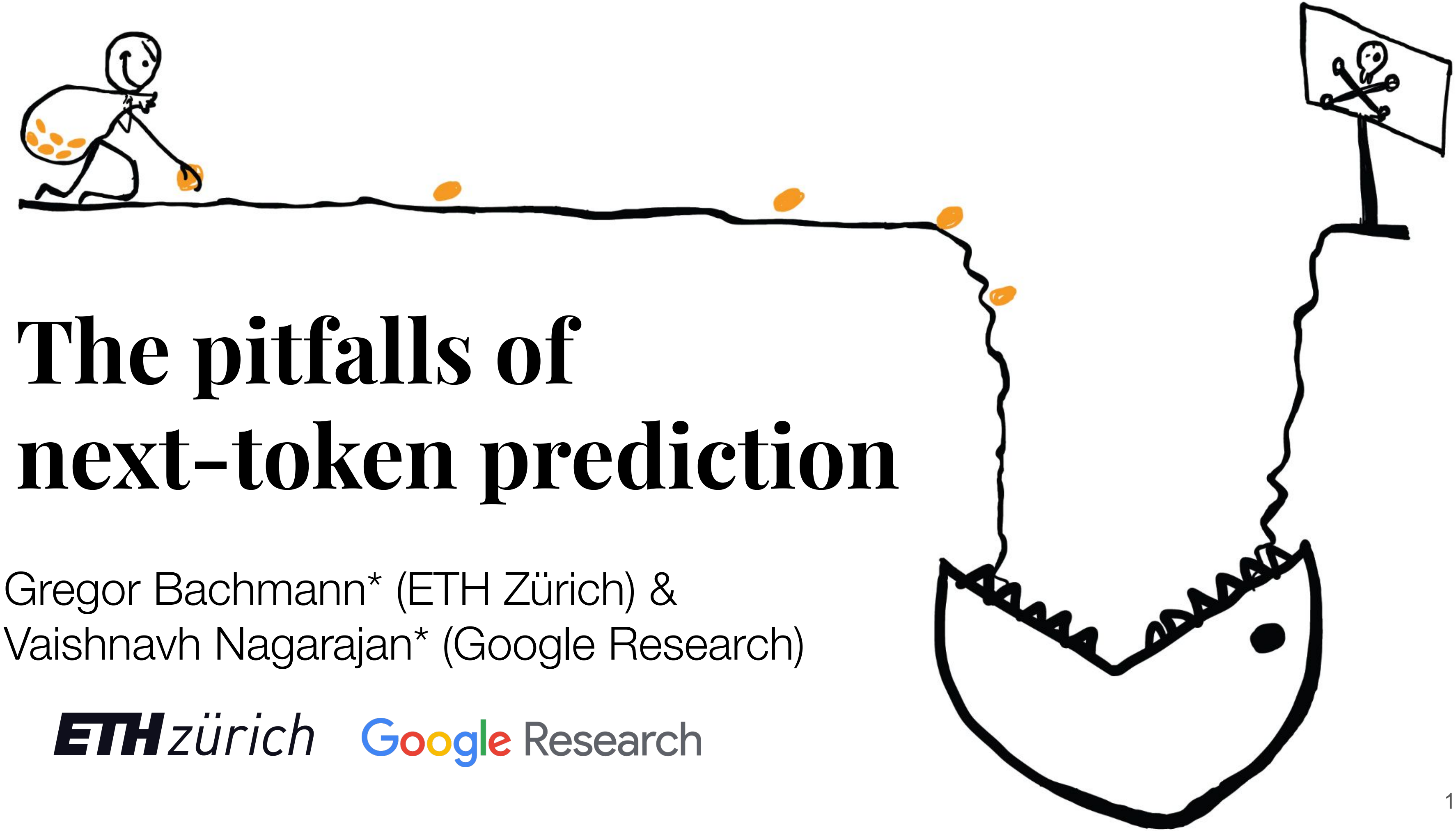


The pitfalls of next-token prediction

Gregor Bachmann* (ETH Zürich) &
Vaishnavh Nagarajan* (Google Research)

ETH zürich Google Research

The next-token prediction debate

Yann LeCun @ylecun

Haha.

Translate post

> **Pedro Domingos** @pmddomingos · Jun 12
>
> Imagine if you talked by picking the next word to be the one people most often say after your previous words.

Richard Ngo @RichardMCNgo

"LLMs are just doing next-token prediction without any understanding" is by now so clearly false it's no longer worth debating.

A highly contentious but fragmented debate!

Ya[…] @[…]

To clarify: there will be language models in 5 years, but they won't be auto-regressive.
Because auto-regressive models are uncontrollable and suffer from exponential divergence as more tokens are produced.

2:15 PM · Mar 25, 2023 · **20.3K** Views

From "Sparks of AGI":

These examples illustrate some of the limitations of the next-word prediction paradigm, which manifest as the model's lack of planning, working memory, ability to backtrack, and reasoning abilities. The model relies on a local and greedy process of generating the next word, without any global or deep understanding of the task or the output. Thus, the model is good at producing fluent and coherent texts, but has limitations

This talk:

- **Part I:** What is missing on both sides
- **Part II & III:** Crystallize a new failure of next-token prediction (NTP)
- **Part IV:** A possible fix: multi-token prediction

Part I: *What's missing on both sides*

Pessimists

If humans simply uttered the next-token, we'd be speaking gibberish.

Even tiny next-token errors snowball exponentially [1, 2, 3]:

Pr[`all tokens correct`]

$$= (1-\epsilon) \times (1-\epsilon) \times (1-\epsilon)\ldots$$

Optimists

By chain rule of probability, *any* distribution can be represented by next-token prediction (NTP)!

$$\Pr[t_1\ t_2\ t_3\ \ldots] = \Pr[\,t_1\,] \times \Pr[\,t_2 \mid t_1\,] \times \Pr[\,t_3 \mid t_1\ t_2\,]\ldots$$

You're just using the NTP backbone incorrectly. Wrap a verifier/backtracker!

Pessimists

If humans simply uttered the next-token, we'd be speaking gibberish.

Even tiny next-token errors propagate exponentially:

Pr[`all tokens correct`]
= (1-є) x (1-є) x (1-є)...

Optimists

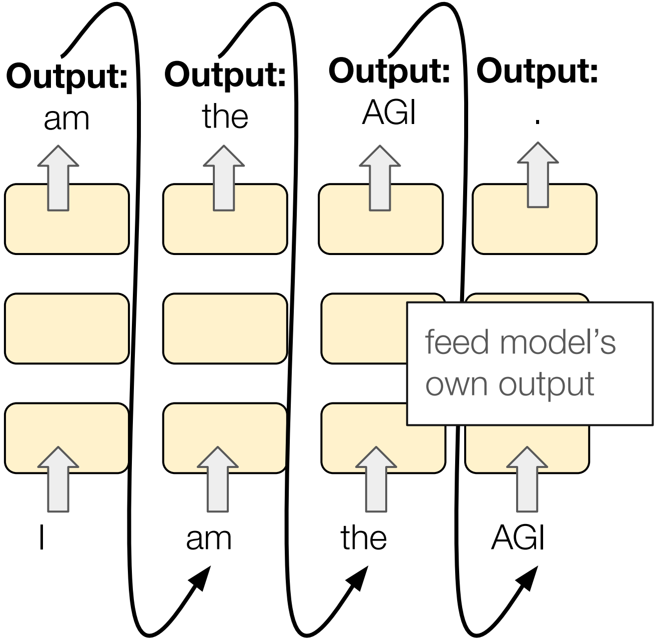
By chain rule of probability, *any* distribution can be represented by next-token prediction (NTP)!

Pr[t_3 | t_1 t_2] ...

Maybe, wrap a verifier/backtracker around the NTP backbone?

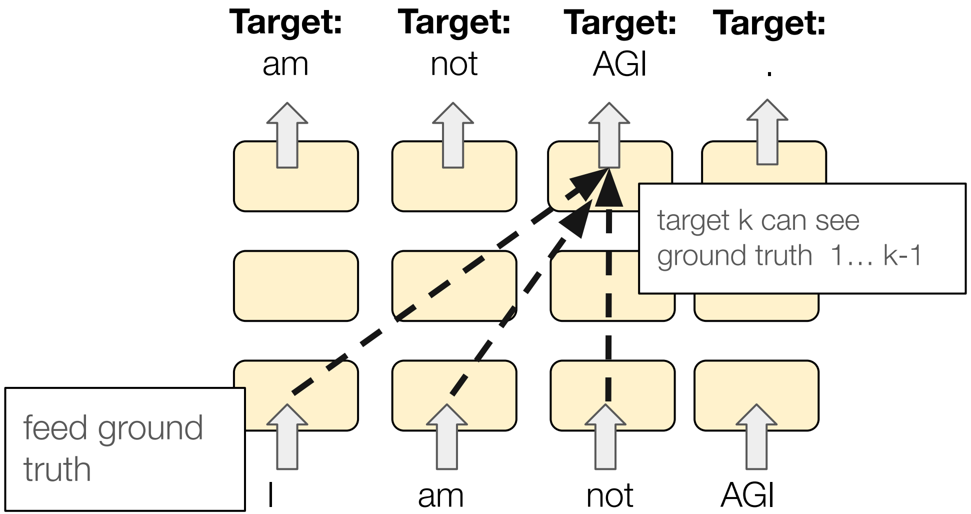
There's a gut feeling that "NTP isn't *the right bias*", but pinning this down seems elusive! What are we missing?

Current NTP debates focus on *representation*.



Inference with autoregression

We need to worry about *learning*!



Training with next-token prediction ("Teacher-forcing")

Sure, (autoregressive) NTP modeling can *represent* any sequence.

But can NTP *learn* any sequence?

Part II: *Failure of NTP learning*

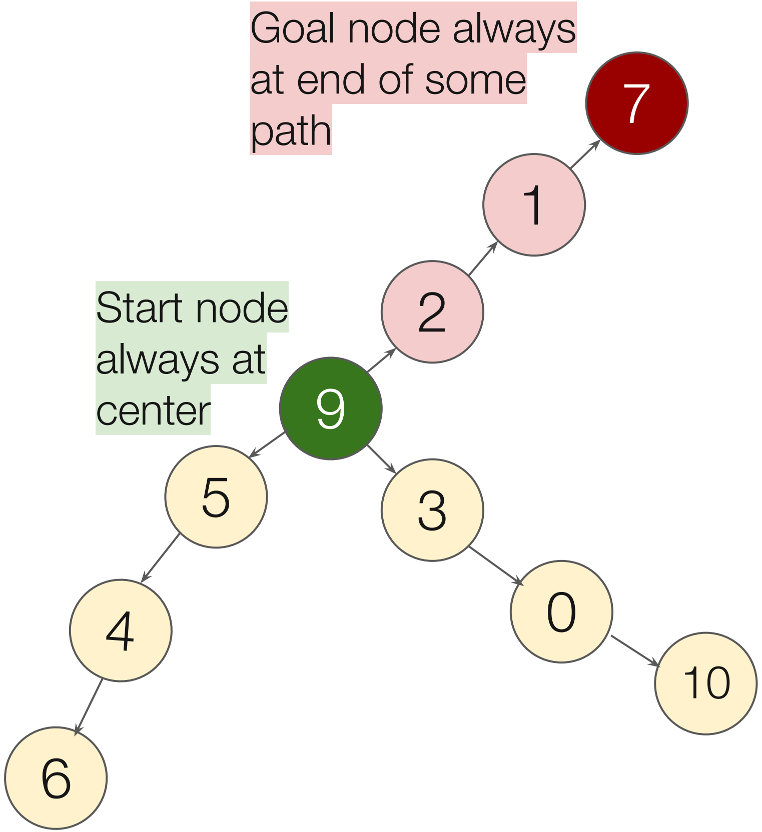
We'll design a **planning** task that is:

1. ***Minimal***
    a. No language understanding required.
    b. No world knowledge required.

2. ***Straightforward***
    a. Intuitively easy to solve
    b. In fact, Transformer/Mamba can solve the task with a slightly different objective!

And **despite** that, training Transformer/Mamba with next-token prediction (empirically) fails to generalize, even **in-distribution**.

A *minimal* task: path-finding on path-star graphs



PROBLEM PREFIX

Randomized adjacency list

2→1, 0→10, 9 → 3, 5 → 4, 4→6, 3→0, 1→7, 9→2, 9→5 ||

find(9 → 7) ?

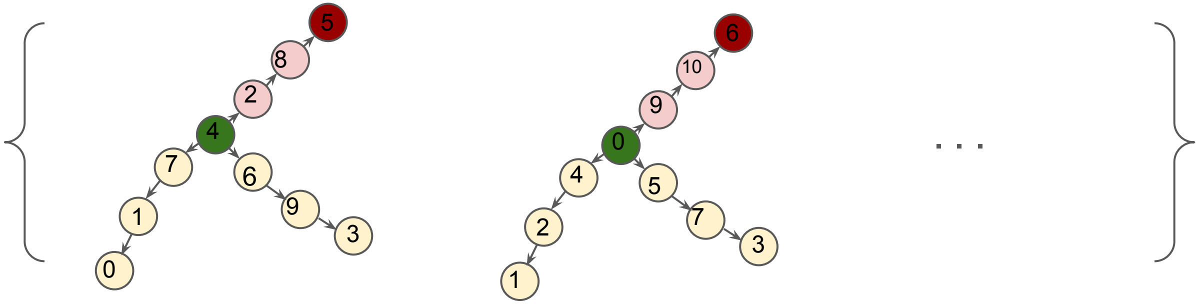
GROUND TRUTH SOLUTION

Start-Goal path 9 →2→1→7

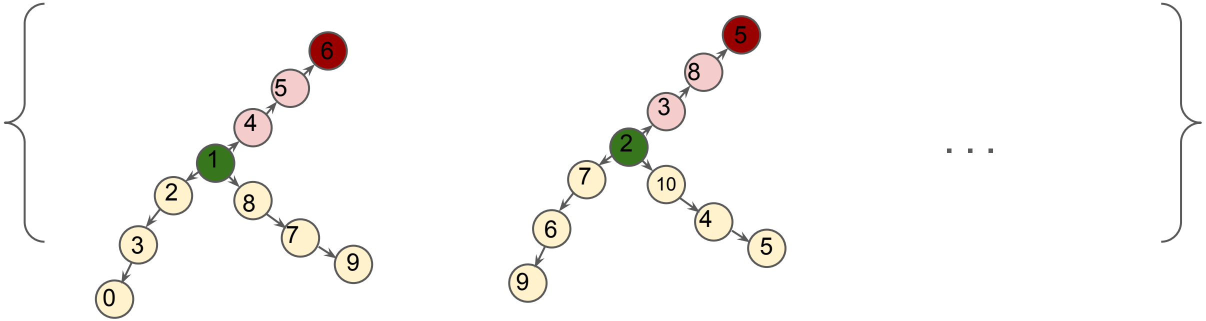
(Each example has same topology, with different randomizations of node IDs and adjacency list orderings.)

Note: Each example has same topology, with different randomizations of node IDs and adjacency list orderings.

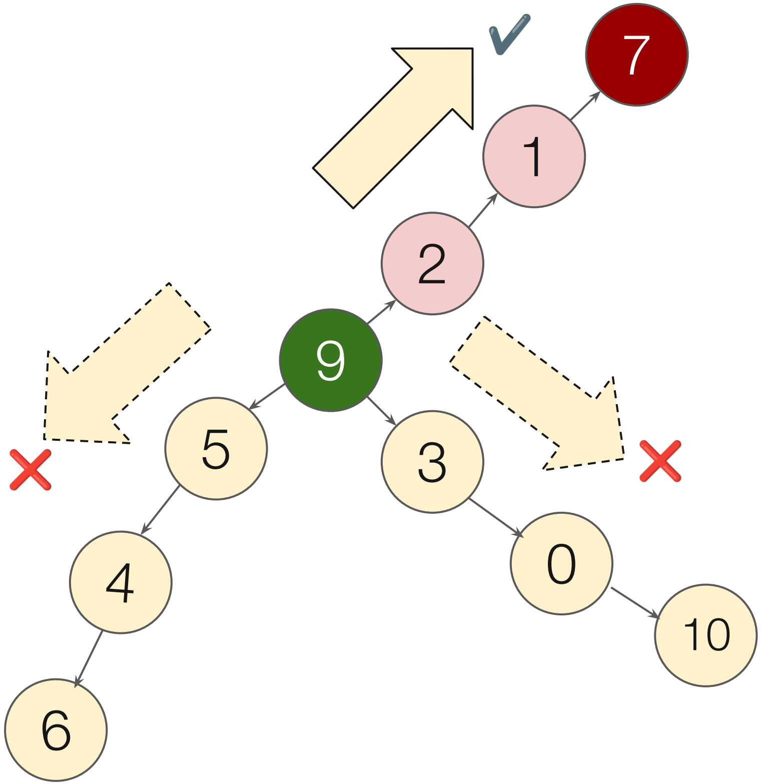
Train data:



Test data:



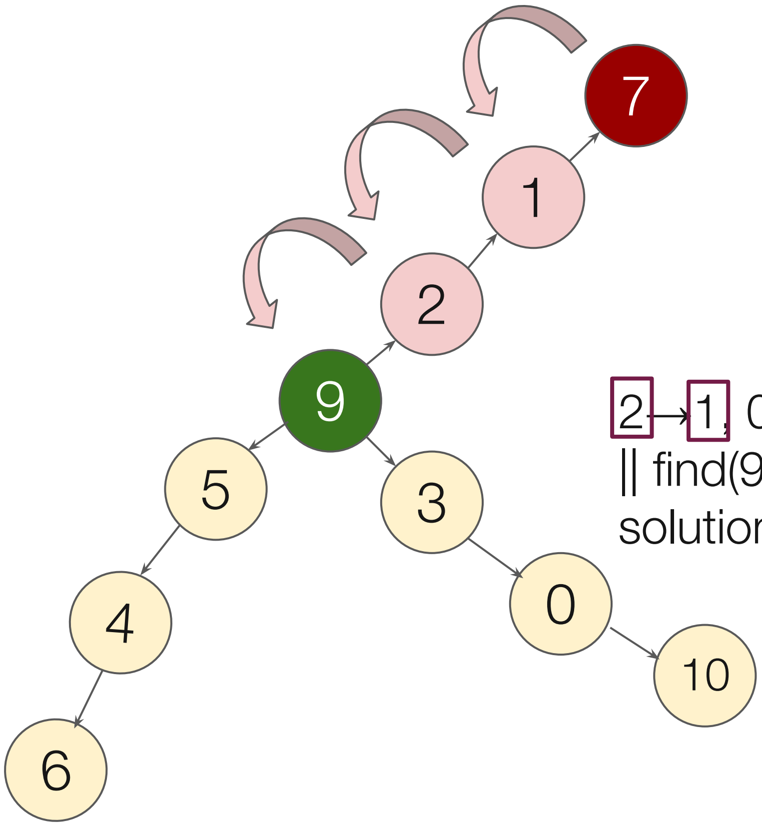
One ideal solution: Plan



Plan:

- examine random path,
- backtrack,
- iterate until goal is found.

Another *straightforward* solution! 💡



Lookahead from right-to-left:

Start from goal and end in start state.
Print reversed path.

$\boxed{2}\to\boxed{1}$, 0→10, 9 → 3, 5 → 4, 4→6, 3→0, $\boxed{1}\to\boxed{7}$, $\boxed{9}\to\boxed{2}$, 9→5
|| find(9 → 7) ||
solution = 9 →2→1→7

Can next-token prediction via teacher-forcing learn either of these mechanisms?

Conceptually and empirically, no.

Can next-token prediction via teacher-forcing learn either of these mechanisms?

Conceptually and empirically, no.

Clever Hans

The horse who did math

by cheating

Image Courtesy: Brittanica

🧑‍🏫 : 2+1 = ?

🐎: *tap*

🧑‍🏫 : 😬

🐎: *tap*

🧑‍🏫 : 😬

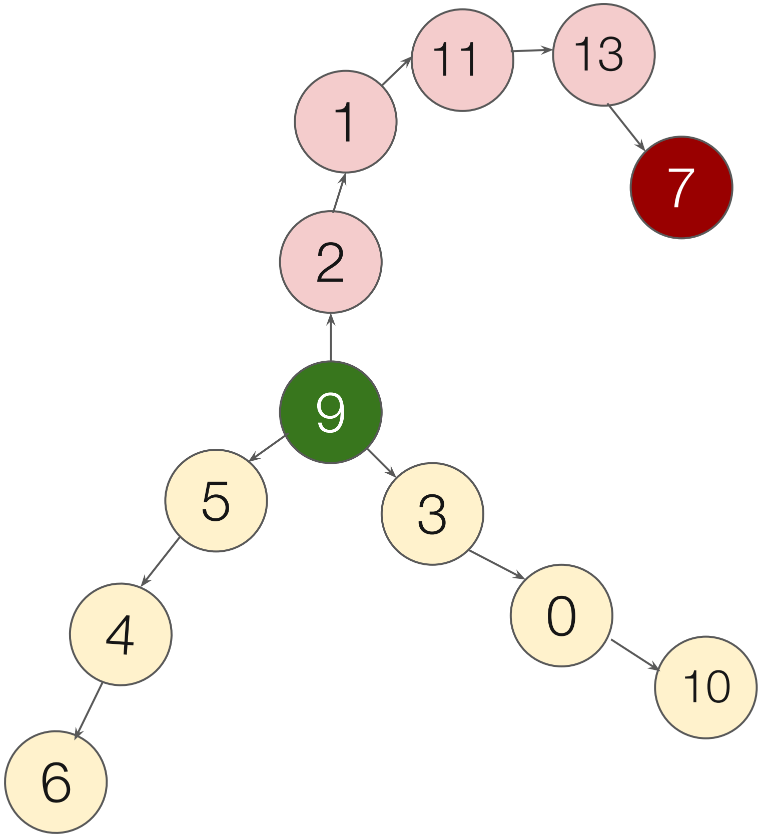
🐎: *tap*

🧑‍🏫 : 🤩

🐎: 🛑

👥👥👥 : 👏👏👏👏

The Clever Hans Cheat



Ideally, learn mapping from *only* the problem ⇒ the solution

plan/lookahead

(adjacency list) + find(9 → 7) ?   9 →2→1→11→13→7

But, in teacher-forcing, learn mapping from problem **+ solution prefixes** ⇒ the solution

….. 9 →2→1→11→13→?   7

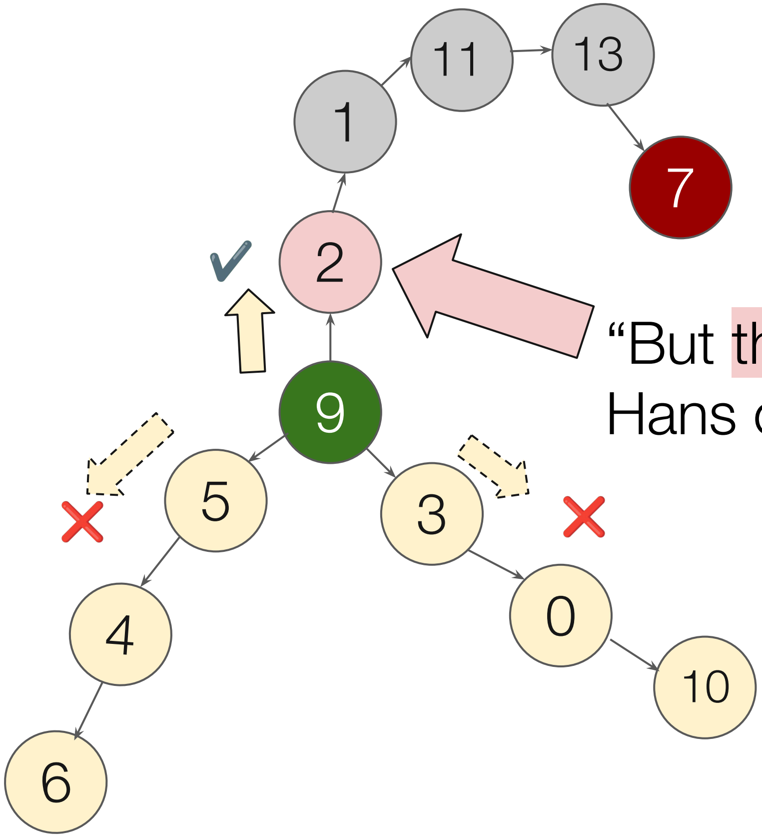
….. 9 →2→1→11→?   13

….. 9 →2→1→?   11

….. 9 →2→?   1

Clever Hans Cheat

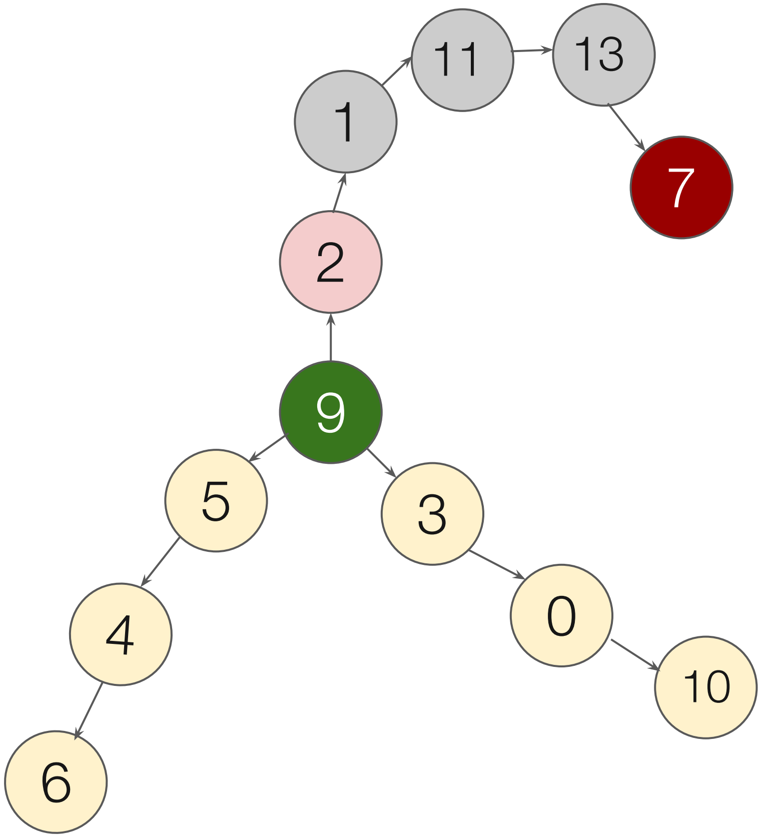
: Just predict adjacent vertex **w/o any plan/lookahead!**



"But the first token is *not* fit using Clever Hans cheat!

Model still has incentive to learn the true mechanism?"

The Indecipherable Token



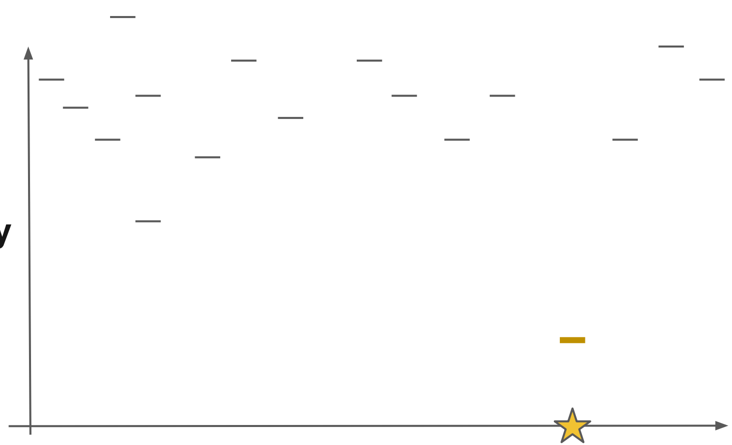
adjacency list|| find(9 → 7) ||
solution = 9 →2 →1→11→13→7

"Intermediate" supervision lost to Clever Hans cheat

Can model infer the mechanism to generate node 2 **without the remaining supervision?**

The Indecipherable Token

Loss **on only 1st token**



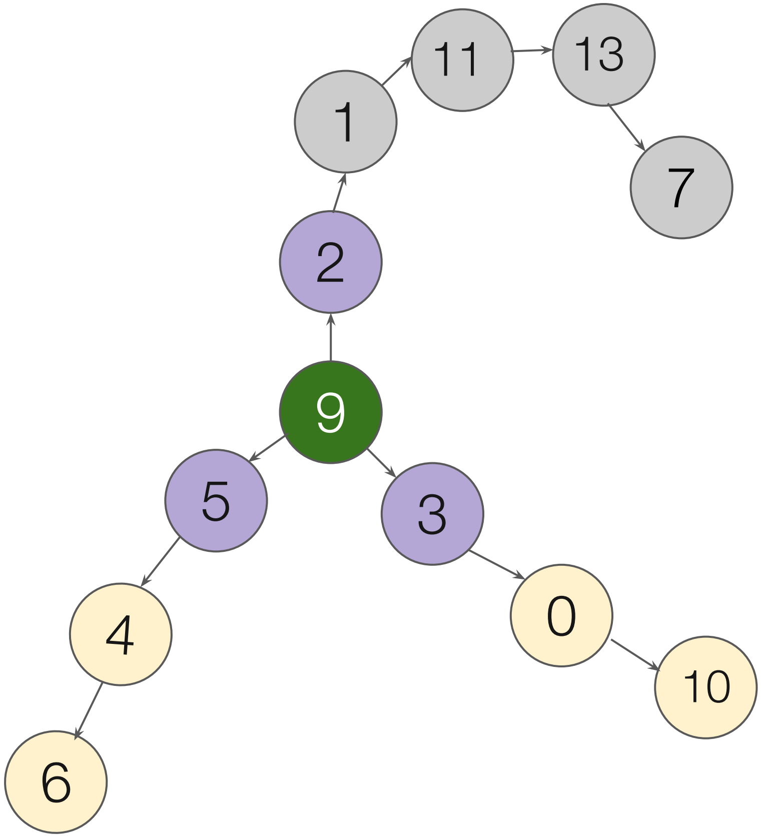
Exponential space of algorithms
($num\_subroutines^{num\_steps}$):

adjacency list|| find(9 → 7) ||
solution = 9 →2 →1→11→13→7

“Intermediate” supervision lost to Clever Hans cheat

Can model infer the mechanism to generate node 2 **without the remaining supervision?**

A very, very hard “needle-in-the-haystack” optimization problem.



Clever Hans cheat
(during training)

+

The Indecipherable Token
(during training)

=

***In*-distribution failure**
(during inference)

Can next-token prediction via teacher-forcing learn the path-star task?

Conceptually and empirically, no.

Part III: *Experiments*

Train transformers on these graphs

1. Start with GPT-style Transformers trained from scratch

2. Finetune pre-trained GPT-2 models

3. Are Transformers the problem? → Try recurrent-style models like Mamba

Recap: How do we train?

1. Given: adj. list

   → Predict **first** node

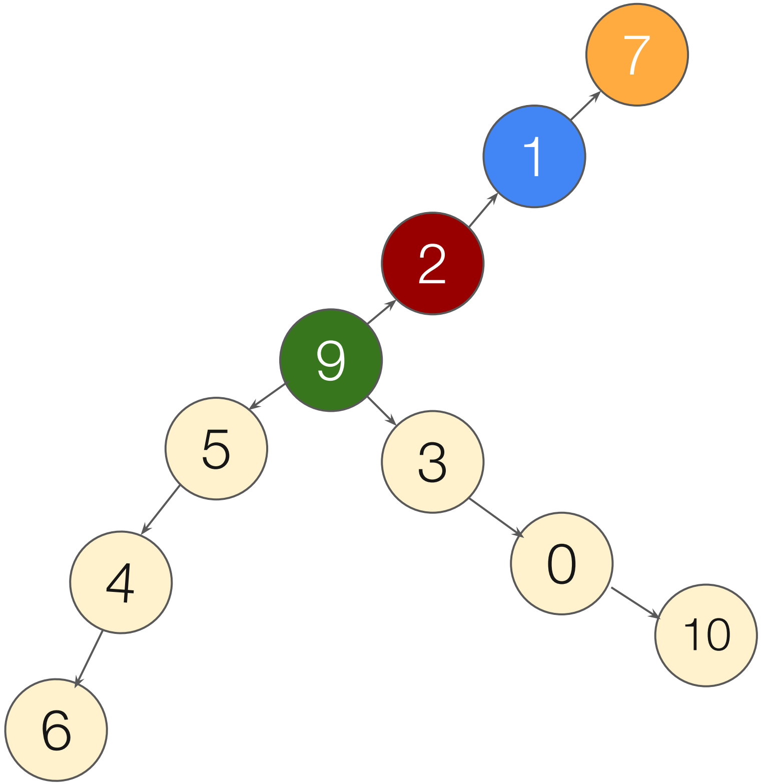
2. Given: adj. list + **first** node

   → Predict **second** node

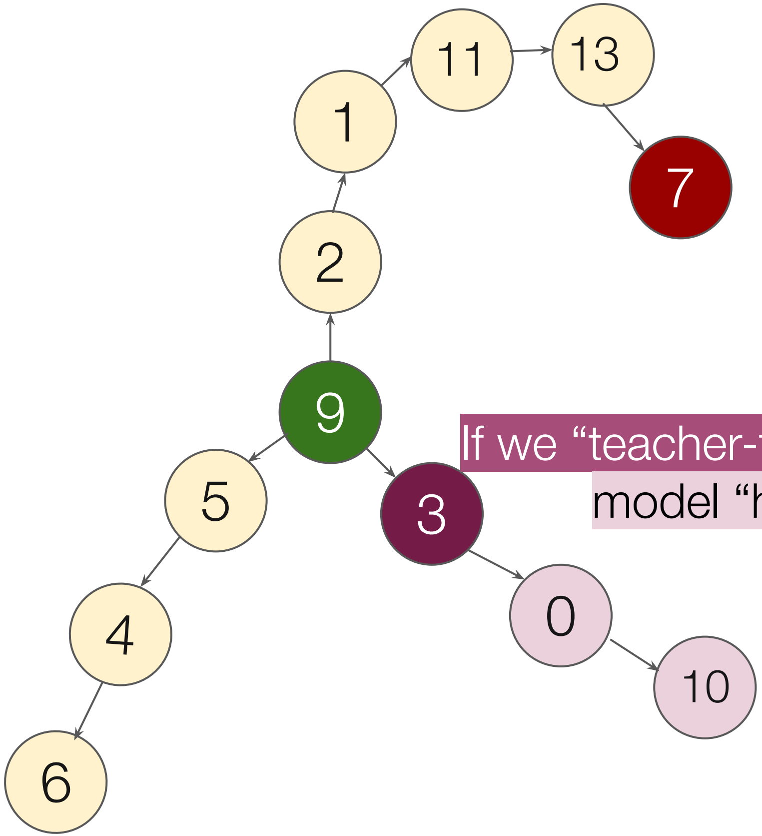
3. Given adj. list + **first** + **second** node

   → Predict **third** node

etc.



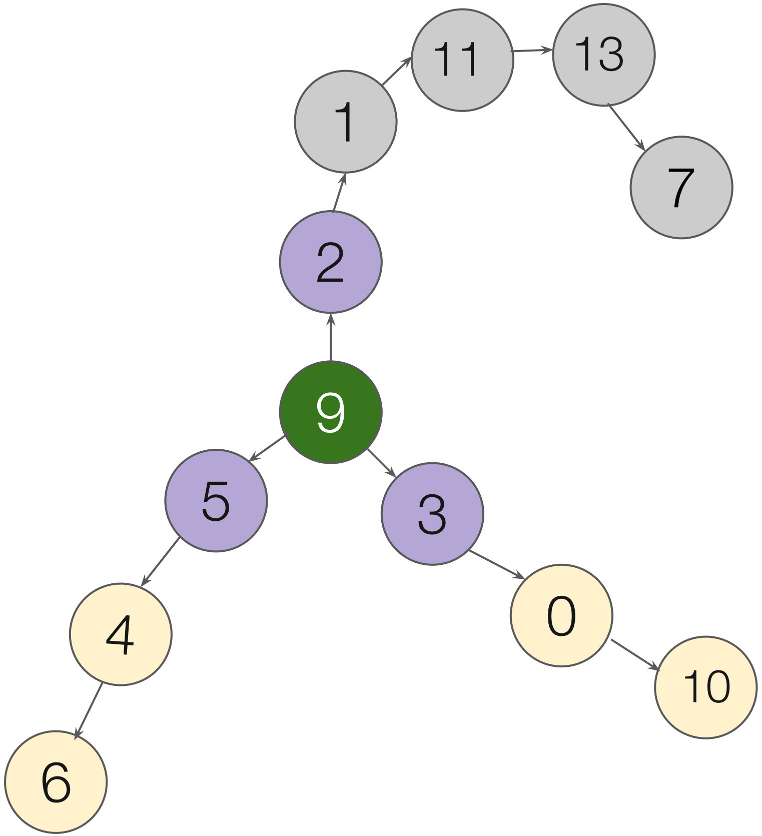
Verifying the Clever Hans cheat empirically



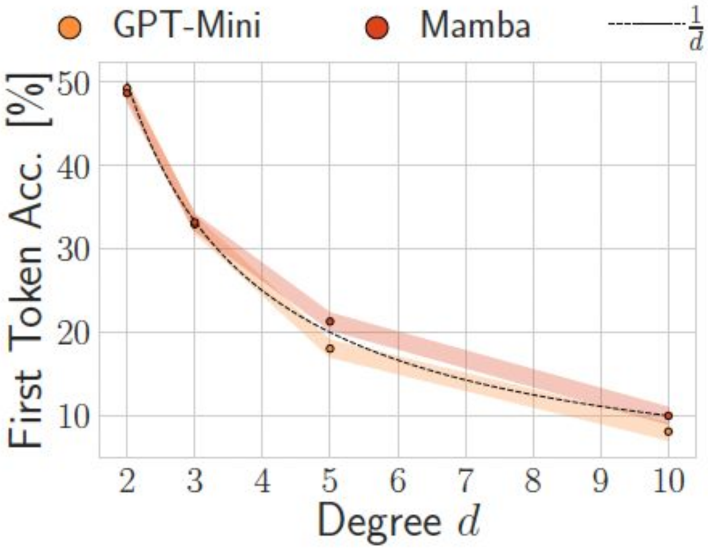
If we "teacher-force" on a wrong node,
model "hallucinates" the next-vertex path with ~100% prob

$G_{\text{degree,length}}$:	$G_{2,5}$	$G_{2,20}$	$G_{5,5}$	$G_{10,5}$
GPT-MINI	99.7	100	100	99.8
GPT2-LARGE	99.8	99.7	100	99.8
MAMBA	97.6	98.3	99.5	95.9

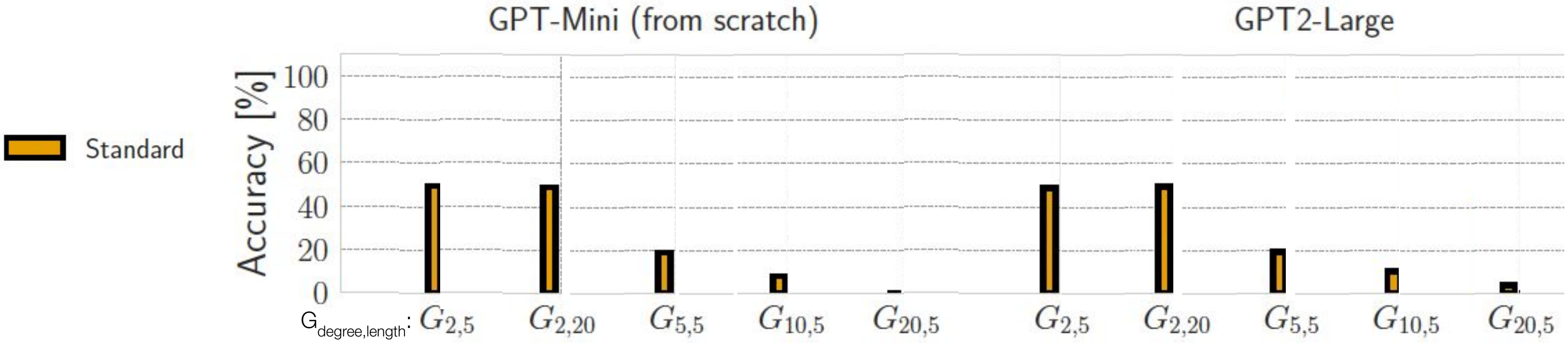
Verifying the Indecipherable Token empirically



Model just learns to output
a random legal first move,
even after 500 epochs on 200k examples.

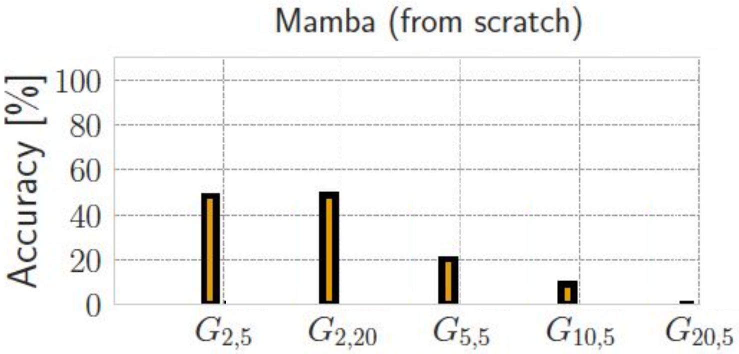


In-distribution Failure



Trivial accuracy for every topology:

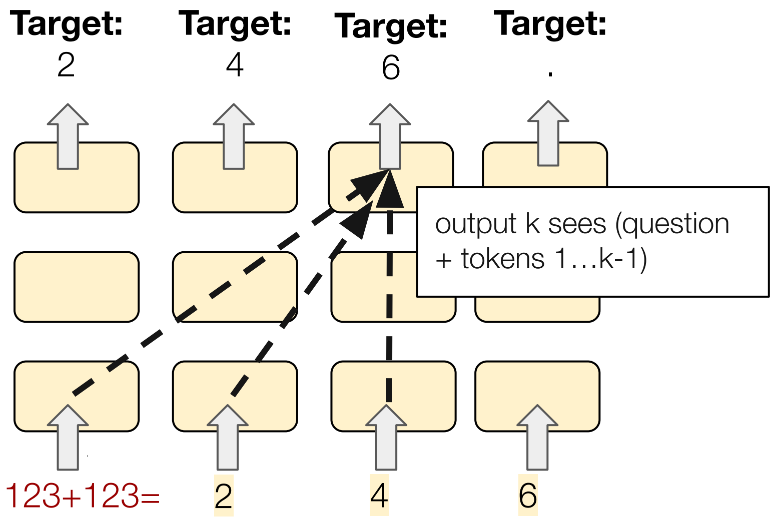
for topology $G_{\text{degree, length}}$ (X axis)
Accuracy (Y axis) = 1/degree



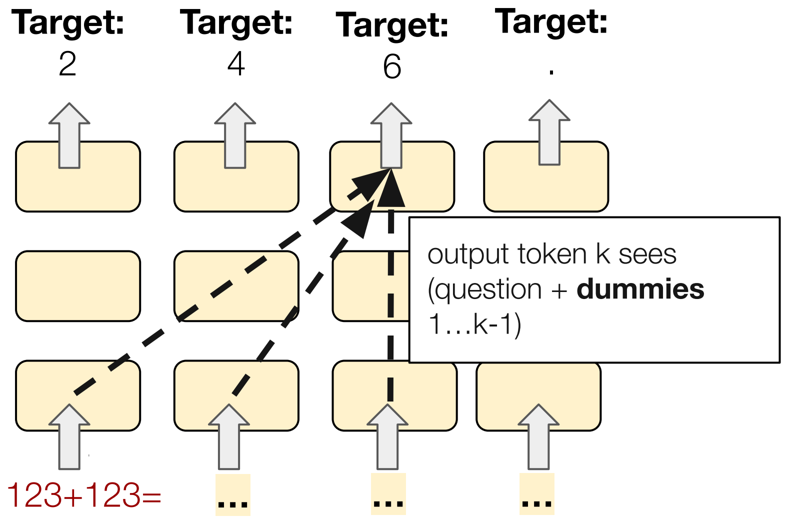
Part IV: *A multi-token prediction fix?*

Idea: *Teacherless* training

Also see PaSS, Monea et al., 2023

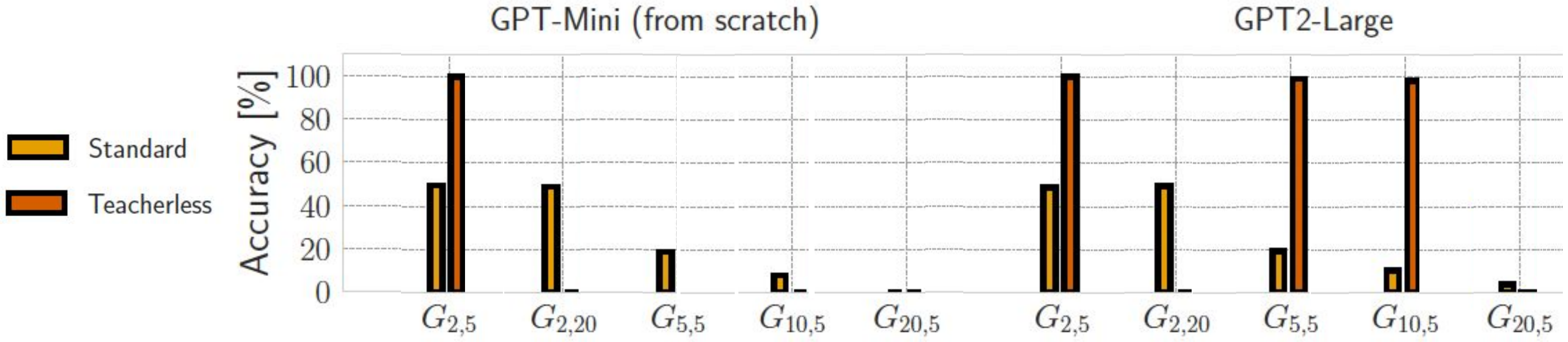


Standard NTP training
a.k.a teacher-forcing

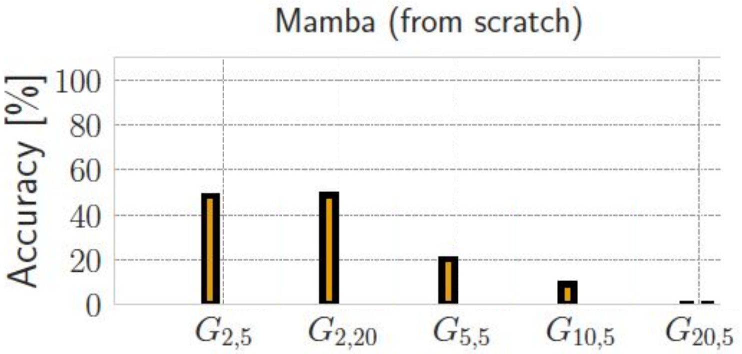


Teacherless training: Replace input-side answer w/ dummies ⇒ enforces multi-token-predicting the answer.

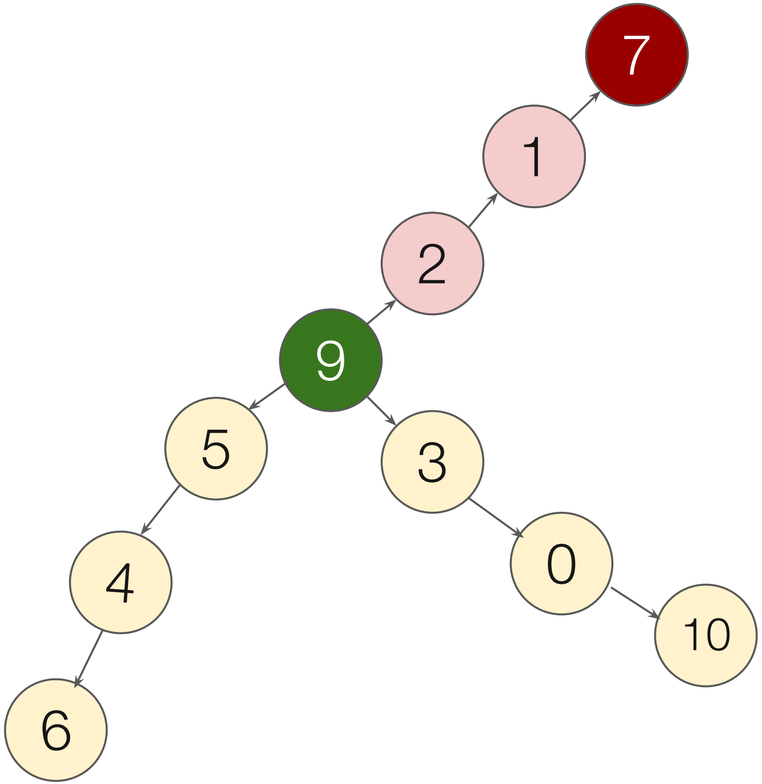
Idea: *Teacherless* training



Standard: random performance
Teacherless: fits both train & test [or neither]



Sidenote: Training with reversed targets



PROBLEM PREFIX

Randomized adjacency list

2→1, 0→10, 9 → 3, 5 → 4, 4→6, 3→0, 1→7, 9→2, 9→5 ||
find(9 → 7) ?

GROUND TRUTH SOLUTION

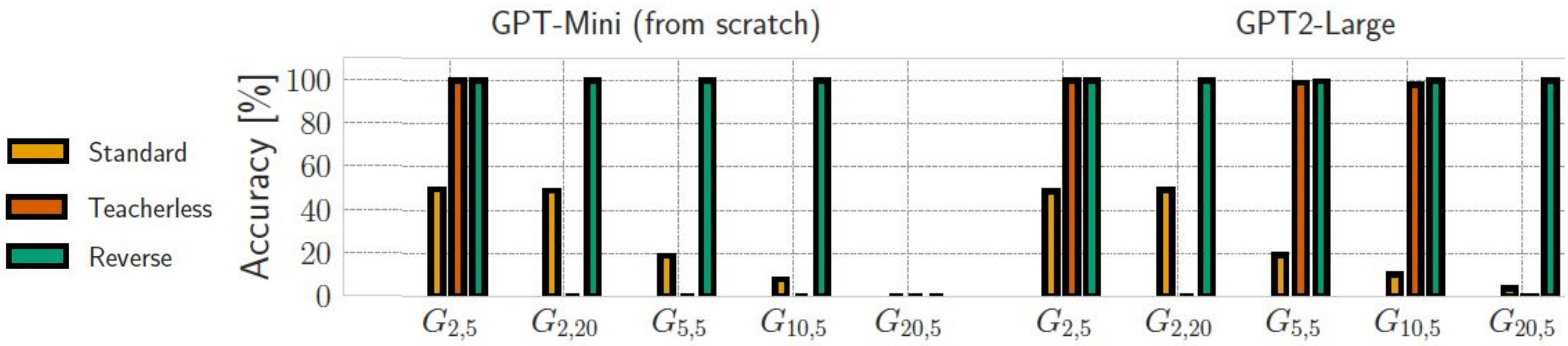
Start-Goal path 9 →2→1→7

REVERSED SOLUTION

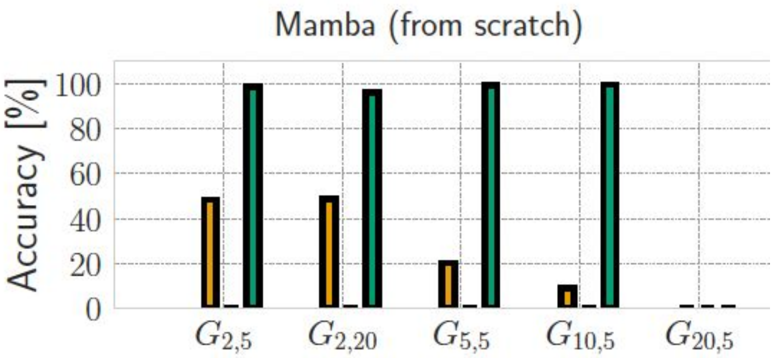
Goal-Start path 7←1←2←9

(BTW: here, reversed target can be treated as CoT.)

Sidenote: Training with reversed targets

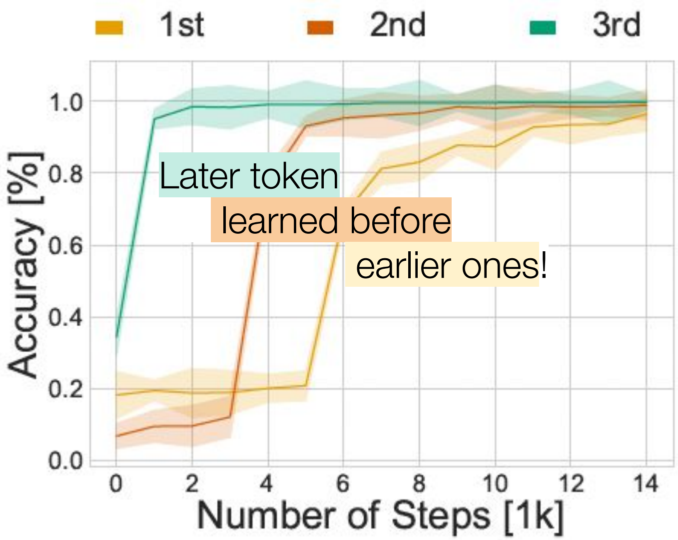


Standard: random performance
Teacherless: fits both train & test [or neither]
Reversed: perfect accuracy!



Reversing the tokens easily solves the problem right-to-left!

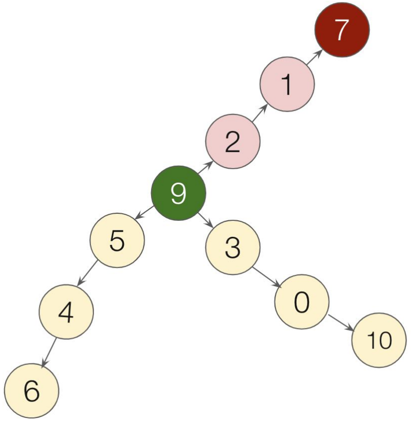
Teacherless training too allows the model to *implicitly* view the problem right-to-left.

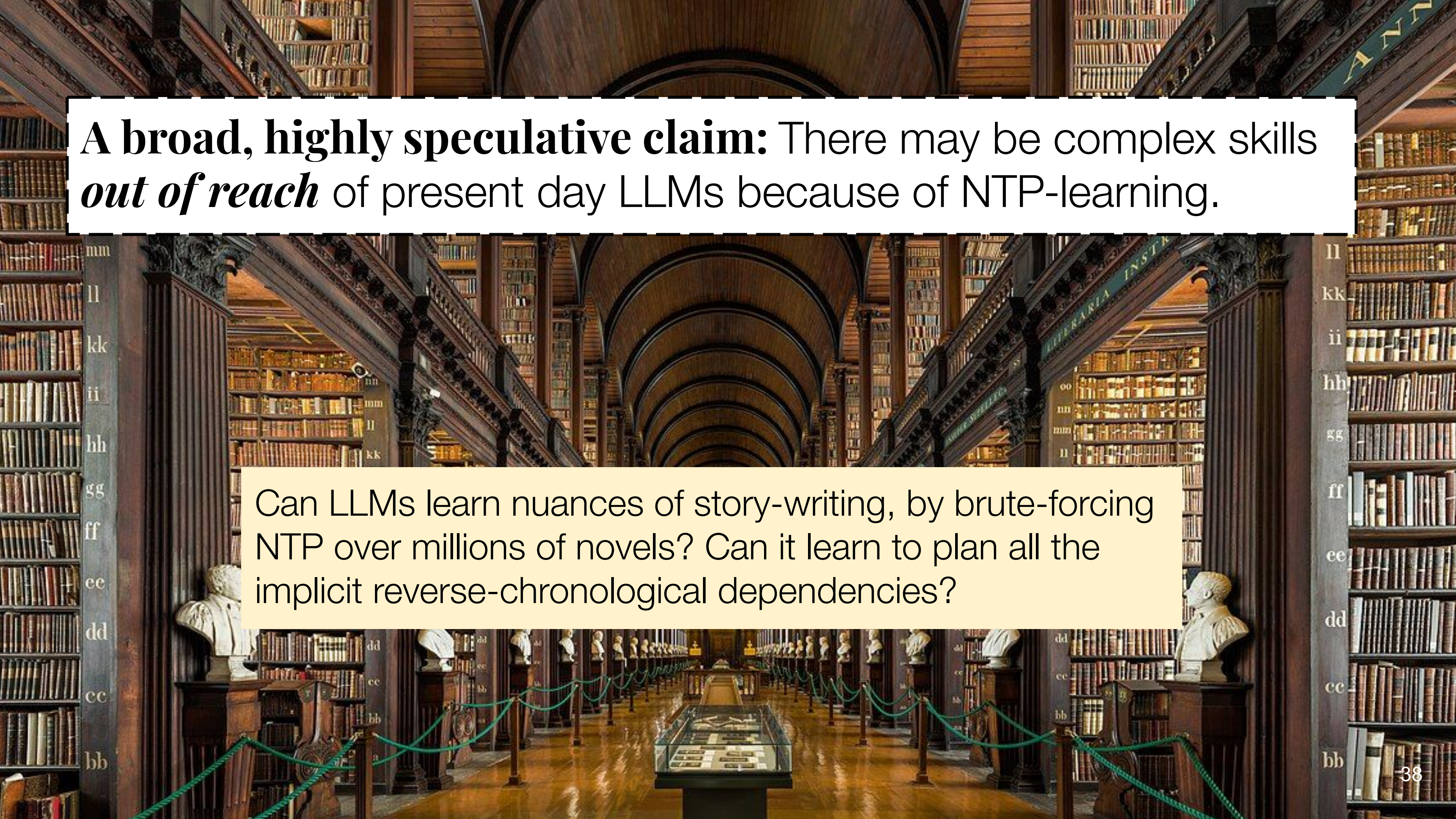


The task *is* easy to learn with given supervision, but remarkably, left-right NTP learning fails.

So what?

Precise claim: NTP-learning from-scratch fails even in this minimal task
(and this isn't due to other factors like the architecture, or autoregression etc.,).



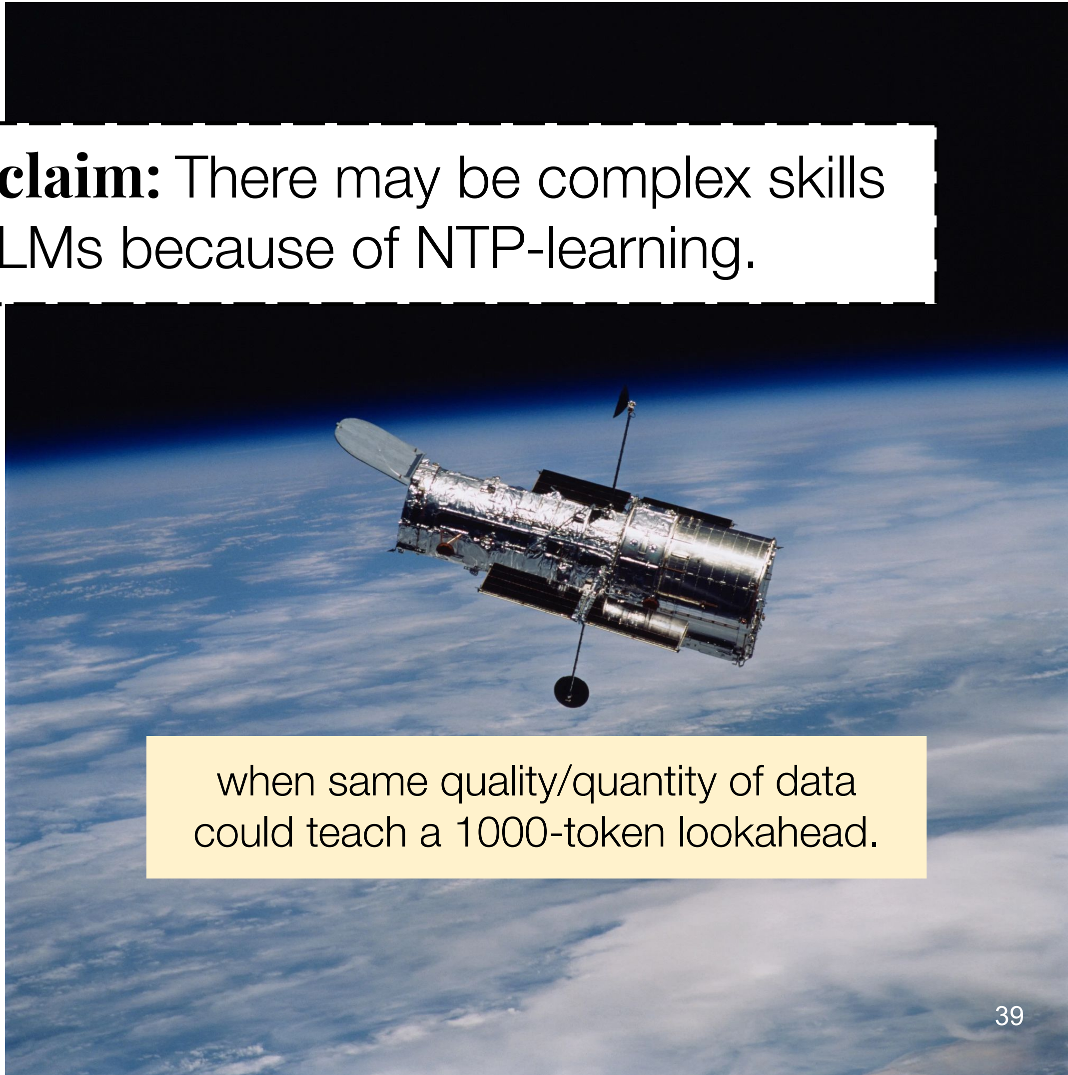


A broad, highly speculative claim: There may be complex skills ***out of reach*** of present day LLMs because of NTP-learning.

Can LLMs learn nuances of story-writing, by brute-forcing NTP over millions of novels? Can it learn to plan all the implicit reverse-chronological dependencies?

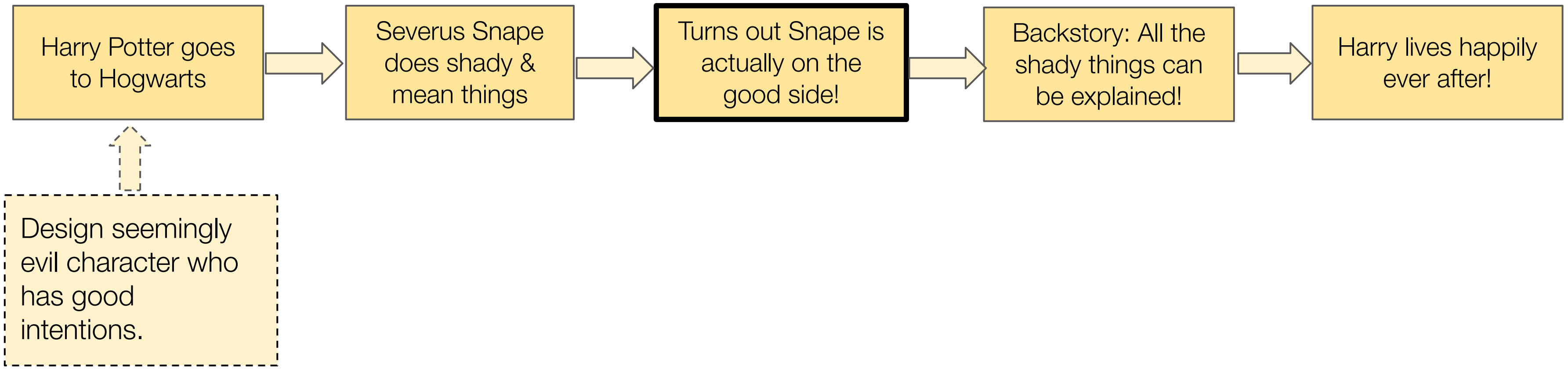
A broad, highly speculative claim: There may be complex skills ***out of reach*** of present day LLMs because of NTP-learning.

Perhaps, models learn to plan only 25 tokens ahead,



when same quality/quantity of data could teach a 1000-token lookahead.

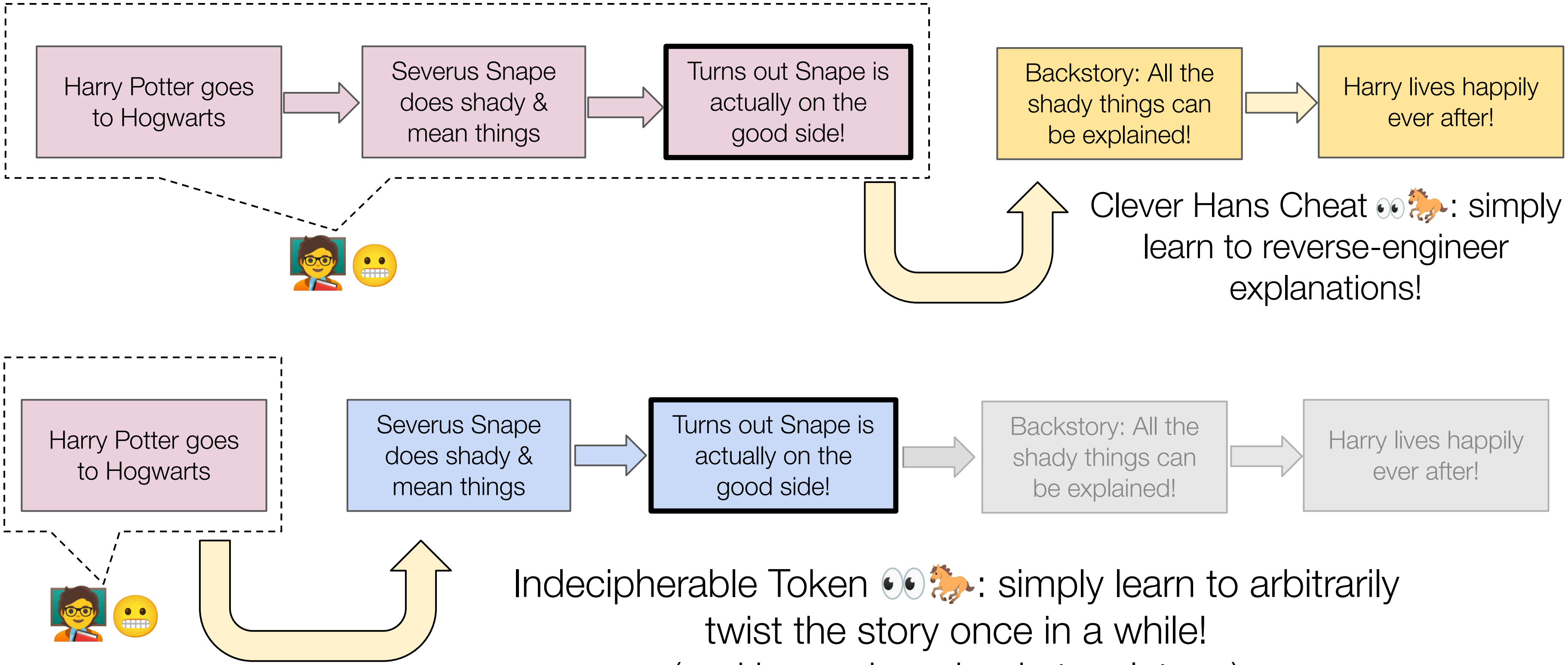
Can learning to predict the next-token on a million novels, learn story-writing?



Ideally, learn to think of plot twists in advance!

But…

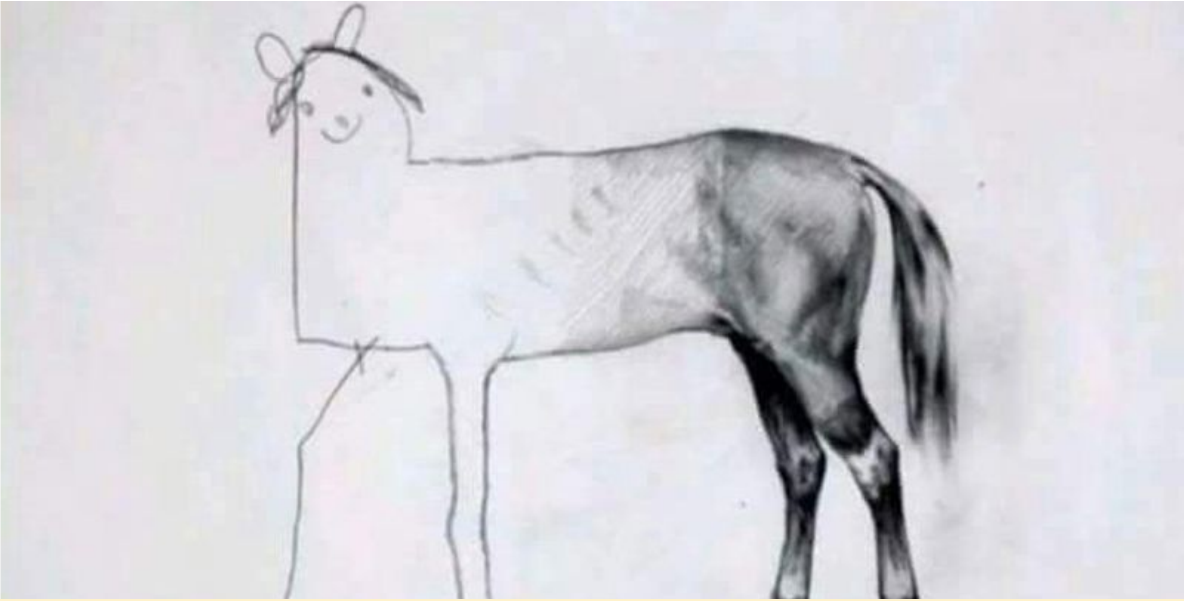
Can learning to predict the next-token on a million novels, learn story-writing?



Clever Hans Cheat 👀🐎: simply learn to reverse-engineer explanations!

Indecipherable Token 👀🐎: simply learn to arbitrarily twist the story once in a while!
(and improvize a backstory later...)

A broad, but more agreeable claim: The NTP-based pretraining paradigm highly under-utilizes signals from the data.



Later tokens well-fit using trivial mechanisms, while earlier tokens become harder to learn.

Sure, (autoregressive) NTP modeling can *represent* any sequence.

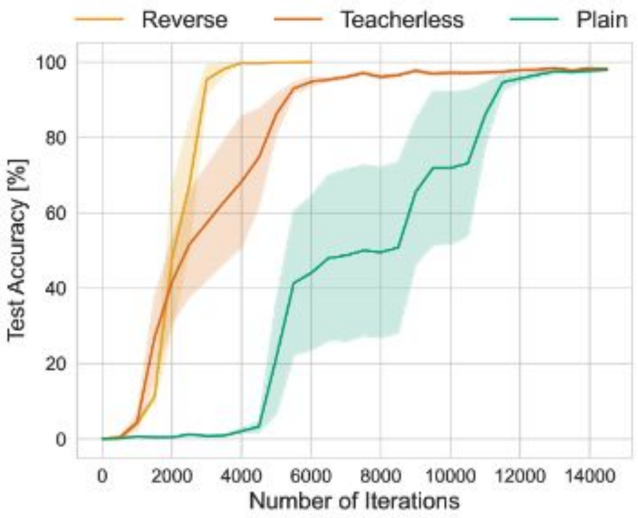
But can NTP *learn* any sequence?

Many exciting open questions!

1. Would multitoken training help in more general problems? What's the right way to optimize it?

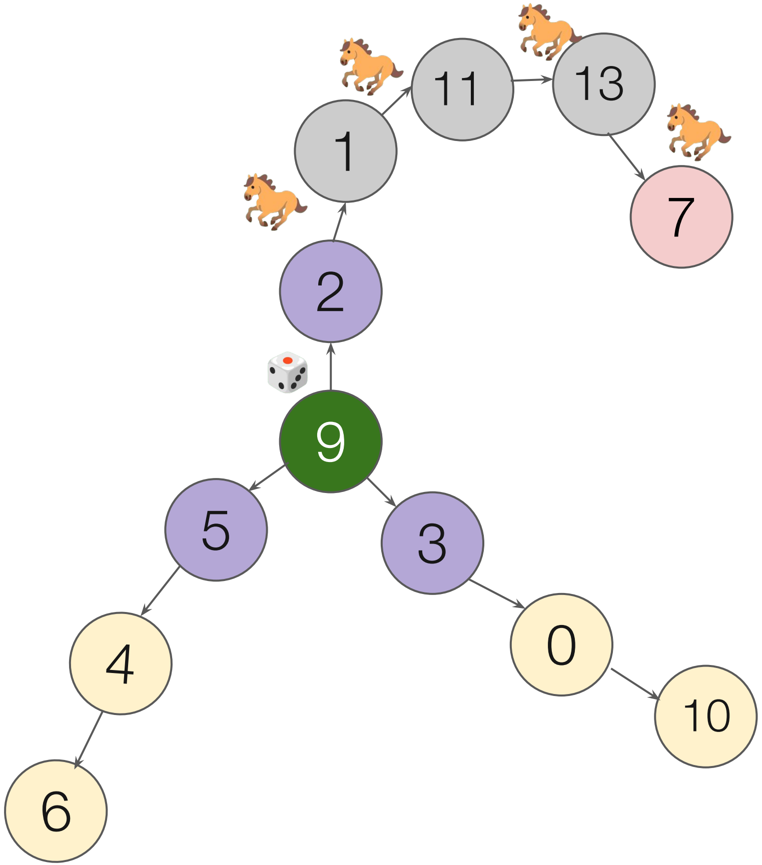
2. Should we pretrain with CoT supervision? How is it even possible for say, story-writing?

3. Lots of open formal questions:
    a. What can NTP+gradient descent (not) learn?
    b. What does multi-token loss surface look like?
    c. …



Multitoken (teacherless) training improves data-efficiency of addition task.

Thank you! Questions?



P.S.: Important disclaimer published after our work: 👇

The Guardian

US

News | Opinion | Sport | Culture | Lifestyle

US US elections 2024 World Environment Ukraine Paraly

Animal behaviour

Horses can plan ahead and think strategically, scientists find

Team hopes findings will help improve equine welfare after showing cognitive abilities include being 'goal-directed'

Donna Ferguson

Sun 11 Aug 2024 19.01 EDT

References

1. Dziri et al. *Faith and fate: Limits of transformers on compositionality.* NeurIPS 2024.
2. Kääriäinen, *Lower bounds for reductions,* 2006
3. Ross & Bagnell, *Efficient Reductions for Imitation Learning,* 2010