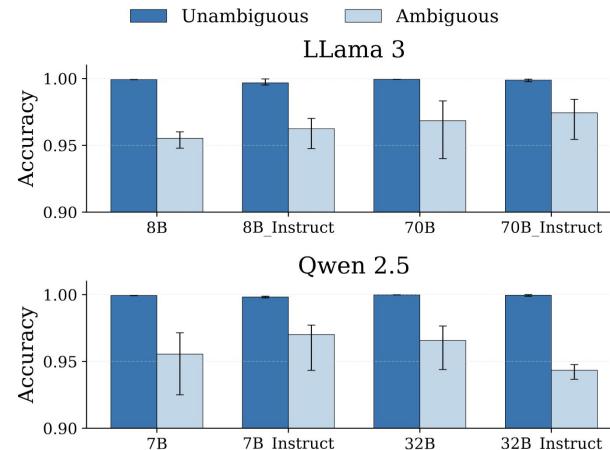
(Non)Unique Copying in LLMs

 Lorem-ipsum style copying:

 Lorem ad ipsum **sed** **do** ipsum **sed** aliqua

 Lorem ad ipsum **sed**...

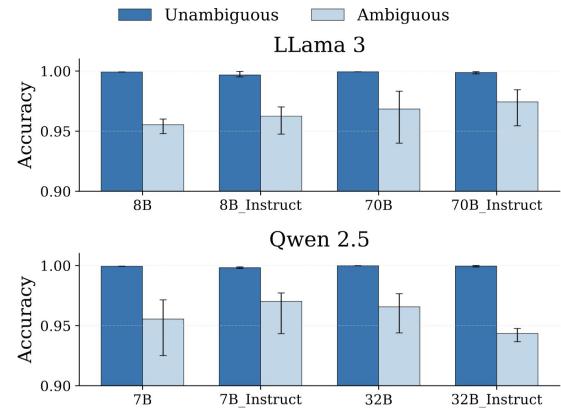
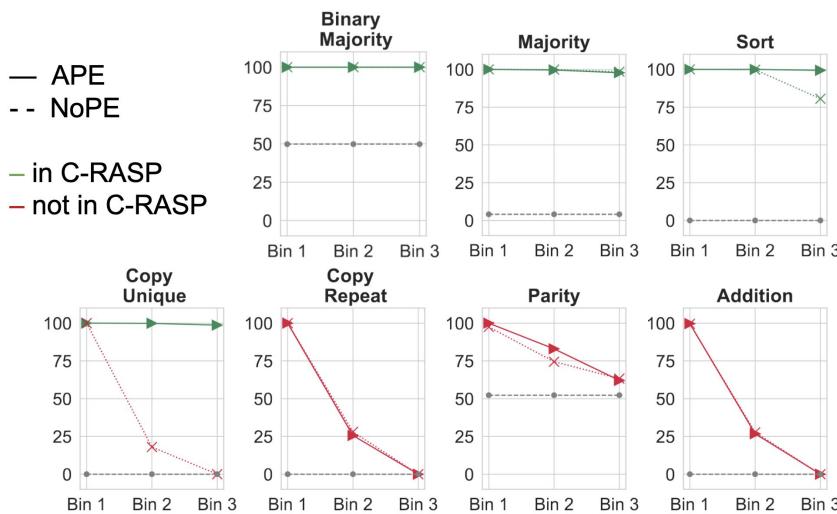
Prediction: copying errors
should appear when bigram
transitions are ambiguous



On which tasks do transformers length-generalize?

UNIQUE COPY

$$\begin{aligned}
 CP_a(i) &:= \# [j \leq i, j = i - 1] Q_a(j) & (1) \\
 CBIGRAM_{ab} &:= \# [j \leq i] Q_b(j) \wedge CP_a(j) \geq 1 & (2) \\
 NEXT_a(i) &:= \bigvee_{\sigma \in \Sigma} [Q_\sigma(i) \wedge CBIGRAM_{\sigma a}(i) \geq 1] & (3)
 \end{aligned}$$



Questions for Future Work

Q1: Quantitative bounds for length generalization.

Questions for Future Work

Q1: Quantitative bounds for length generalization.

How big is $N_0(f)$?

First steps:

Non-Asymptotic Length Generalization

Thomas Chen¹ Tengyu Ma¹ Zhiyuan Li²

Quantitative Bounds for Length Generalization in Transformers

Zachary Izzo*
NEC Labs America

ZACH@NEC-LABS.COM

Eshaan Nichani*
Princeton University

ESHNICH@PRINCETON.EDU

Jason D. Lee
Princeton University

JASONDLEE88@GMAIL.COM

Questions for Future Work

Q2: Beyond individual tasks

Questions for Future Work

Q2: Beyond individual tasks

How can circuit sharing between task help length generalization?

First steps:

Extrapolation by Association: Length Generalization Transfer in Transformers

Ziyang Cai*
University of Wisconsin-Madison

Nayoung Lee
University of Wisconsin-Madison

Avi Schwarzschild
Carnegie Mellon University

Samet Oymak
University of Michigan

Dimitris Papailiopoulos
University of Wisconsin-Madison
Microsoft Research

Length Generalization via Auxiliary Tasks

Pranjal Awasthi
Google
pranjalawasthi@google.com

Anupam Gupta
NYU
anupam.g@nyu.edu

Ravi Kumar
Google
ravi.k53@gmail.com

Questions for Future Work

Q2: Beyond individual tasks

How can circuit sharing between task help length generalization?

Is there data that provably unlocks broad length generalization in reasoning?

First steps:

**Extrapolation by Association: Length Generalization
Transfer in Transformers**

Ziyang Cai*
University of Wisconsin-Madison

Nayoung Lee
University of Wisconsin-Madison

Avi Schwarzschild
Carnegie Mellon University

Samet Oymak
University of Michigan

Dimitris Papailiopoulos
University of Wisconsin-Madison
Microsoft Research

Length Generalization via Auxiliary Tasks

Pranjal Awasthi
Google
pranjalawasthi@google.com

Anupam Gupta
NYU
anupam.g@nyu.edu

Ravi Kumar
Google
ravi.k53@gmail.com

Basic Algorithmic Abilities

1-step

Which abilities can transformers learn?

High-level Reasoning Abilities

CoT

How costly is reasoning?

Lower Bounds for Chain-of-Thought Reasoning in Hard-Attention Transformers

Alireza Amiribavandpour, Xinting Huang, Mark Rofin, Michael Hahn



Without CoT

% 0 1 1 0 \$ e @

Without CoT



% 0 1 1 0 \$ e @

very difficult

even in-distribution
generalization is notoriously
difficult on long strings

Without CoT 

With CoT 

% 0 1 1 0 \$ e @

% 0 1 1 0 \$

e.g. Anil et al 2022

Without CoT 

% 0 1 1 0 \$ e @

With CoT 

% 0 1 1 0 \$ e

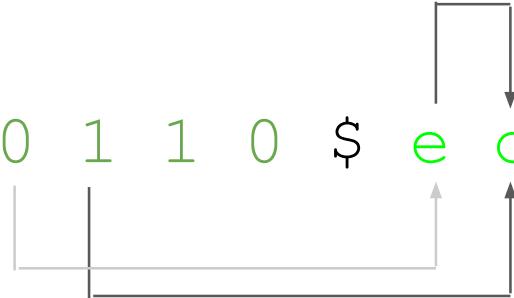


Without CoT 

% 0 1 1 0 \$ e @

With CoT 

% 0 1 1 0 \$ e o

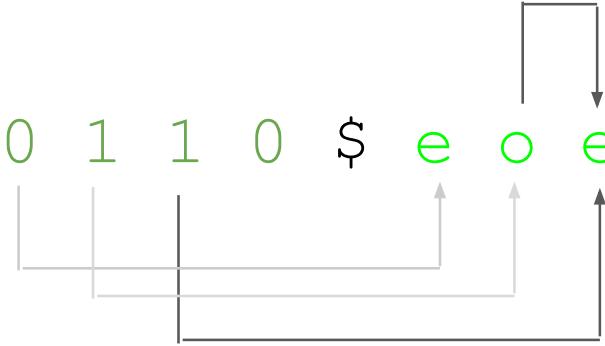


Without CoT 

% 0 1 1 0 \$ e @

With CoT 

% 0 1 1 0 \$ e o e

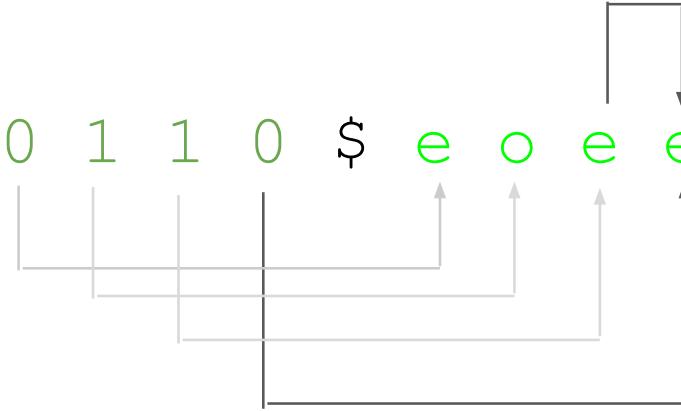


Without CoT 

% 0 1 1 0 \$ e @

With CoT 

% 0 1 1 0 \$ e o e e

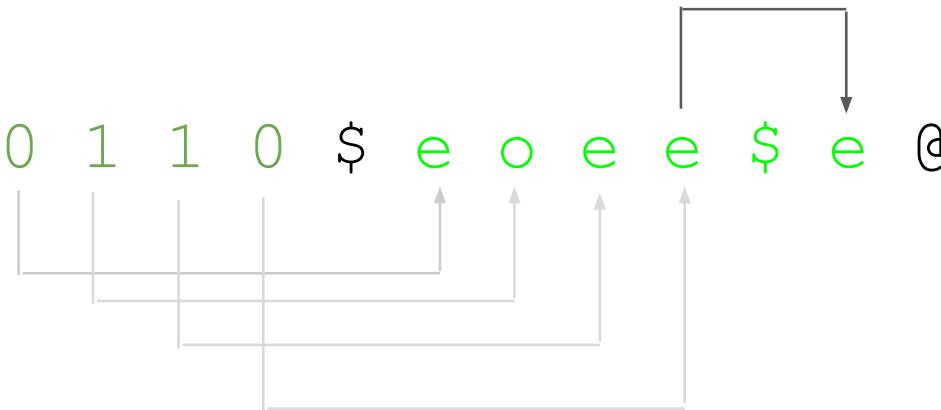


Without CoT 

% 0 1 1 0 \$ e @

With CoT 

% 0 1 1 0 \$ e o e e \$ e @



Without CoT

% 0 1 1 0 \$ e @

Got it — let's carefully trace the movements step by step:

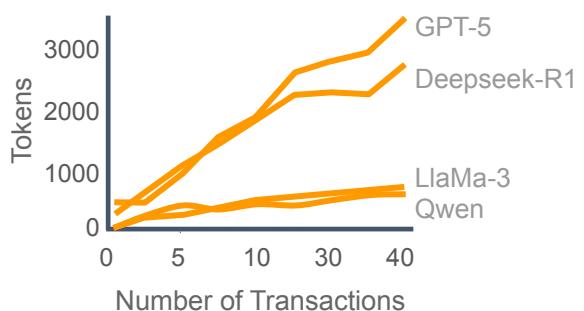
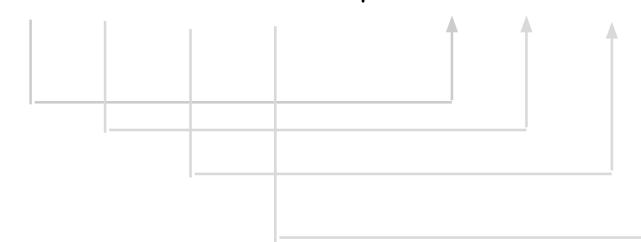
1. **Initial ownership:**
 - Paul → *encyclopedia*
 - Mary → *novel*
2. **Paul gave his book (encyclopedia) to Josh:**
 - Josh → *encyclopedia*
 - Paul → nothing
3. **Mary gave her book (novel) to Paul:**
 - Paul → *novel*
 - Mary → nothing
4. **Josh gave his book (encyclopedia) to Kim:**
 - Kim → *encyclopedia*
 - Josh → nothing
5. **Paul gave his book (novel) to Ruth:**
 - Ruth → *novel*
 - Paul → nothing

✓ Final ownership:

- Kim → *encyclopedia*
- Ruth → *novel*
- Paul, Mary, Josh → nothing

Answer: Kim has the encyclopedia.

With CoT

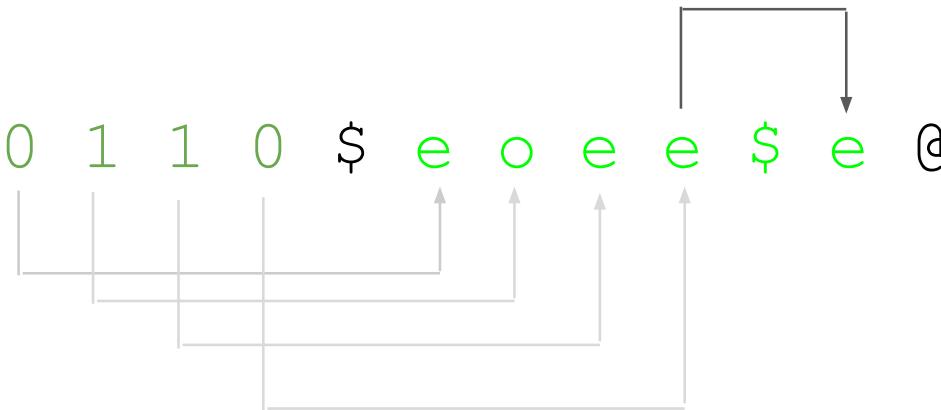


Without CoT

% 0 1 1 0 \$ e @

With CoT

% 0 1 1 0 \$ e o e e \$ e @



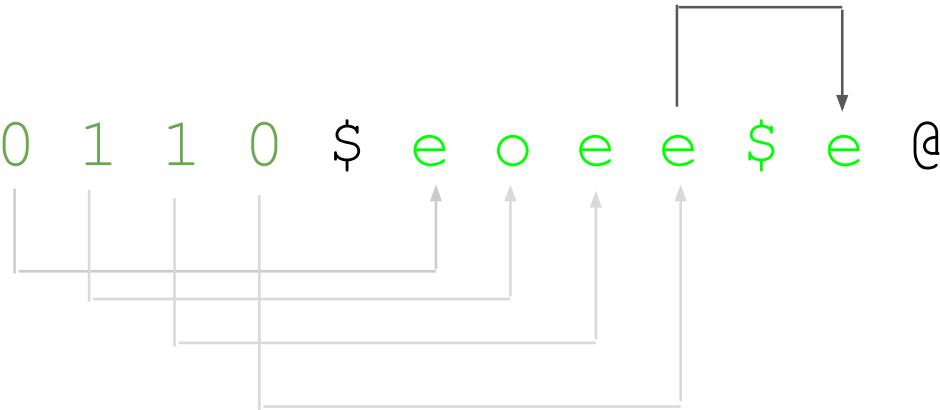
How much of this extra cost is needed?

Without CoT

% 0 1 1 0 \$ e @

With CoT

% 0 1 1 0 \$ e o e e \$ e @



How much of this extra cost is needed?

Can CoTs be condensed?

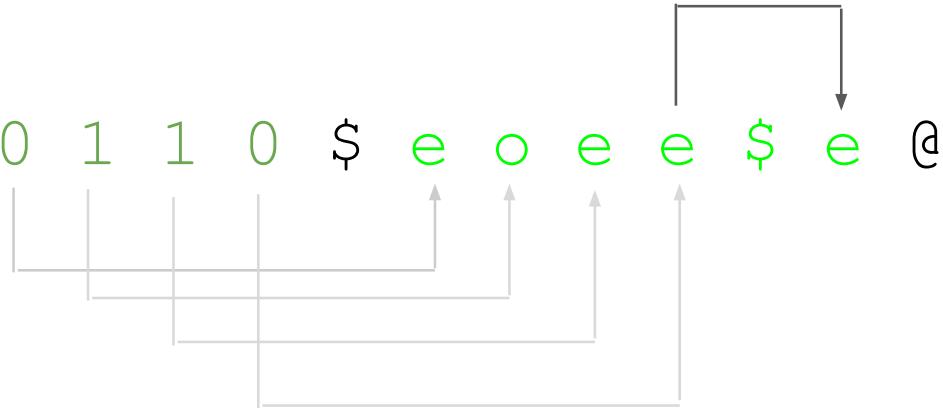
Perhaps to length $\log N$? \sqrt{N} ?

Without CoT

% 0 1 1 0 \$ e @

With CoT

% 0 1 1 0 \$ e o e e \$ e @



How much of this extra cost is needed?

Can CoTs be condensed?

Perhaps to length $\log N$? \sqrt{N} ?

This paper:
Mostly, no
compression
possible.

Given an algorithmic problem, how does the CoT length scale with input length N?

Given an algorithmic problem, how does the CoT length scale with input length N ?

Analogue to classical computational complexity, but for LLMs.

Our contribution: An Incompressibility Theorem for CoT.

Given an algorithmic problem, how does the CoT length scale with input length N ?

Analogue to classical computational complexity, but for LLMs.

Our contribution: An Incompressibility Theorem for CoT.

Results in this paper apply in the regime where numerical precision in transformer is fixed while $N \rightarrow \text{infty}$.

Equivalently to “unique hardmax attention”.

Upper Bound (prior work)

Theorem: If a Turing machine solves a problem in time $T(n)$, then a Transformer can solve with a CoT of length $O(T(n))$.

e.g. Perez et al 2019; Merrill and Sabharwal 2024; and others

Upper Bound (prior work)

Theorem: If a Turing machine solves a problem in time $T(n)$, then a Transformer can solve with a CoT of length $O(T(n))$.

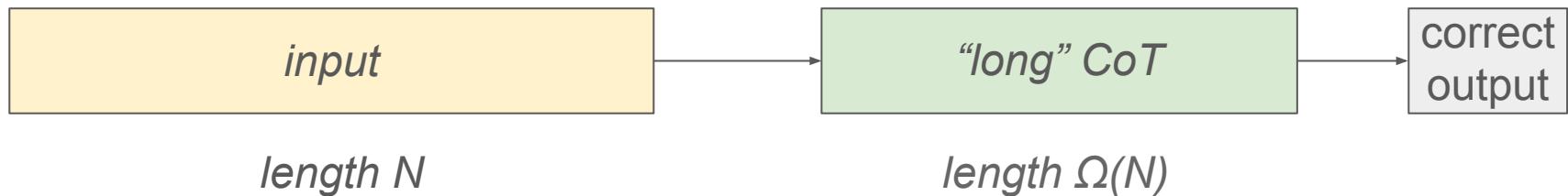
But this bound is not tight!

Transformers solve many problems without CoT.

Our Lower Bound

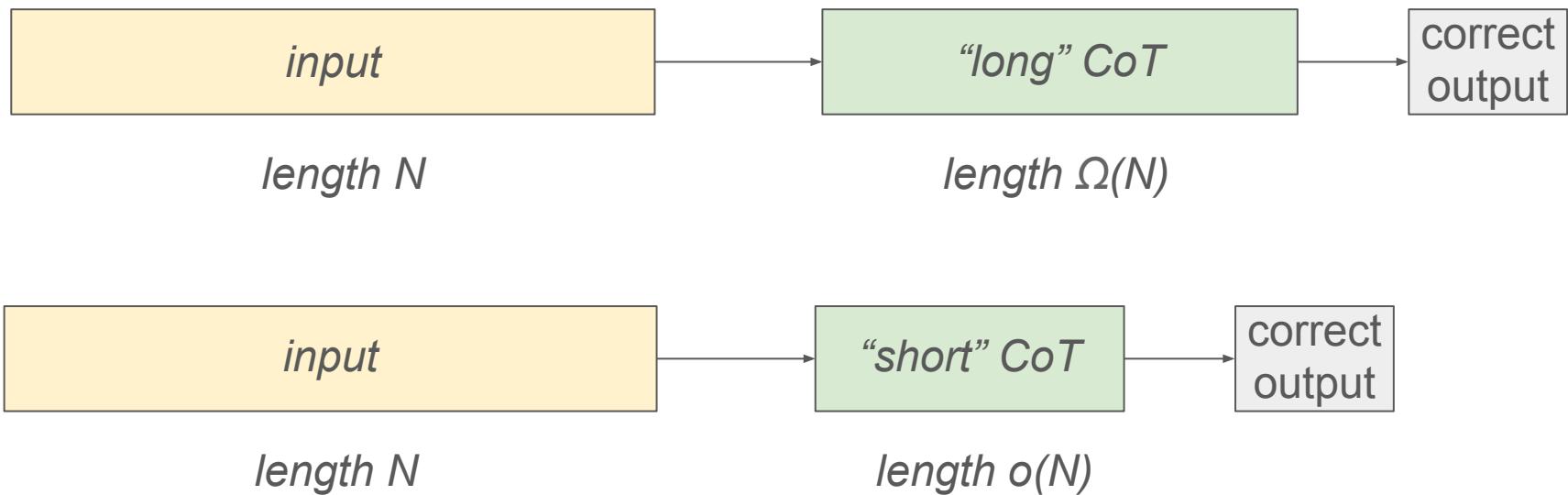
Our Lower Bound

Theorem



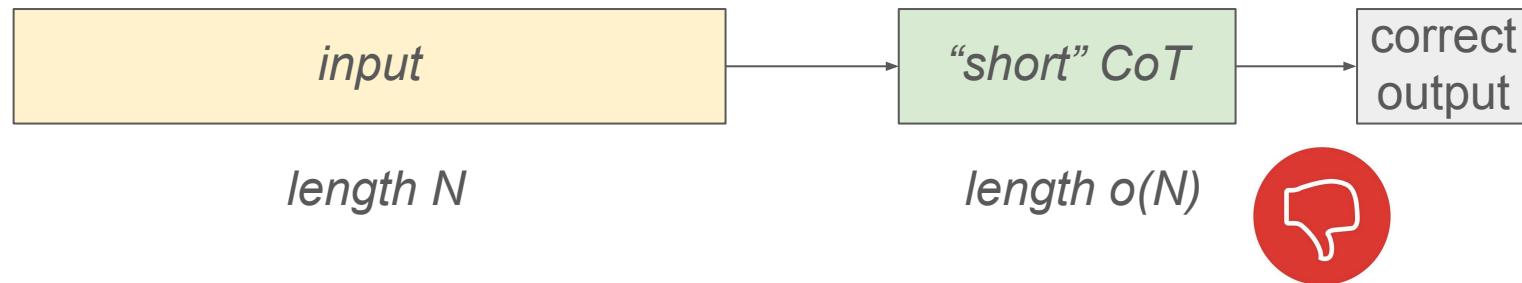
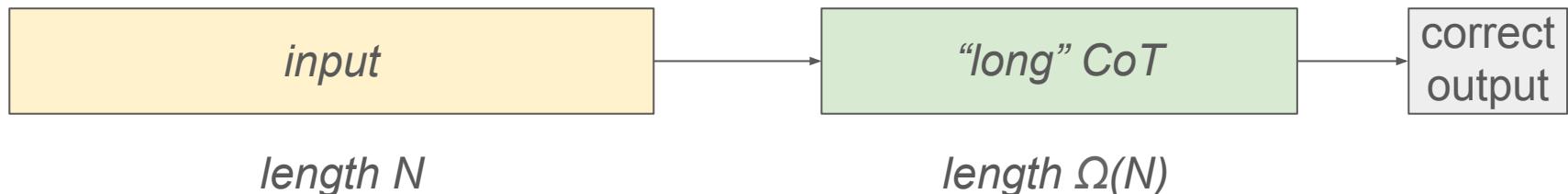
Our Lower Bound

Theorem



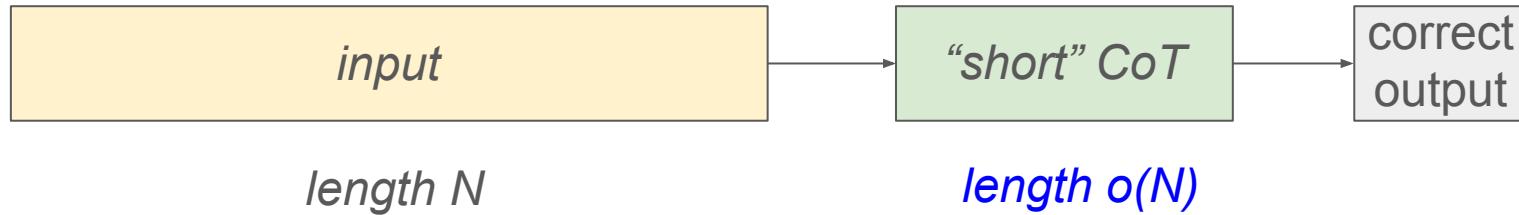
Our Lower Bound

Theorem

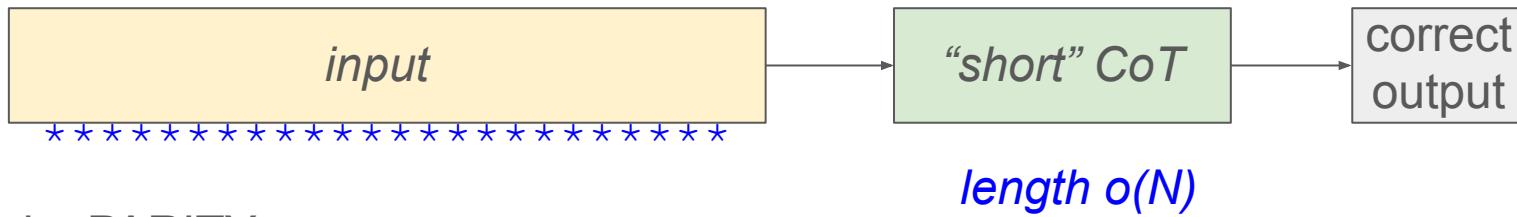


Our Lower Bound

Theorem



Our Lower Bound Theorem

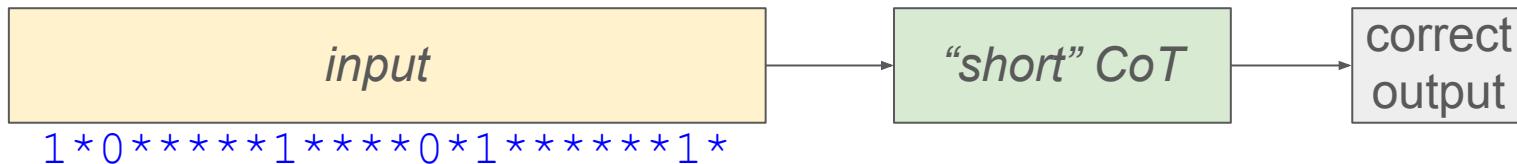


Example: PARITY

010110111001010110101001 \longrightarrow (some CoT) \longrightarrow "odd"

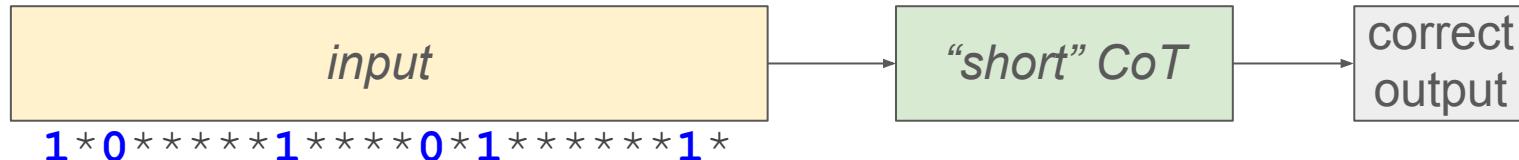
Our Lower Bound

Theorem



Our Lower Bound

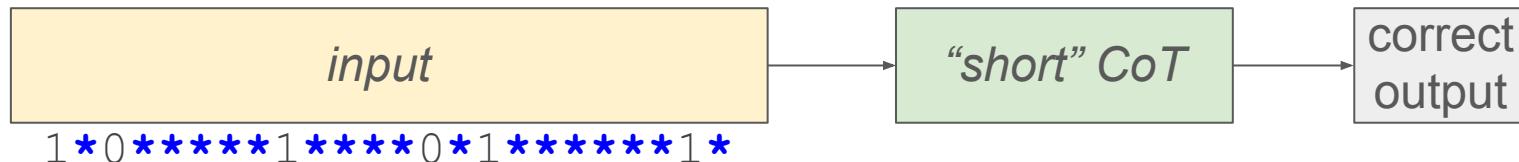
Theorem



fixed to a
specific input
symbol

Our Lower Bound

Theorem

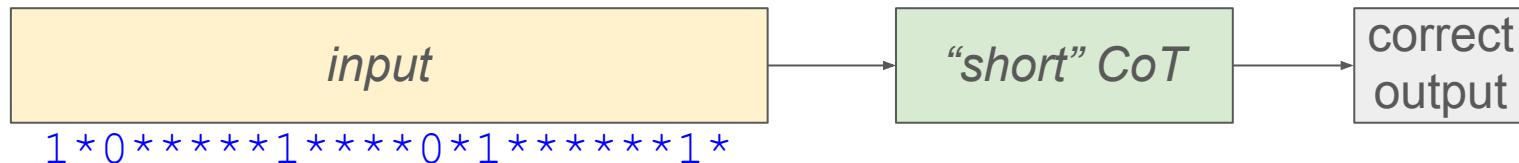


fixed to a
specific input
symbol

remain
unassigned

Our Lower Bound

Theorem



10010001**1**0110**011**011100**10**

10011011**1**0000**001**000101**11**

11000000**1**1111**001**101010**10**

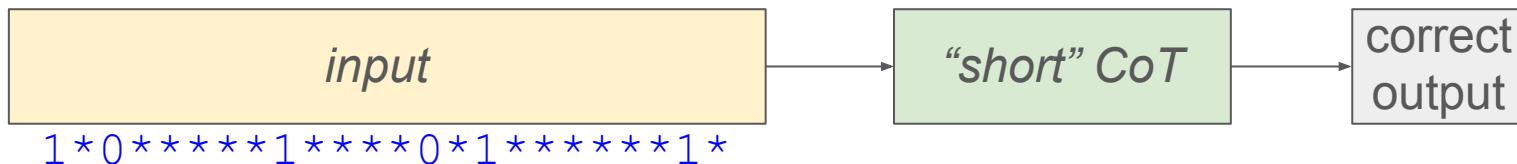
11011101**1**1110**011**010010**10**

10001001**1**0000**001**001111**11**

· · ·

Our Lower Bound

Theorem



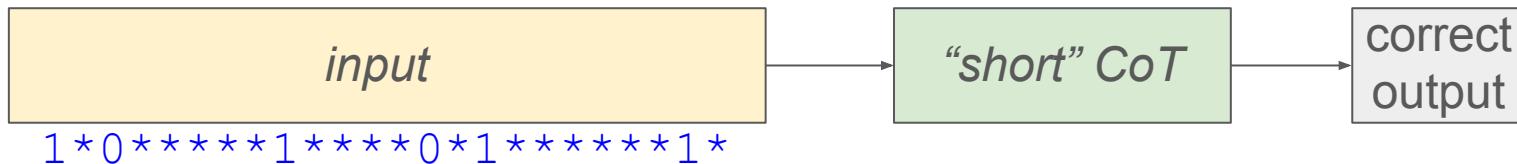
10010001**1**0110**011**011100**10**
10011011**1**0000**001**000101**11**
11000000**1**1111**001**10101010**10**
11011101**1**1110**011**01001010**10**
10001001**1**0000**001**0011111**11**
. . .

abbaab
abbaab
abbaab
abbaab
abbaab
. . .

*all matching
inputs generate
the same CoT*

Our Lower Bound

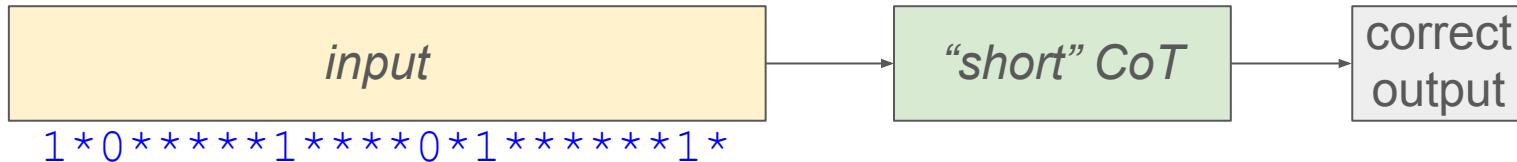
Theorem



In words:

Our Lower Bound

Theorem

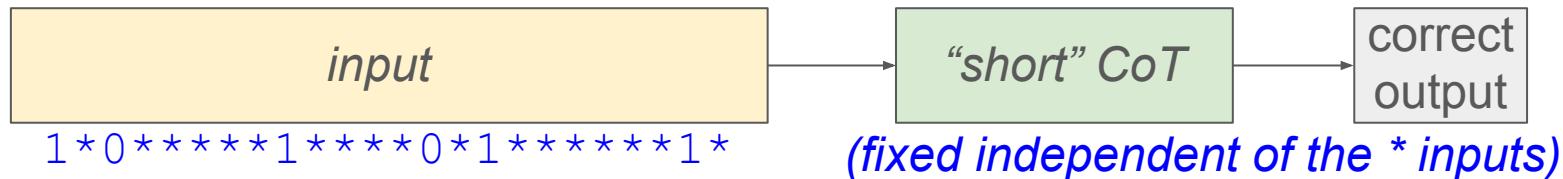


In words:

Assume a (fixed-precision) transformer computes a CoT of length $o(N)$.

Our Lower Bound

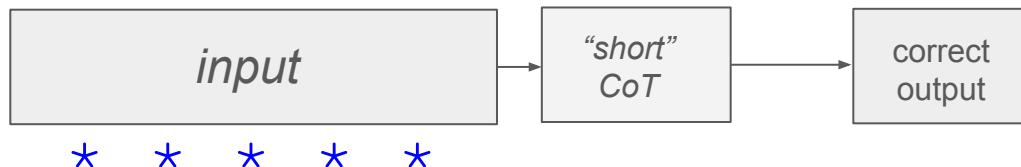
Theorem



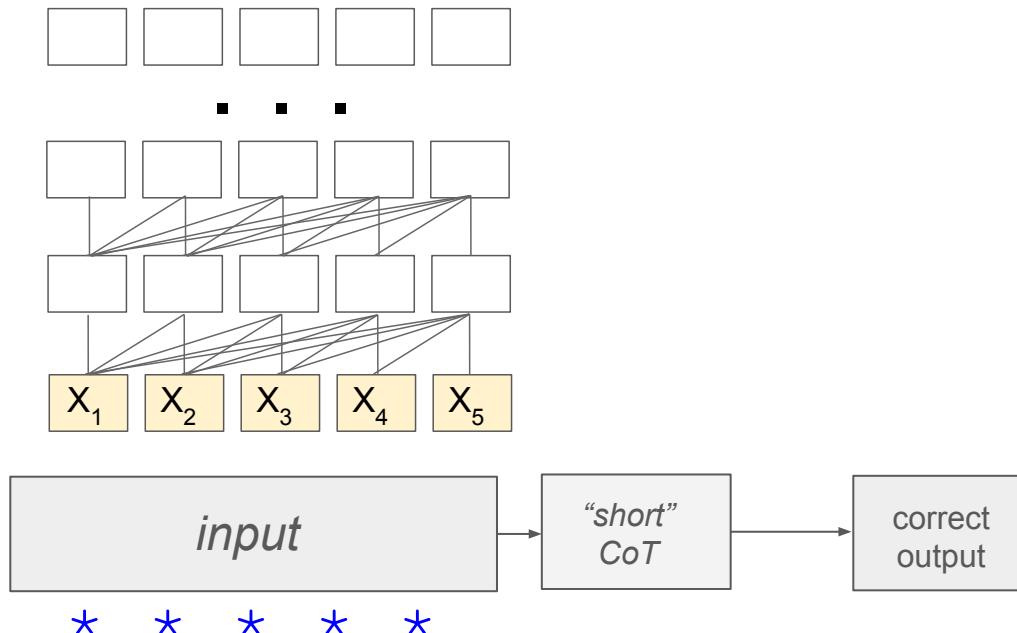
In words:

Assume a (fixed-precision) transformer computes a CoT of length $o(N)$.
One can fix a small fraction of input bits that makes the CoT constant on all compatible input strings.

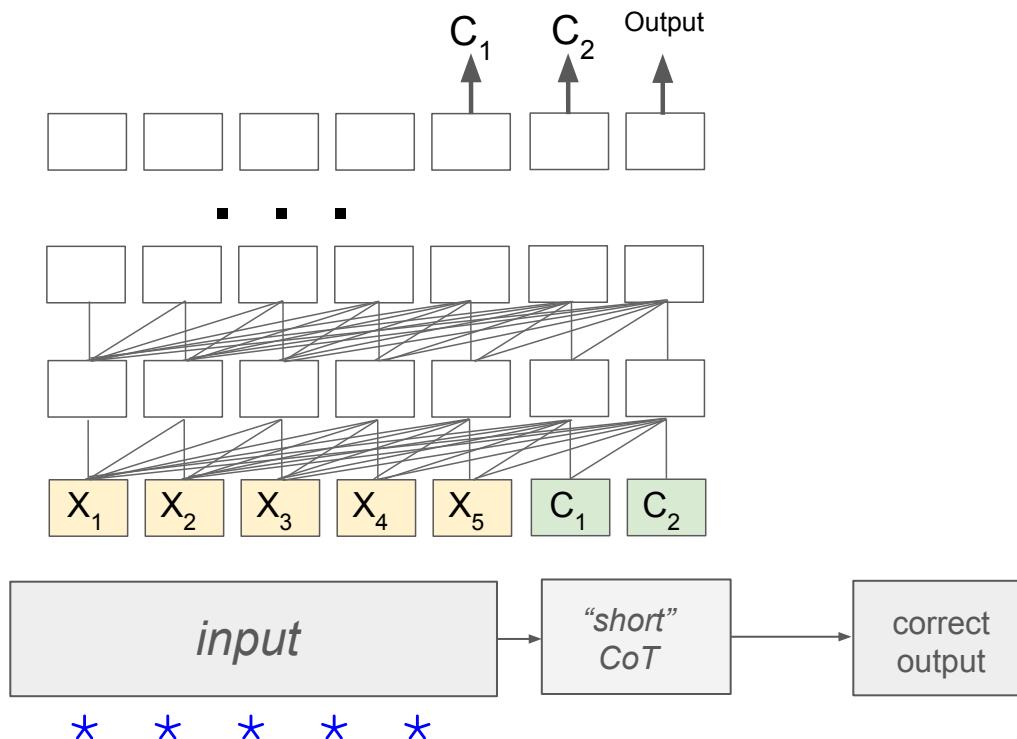
Proof Idea



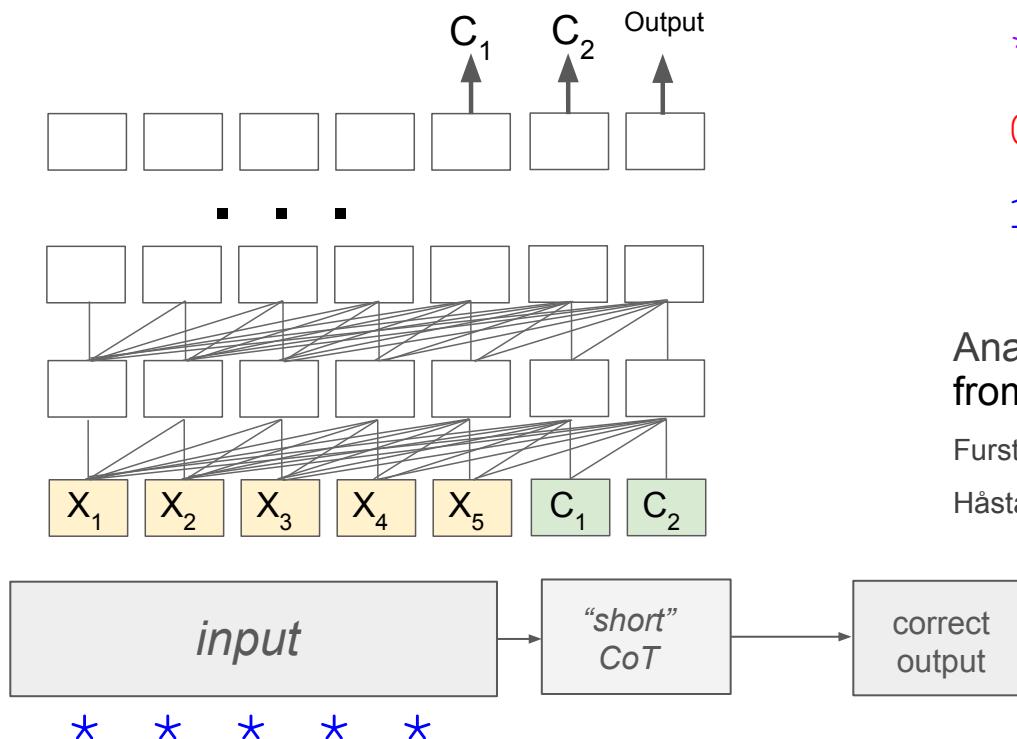
Proof Idea



Proof Idea



Proof Idea



Set each input token i.i.d. to

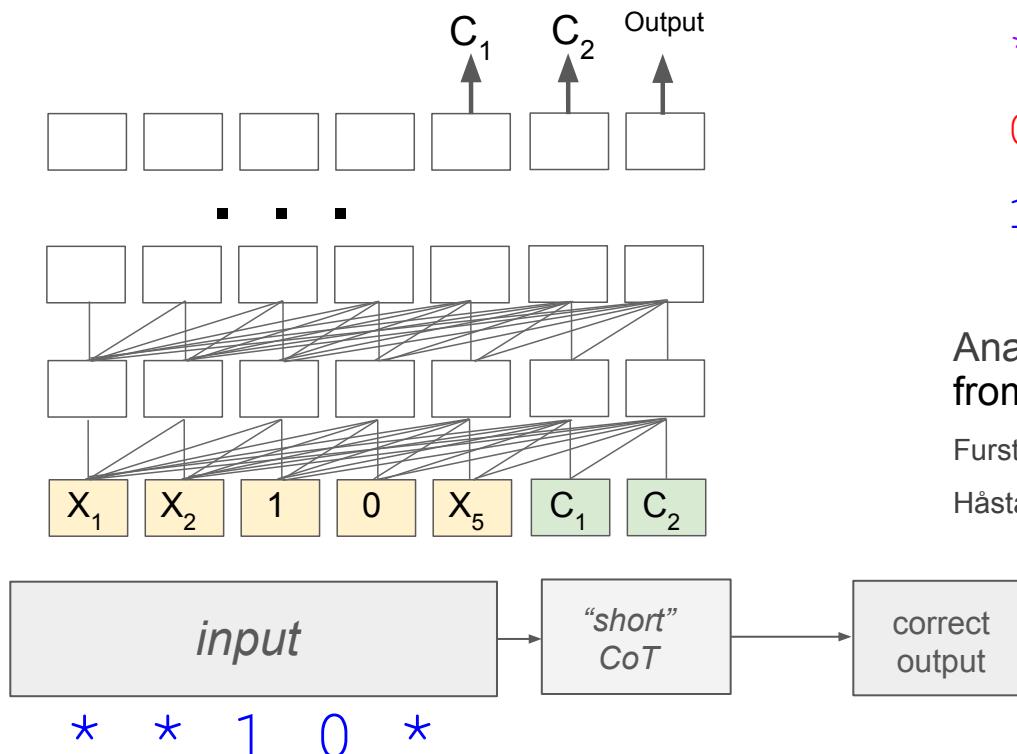
- * with $p=80\%$
- 0 with $p=10\%$
- 1 with $p=10\%$

Analogous to [Random Restrictions](#) from Circuit Complexity.

Furst, Saxe, Sipser 1984

Håstad 1986

Proof Idea



Set each input token i.i.d. to

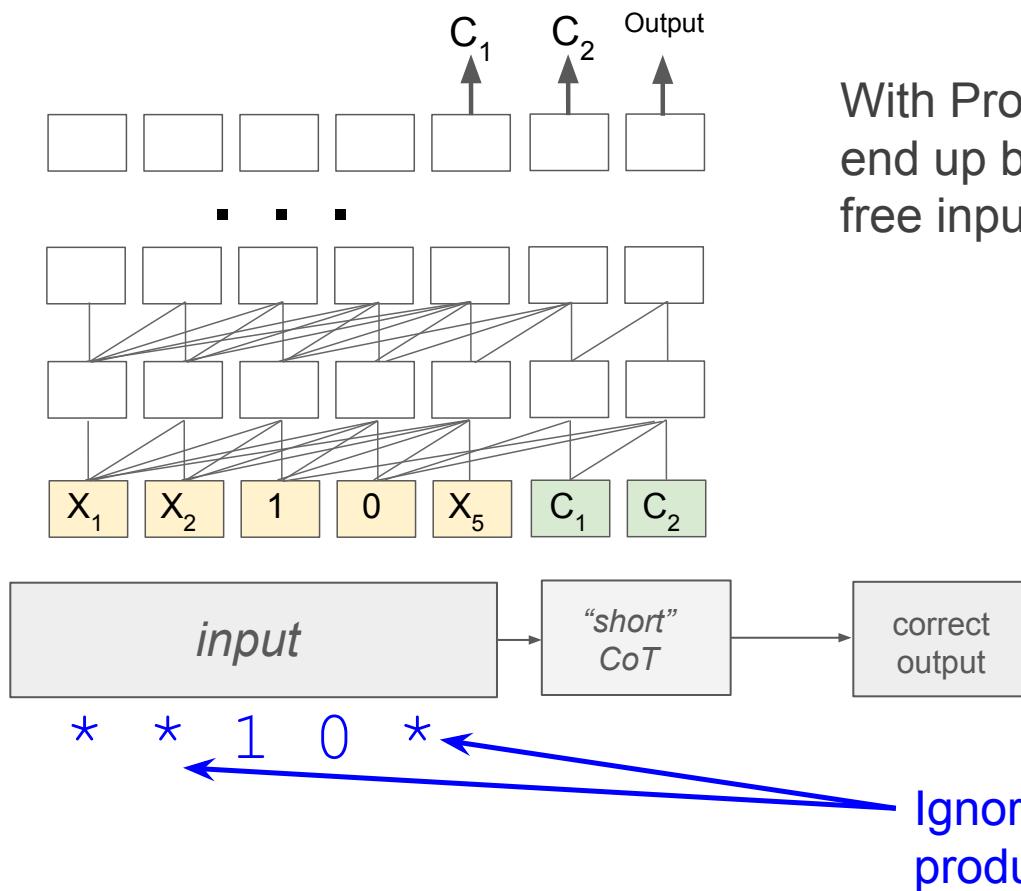
- \star with $p=80\%$
- 0 with $p=10\%$
- 1 with $p=10\%$

Analogous to **Random Restrictions** from Circuit Complexity.

Furst, Saxe, Sipser 1984

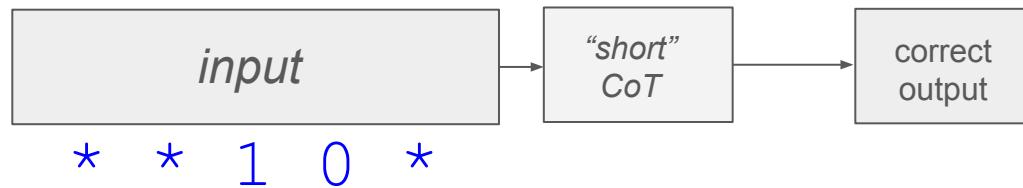
Håstad 1986

Proof Idea



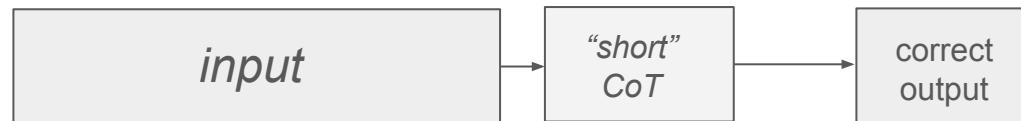
With Prob > 0, the transformer will end up being “distracted” from all free inputs.

Proof Idea



A General Lower Bound for CoT Length

Theorem



* * 1 0 *

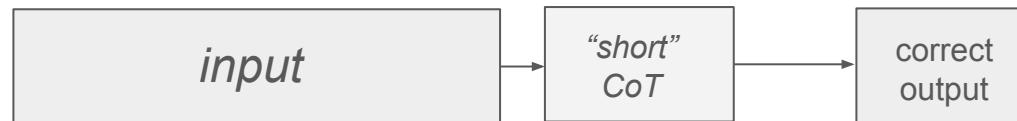
| | | | | | |
|---|---|----------|----------|---|----|
| 0 | 0 | 1 | 0 | 0 | ab |
| 0 | 1 | 1 | 0 | 0 | ab |
| 1 | 0 | 1 | 0 | 1 | ab |
| 0 | 0 | 1 | 0 | 1 | ab |

*all matching
inputs generate
the same CoT*

• • •

A General Lower Bound for CoT Length

Example: AND function



* * 1 0 *

0 0 1 0 0

0 1 1 0 0

1 0 1 0 1

0 0 1 0 1

ab

ab

ab

ab

ab

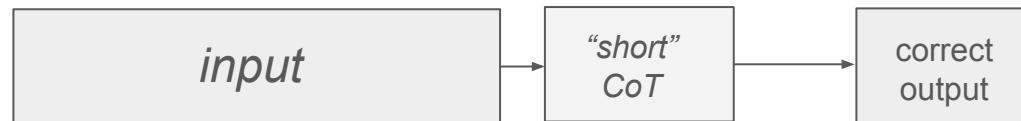
...

...



A General Lower Bound for CoT Length

Example: PARITY function



* * 1 0 *

0 0 1 0 0

ab

ODD

0 1 1 0 0

ab

EVEN

1 0 1 0 1

ab

ODD

0 0 1 0 1

ab

EVEN



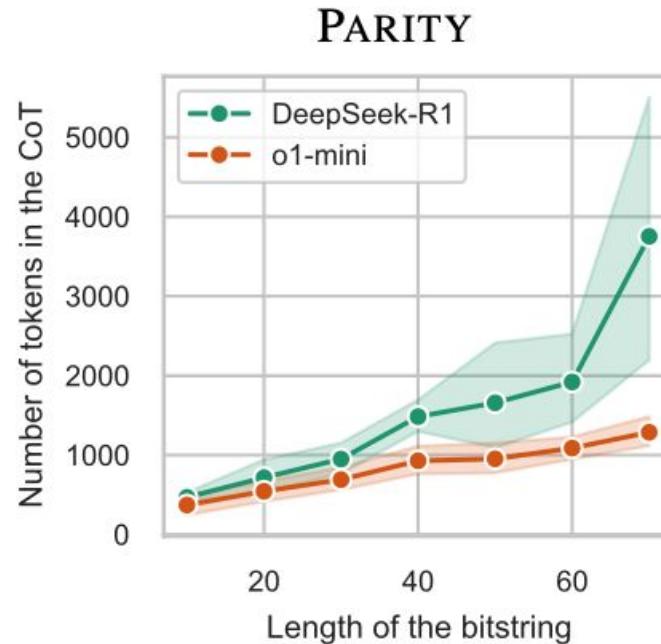
...

State Tracking

Corollary:

Any CoT for PARITY has length $\Omega(N)$.

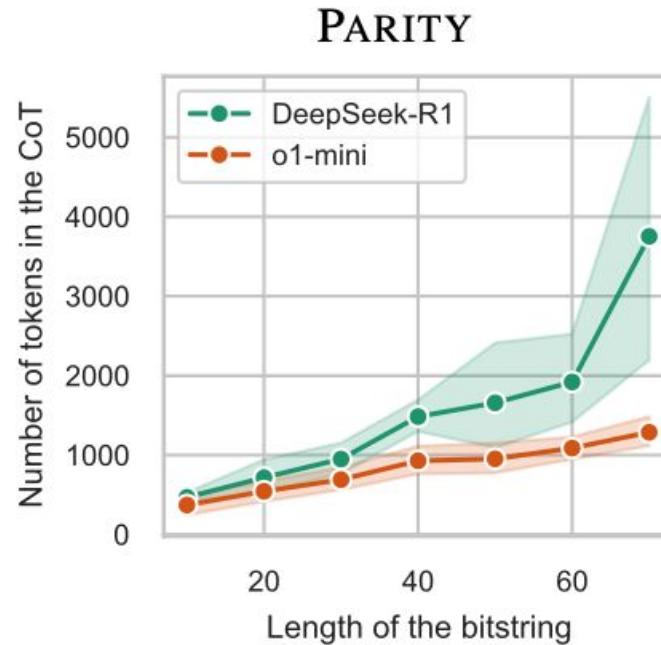
Parity



We only include
traces leading to the
correct answer

Empirical results of LLMs performing PARITY

Parity



We only include
traces leading to the
correct answer

The growth of CoT
appears at least linear

Empirical results of LLMs performing PARITY

Please answer very briefly, and without external resources (e.g. Python interpreter). What is 61157×4555 ?

The result of 61157×4555 is **278657635**.

Please answer very briefly, and without external resources (e.g. Python interpreter). What is 61157×4555 ?

The result of 61157×4555 is 278657635. *incorrect*

actual: 278570135

See also:

for GPT4 zero shot: Dziri et al, NeurIPS 2023.

for: o3-mini: <https://x.com/yuntiandeng/status/1889704768135905332>

for DeepSeek R1: <https://x.com/nouhadziri/status/1886423897567805835>

Multiplication

Theorem:

Under a combinatorial conjecture, multiplying two N -bit numbers requires a CoT of length $\Omega(N)$.

What is the upper bound?

“naive”
grade
school
algorithm:
 $\Theta(N^2)$ steps



To compute $61,157 \times 4,555$ using intermediate steps, let's break it down step-by-step.

We'll use the traditional method of multiplication, multiplying 61,157 by each digit of 4,555 (starting from the right), and then summing up the partial products.

Step 1: Break down the numbers

- 61,157 (the multiplicand)
- 4,555 (the multiplier)

Write 4,555 as:

$$4,555 = 4,000 + 500 + 50 + 5$$

Thus, we need to compute:

$$61,157 \times 4,555 = 61,157 \times (4,000 + 500 + 50 + 5)$$

This becomes:

$$(61,157 \times 4,000) + (61,157 \times 500) + (61,157 \times 50) + (61,157 \times 5)$$

Step 2: Compute partial products

1. $61,157 \times 5$

$$61,157 \times 5 = 305,785$$

2. $61,157 \times 50$

$$61,157 \times 50 = 3,057,850$$

3. $61,157 \times 500$

$$61,157 \times 500 = 30,578,500$$

4. $61,157 \times 4,000$

$$61,157 \times 4,000 = 244,628,000$$

Step 3: Add the partial products

Now, let's sum all the partial products:

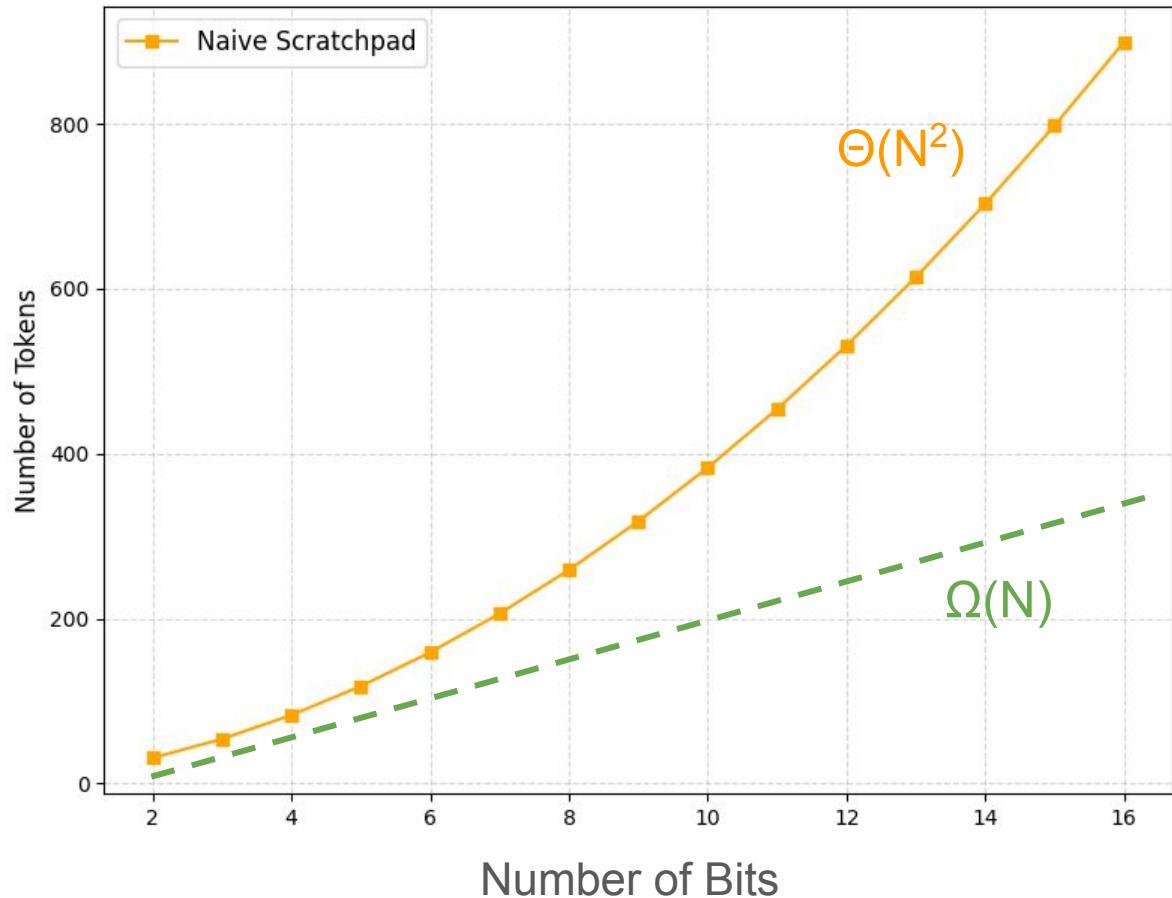
1. 305,785

2. 3,057,850

3. 30,578,500

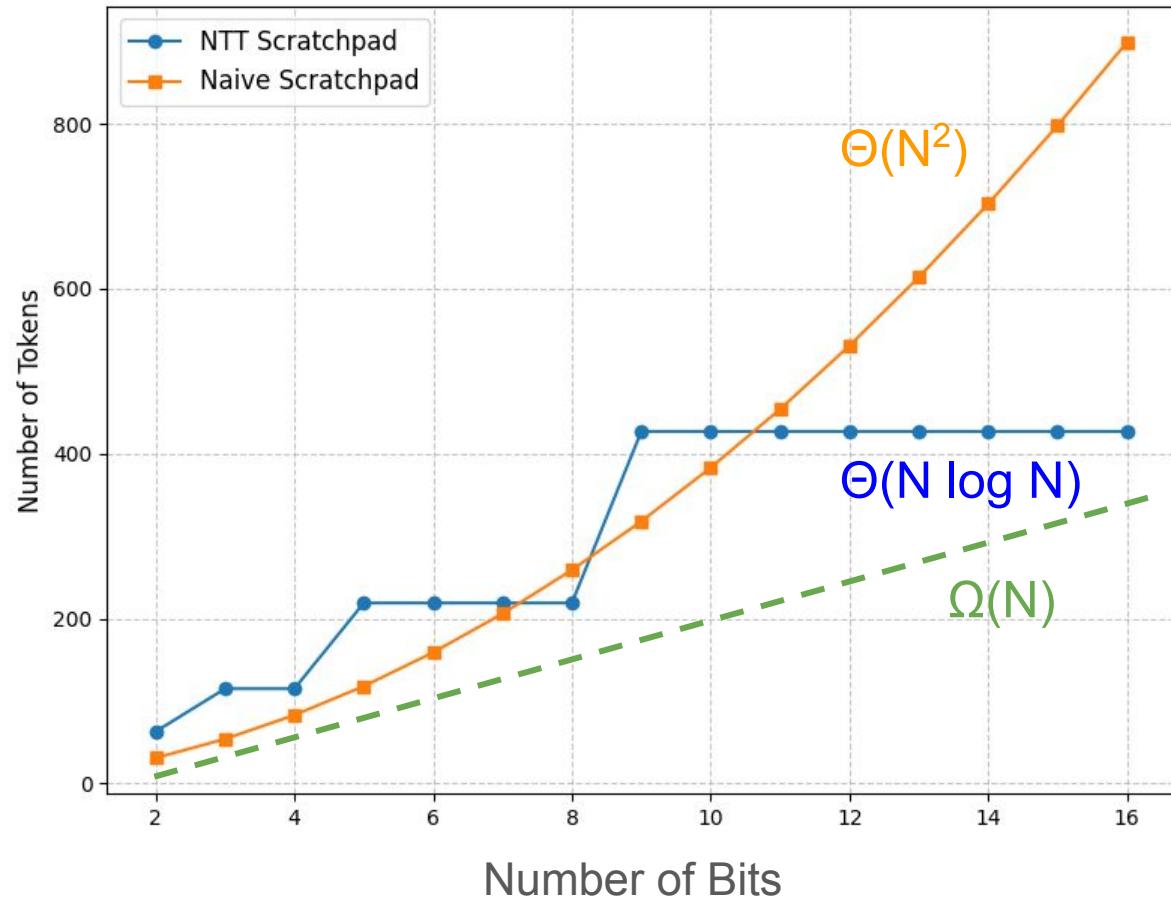
4. 244,628,000

Naive CoT requires $\Theta(N^2)$ steps.



Naive CoT requires
 $\Theta(N^2)$ steps.

CoT with $\Theta(N \log N)$ steps exists.



Order Statistics

8 45 82 71 5 28 65

list of N integers

Order Statistics

8 45 82 71 5 28 65 -> 45

list of N integers median

Order Statistics

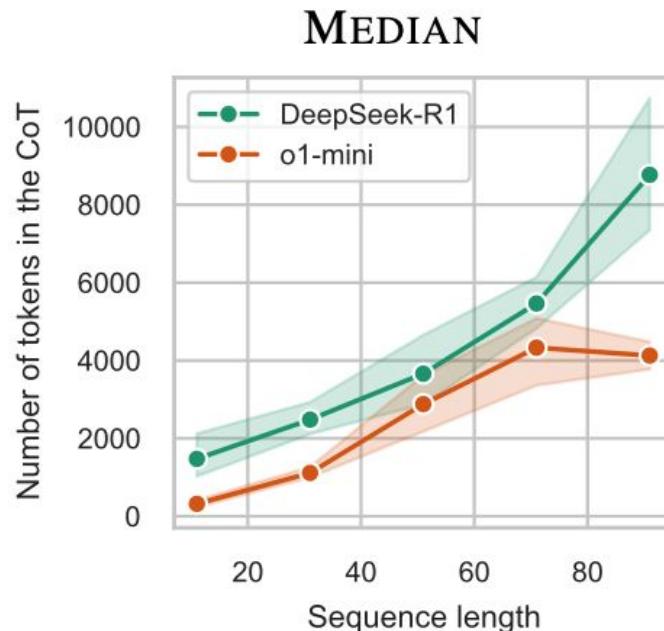
8 45 82 71 5 28 65 -> 45

list of N integers median

Theorem:

For MEDIAN, any CoT requires length $\Omega(N)$. This bound is attained.

Order Statistics

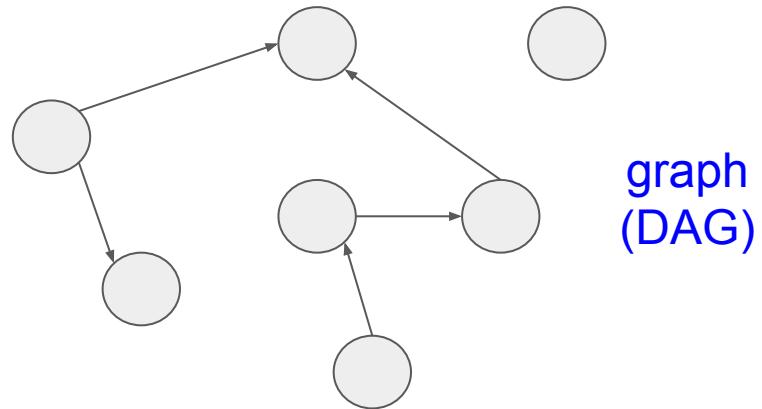


We only include
traces leading to the
correct answer

the growth of CoT
appears at least linear

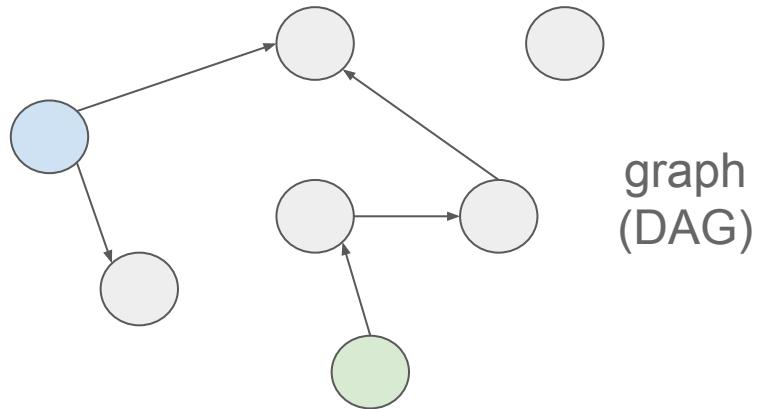
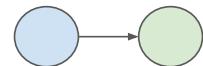
Empirical results of LLMs performing MEDIAN

DAG reachability



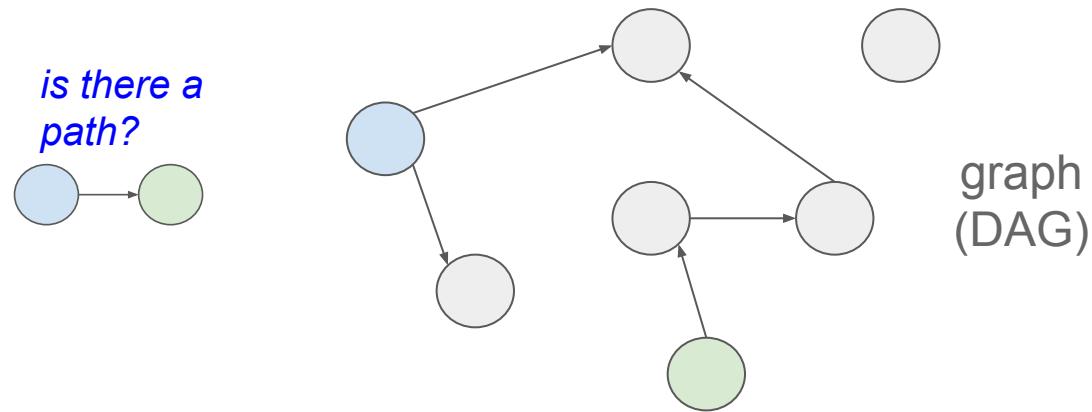
DAG reachability

*is there a
path?*



graph
(DAG)

DAG reachability



Theorem:

Any CoT requires length $\Omega(|E|\log|V|)$ in the worst case.

This bound is attained.

Barriers on Hidden Reasoning

Pfau, Jacob, William Merrill, and Samuel R. Bowman.
"Let's think dot by dot: Hidden computation in
transformer language models." COLM 2024.

Barriers on Hidden Reasoning

$e=2.71828\dots$

Prompt: "How many of the first 6 digits of **e** are >5?"

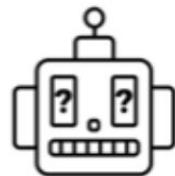
Pfau, Jacob, William Merrill, and Samuel R. Bowman.
"Let's think dot by dot: Hidden computation in
transformer language models." COLM 2024.

Barriers on Hidden Reasoning

$e=2.71828\dots$

Prompt: "How many of the first 6 digits of **e** are >5?"

Immediate answer



"7 digits are greater than 5"

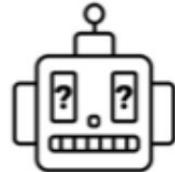
Pfau, Jacob, William Merrill, and Samuel R. Bowman.
"Let's think dot by dot: Hidden computation in
transformer language models." COLM 2024.

Barriers on Hidden Reasoning

$e=2.71828\dots$

Prompt: "How many of the first 6 digits of **e** are >5?"

Immediate answer



"7 digits are greater than 5"

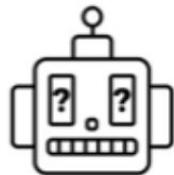
Pfau, Jacob, William Merrill, and Samuel R. Bowman.
"Let's think dot by dot: Hidden computation in
transformer language models." COLM 2024.

Barriers on Hidden Reasoning

$e=2.71828\dots$

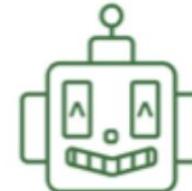
Prompt: "How many of the first 6 digits of e are >5?"

Immediate answer



"7 digits are greater than 5"

Chain of thought



LM Continuations

"2<5, 7>5, 1<5, 8>5, 2<5, 8>5,
that's 3 digits"

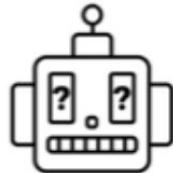
Pfau, Jacob, William Merrill, and Samuel R. Bowman.
"Let's think dot by dot: Hidden computation in
transformer language models." COLM 2024.

Barriers on Hidden Reasoning

$e=2.71828\dots$

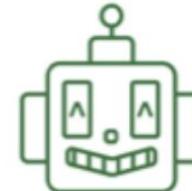
Prompt: "How many of the first 6 digits of e are >5?"

Immediate answer



"7 digits are greater than 5"

Chain of thought



LM Continuations

"2<5, 7>5, 1<5, 8>5, 2<5, 8>5,
that's 3 digits"



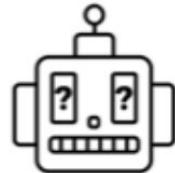
Pfau, Jacob, William Merrill, and Samuel R. Bowman.
"Let's think dot by dot: Hidden computation in
transformer language models." COLM 2024.

Barriers on Hidden Reasoning

$e=2.71828\dots$

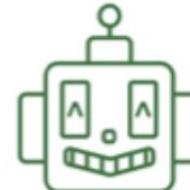
Prompt: "How many of the first 6 digits of **e** are >5 ?"

Immediate answer



"7 digits are greater than 5"

Chain of thought



LM Continuations

"2<5, 7>5, 1<5, 8>5, 2<5, 8>5,
that's 3 digits"



Filler tokens



" 3 digits"

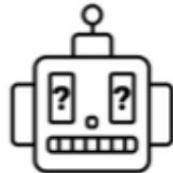
Pfau, Jacob, William Merrill, and Samuel R. Bowman.
"Let's think dot by dot: Hidden computation in
transformer language models." COLM 2024.

Barriers on Hidden Reasoning

$e=2.71828\dots$

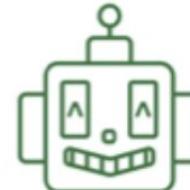
Prompt: "How many of the first 6 digits of **e** are >5 ?"

Immediate answer



"7 digits are greater than 5"

Chain of thought

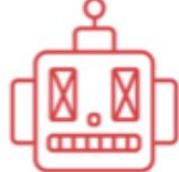


LM Continuations

"2<5, 7>5, 1<5, 8>5, 2<5, 8>5,
that's 3 digits"



Filler tokens



" 3 digits"

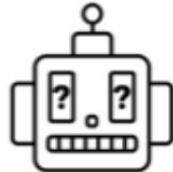
Pfau, Jacob, William Merrill, and Samuel R. Bowman.
"Let's think dot by dot: Hidden computation in
transformer language models." COLM 2024.

Barriers on Hidden Reasoning

$e=2.71828\dots$

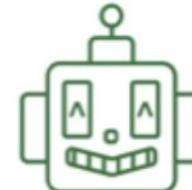
Prompt: "How many of the first 6 digits of e are >5?"

Immediate answer



"7 digits are greater than 5"

Chain of thought

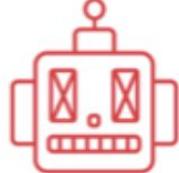


LM Continuations

"2<5, 7>5, 1<5, 8>5, 2<5, 8>5,
that's 3 digits"



Filler tokens



" 3 digits"



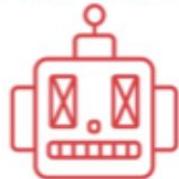
Pfau, Jacob, William Merrill, and Samuel R. Bowman.
"Let's think dot by dot: Hidden computation in
transformer language models." COLM 2024.

Barriers on Hidden Reasoning

$e=2.71828\dots$

Prompt: "How many of the first 6 digits of e are >5?"

Filler tokens



" 3 digits"



Pfau, Jacob, William Merrill, and Samuel R. Bowman.
"Let's think dot by dot: Hidden computation in
transformer language models." COLM 2024.

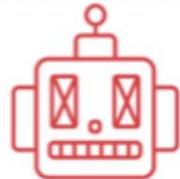
Barriers on Hidden Reasoning

$e=2.71828\dots$

Prompt: "How many of the first 6 digits of e are >5?"

How powerful are such “dot-by-dot” CoTs?

Filler tokens



“ 3 digits”



Pfau, Jacob, William Merrill, and Samuel R. Bowman.
"Let's think dot by dot: Hidden computation in
transformer language models." COLM 2024.

Barriers on Hidden Reasoning

$e=2.71828\dots$

Prompt: “How many of the first 6 digits of **e** are >5?”

How powerful are such “dot-by-dot” CoTs?

How well can transformers hide their reasoning?



“ 3 digits”



Pfau, Jacob, William Merrill, and Samuel R. Bowman.
"Let's think dot by dot: Hidden computation in
transformer language models." COLM 2024.

Barriers on Hidden Reasoning

Theorem:

“Dot-by-dot” CoTs for PARITY exist...

Barriers on Hidden Reasoning

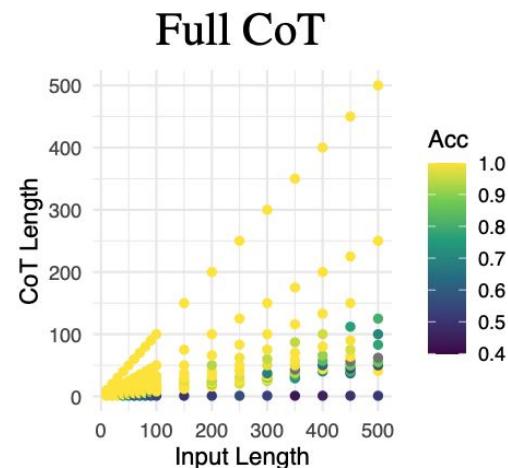
Theorem:

“Dot-by-dot” CoTs for PARITY exist, but require **super-polynomial** length.

Barriers on Hidden Reasoning

Theorem:

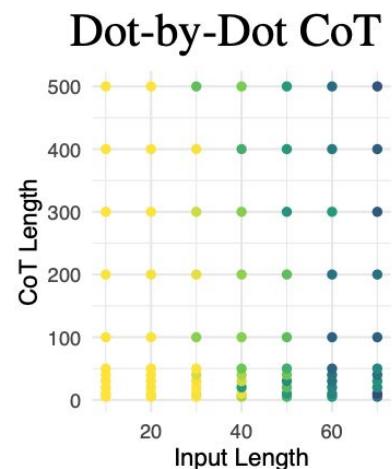
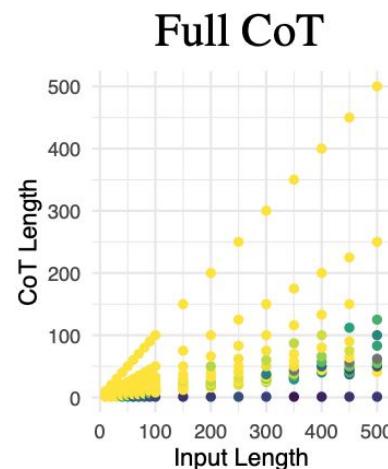
“Dot-by-dot” CoTs for PARITY exist, but require super-polynomial length.



Barriers on Hidden Reasoning

Theorem:

“Dot-by-dot” CoTs for PARITY exist, but require super-polynomial length.



Empirical results of transformers *trained* on PARITY

Barriers on Hidden Reasoning

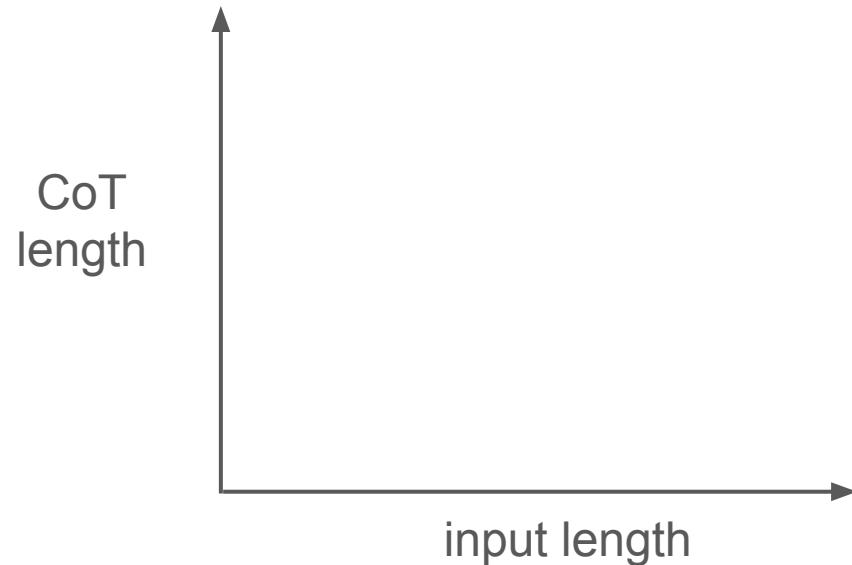
Theorem:

“Dot-by-dot” CoTs for PARITY exist, but require super-polynomial length.

Hidden Reasoning is possible,
but very expensive.

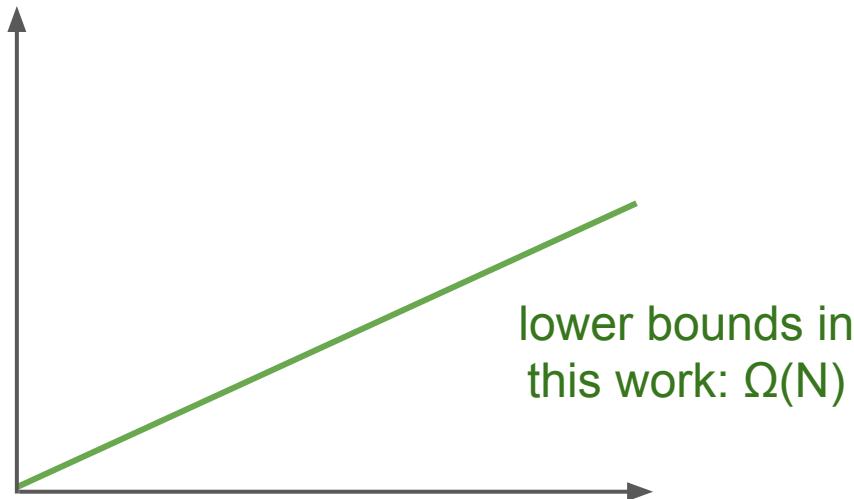
Questions for Future Work

Q3: Superlinear lower bounds for CoTs



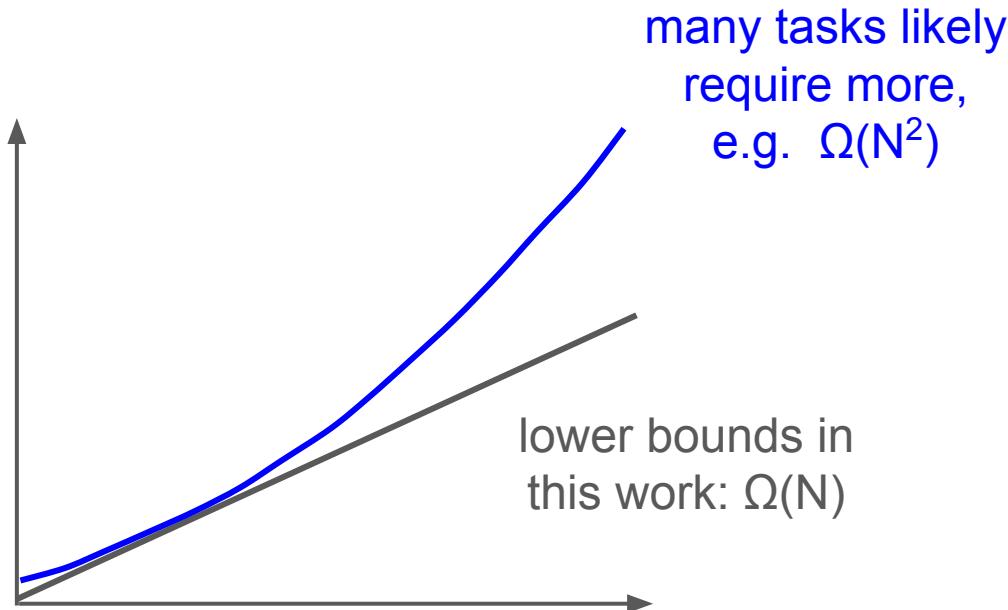
Questions for Future Work

Q3: Superlinear lower bounds for CoTs



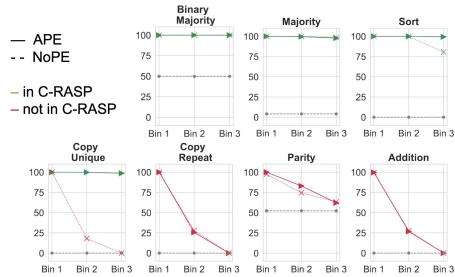
Questions for Future Work

Q3: Superlinear lower bounds for CoTs

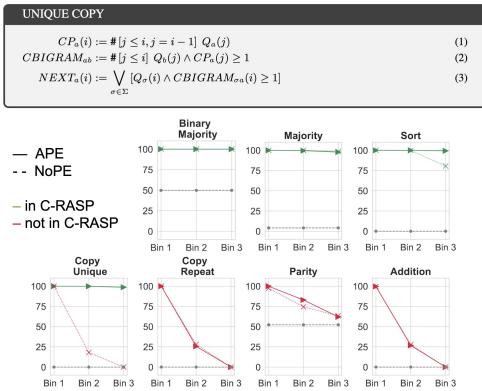


Length Generalization

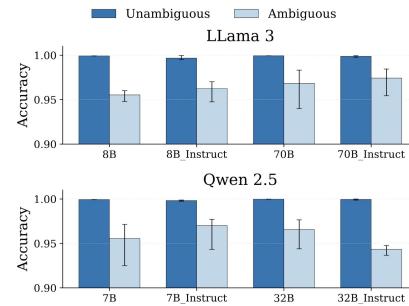
| UNIQUE COPY | |
|--|-----|
| $CP_a(i) := \#\{j \leq i, j = i - 1\} Q_a(j)$ | (1) |
| $CBIGRAM_{ab} := \#\{j \leq i\} Q_b(j) \wedge CP_a(j) \geq 1$ | (2) |
| $NEXT_n(i) := \bigvee_{\sigma \in \Sigma} \{Q_\sigma(i) \wedge CBIGRAM_{\sigma n}(i) \geq 1\}$ | (3) |



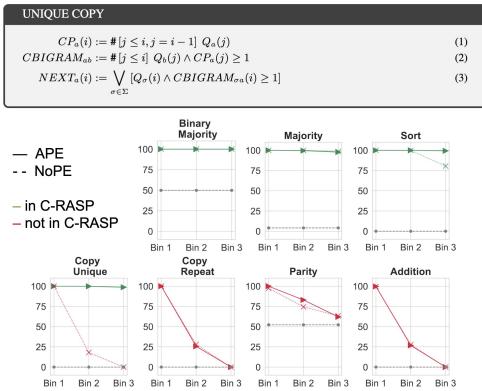
Length Generalization



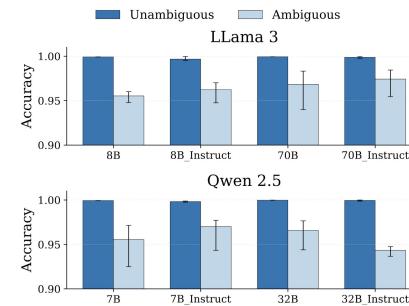
Constrains LLM Abilities



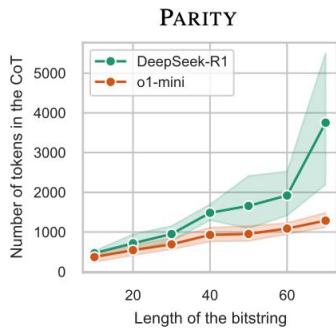
Length Generalization



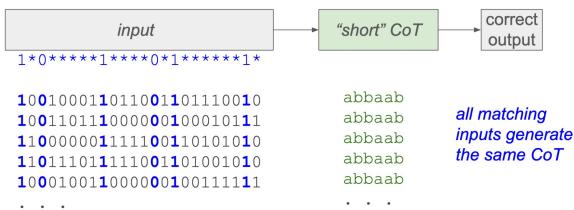
Constrains LLM Abilities



Lower Bounds for CoT



Theorem



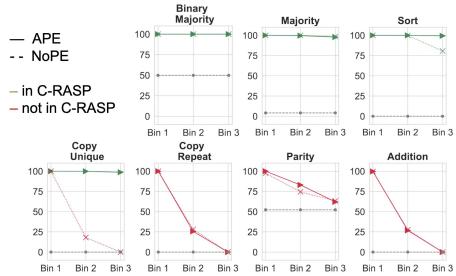
Length Generalization

UNIQUE COPY

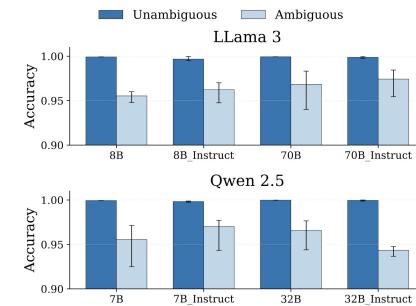
$$CP_n(i) := \#\{j \leq i, j = i-1] \mid Q_n(j)\} \quad (1)$$

$$CBIGRAM_{ab} := \#\{j \leq i] \mid Q_b(j) \wedge CP_a(j) \geq 1\} \quad (2)$$

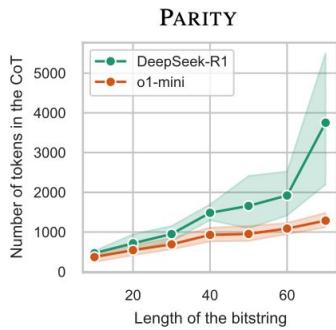
$$NEXT_n(i) := \bigvee_{\sigma \in \Sigma} \{Q_\sigma(i) \wedge CBIGRAM_{\sigma n}(i) \geq 1\} \quad (3)$$



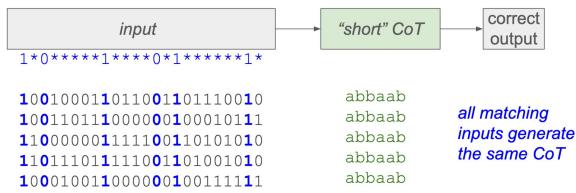
Constrains LLM Abilities



Lower Bounds for CoT



Theorem



Barriers against Hidden Reasoning

