THE PITFALLS OF NEXT-TOKEN PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Can a mere next-token predictor faithfully model human intelligence? Our work is aimed at crystallizing this intuitive concern, which is currently fragmented in the literature. As a starting point, we advocate isolating the two phases of next-token prediction that are often conflated: autoregression during inference vs. teacher-forcing during training. We argue that the previously-identified problem of "exponential error accumulation" is a symptom of autoregressive inference. We then identify a more concerning problem: teacher-forcing can let the model fit the training data by *cheating*, causing total in-distribution failure during inference. We design a minimal planning task where empirically both the Transformer and the Mamba architecture fail in this manner — remarkably, despite the task being easy to learn. Our work consolidates these and other essential arguments surrounding next-token prediction. We hope our effort can ground the next-token prediction debate and inspire further explorations beyond this paradigm.

1 INTRODUCTION

Long after its inception in the seminal work of Shannon (1948; 1951), next-token prediction has made its way into becoming a core part of the modern language model. But despite its long list of achievements, there is a small but growing belief that a next-token predicting model is merely an impressive *improv* artist that cannot truly model human thought. Humans, when navigating the world, meticulously imagine, curate and backtrack plans in their heads before executing them. Such strategies are unfortunately not explicitly built into the present-day language model. This form of criticism has been floating around as an informal viewpoint barring a few concrete observations of failure scattered in the literature (Dziri et al., 2023; LeCun, 2024; Bubeck et al., 2023). Our paper is aimed at crystallizing this intuitive criticism of next-token prediction, and developing the core arguments of this debate.

We start by making more precise, what it means to say that human-generated language, or problem-solving, does not follow next-token prediction. When formalizing this, we hit an immediate roadblock: isn't every sequence generation task possible autoregressively? Put differently, an optimist would say, every distribution over a sequence of tokens can be captured by an appropriately sophisticated next-token predictor simulating the chain rule of probability i.e., $\mathbb{P}(r_1, r_2, \ldots) = \prod_i \mathbb{P}(r_i | r_1 \ldots r_{i-1})$. Thus, the autoregressivity in our systems is not antithetical to learning human language, after all.

However, a pessimist would worry, even with minor imperfections in the next-token predictor, the accuracy may break down spectacularly for long sequences. Say, even if the per-token error is as little as 0.01, the probability of seeing an erroneous token exponentially accumulates along the way, and by the end of 200 tokens, blows up to 0.86 (LeCun, 2024; Dziri et al., 2023).

We point out that this critique is still limited as it does not clearly distinguish the two phases at which next-token prediction is involved: one during autoregressive inference, and the other during training via *teacher-forcing* (Williams & Zipser, 1989) i.e., where we teach the model to predict token-by-token, offering access to all previous ground truth tokens. The aforementioned error accumulation solely corresponds to performing inference autoregressively. Thus, it could inspire solutions that are post-hoc e.g., look for erroneous tokens and explicitly backtrack. However, training via teacher-forcing can potentially induce more severe pathologies into the backbone model. Consequently, such pathologies cannot be cured by simple post-hoc approaches.

In particular, we argue that, in tasks that involve planning, teacher-forcing provides supervision that is both *incorrect* and *impossible* to learn from. First, teacher-forcing can induce problematic,

cheating-based shortcuts that use the revealed prefix of the ground truth answer to fit future answer tokens. We call this the *Clever Hans cheat*, named after a famous arithmetic-solving horse that was debunked to have been following subtle cues from a trainer. Now, while the later tokens become easy to fit by the Clever Hans cheat, in contrast, the earlier answer tokens become impossible to learn. This is because they no longer come with any supervision about the full answer — part of the supervision is lost to the Clever Hans cheat. These two flaws together would prevent the model from recovering the underlying plan, and consequently, from even generalizing to in-distribution sequences.

Empirically, we demonstrate this failure in a minimal graph path finding setup. The setup captures the core challenges of problems that require lookahead/planning, but at the same time, is so straightforward to solve that the failure of any model is remarkable. We observe failure for both the Transformer (Vaswani et al., 2017) and the Mamba architecture, a structured state space model (Gu & Dao, 2023) verifying that the issues we identify are indeed inherent to next-token prediction, and are not an artifact of architectural choices like convolution or recurrence.

We summarize our contributions below.

1. We consolidate existing critiques against next-token prediction and crystallize other possible points of contention (§3 and §6).
2. We identify that the next-token prediction debate must not conflate next-token prediction involved in autoregressive inference and that of teacher-forcing. Both lead to vastly different issues, with different solution strategies (§3,§A).
3. We conceptually argue that in planning-based tasks, teacher-forcing can provide flawed supervision, severely detrimental to even in-distribution performance (§4).
4. We design a minimal planning task (§4.1). where we demonstrate the above failure for the Transformer and Mamba architectures, despite the task being easy to learn (§5).
5. We identify a promising "teacherless" form of training to (sometime) circumvent these failures (§5, Eq 6).

We hope that our work can place future debates on next-token prediction on solid ground. We also hope that our contributions — such as the minimal example of failure and the teacherless training — can help explore alternative paradigms of training.

2 The Two Modes of Next-Token Prediction

Consider a set of tokens $\mathcal{V}$. Let $\mathcal{D}$ be a ground truth distribution over sequences that consist of a prefix $\boldsymbol{p}$ and a response $\boldsymbol{r}$, denoted as $\boldsymbol{s} = \boldsymbol{p}, \boldsymbol{r}$ where $\boldsymbol{p} = (p_1, p_2, \ldots,) \in \mathcal{V}^{L_{\text{pref}}}$ and $\boldsymbol{r} = (r_1, r_2, \ldots) \in \mathcal{V}^{L_{\text{resp}}}$. We assume sequences of fixed length merely for simplicity.

For any sequence $\boldsymbol{s}$, let $\boldsymbol{s}_{<i}$ denote the first $i-1$ tokens of $\boldsymbol{s}$, and $\boldsymbol{s}_{i<}$ the tokens following the ith token. Note that $\boldsymbol{s}_{<1}$ is the empty prefix. With an abuse of notation, let $\mathbb{P}_{\mathcal{D}}(s_i|\boldsymbol{s}_{<i})$ denote the ground truth probability mass on s_i being the ith token given the prefix $\boldsymbol{s}_{<i}$. Consider a next-token-predicting language model $\texttt{LM}_\theta$ (with parameters θ) such that $\texttt{LM}_\theta(\hat{s}_i = s_i; \boldsymbol{s}_{<i})$ is the probability that the model assigns to the ith output $\hat{s}_i$ taking the value s_i, given as *input* the sequence $\boldsymbol{s}_{<i}$. Note that the next-token predictor only defines the probability for a single future token given an input, but not the joint probability of multiple future tokens. This joint probability is axiomatically defined analagous to the chain rule of probability. Letting $\hat{\boldsymbol{r}} = \boldsymbol{r}$ denote an exact token-by-token match, we define:

$$\texttt{LM}_\theta(\hat{\boldsymbol{r}} = \boldsymbol{r}\ ; \boldsymbol{p}) \coloneqq \prod_{i=1}^{L_{\text{resp}}} \texttt{LM}_\theta\left(\hat{r}_i = r_i; \boldsymbol{p}, \boldsymbol{r}_{<i}\right) \tag{1}$$

To model the above probability, two distinct types of next-token prediction are used. First, during inference, for a given prefix, we autoregressively sample from the model token-by-token, providing as input the prefix and all previously-generated tokens. Formally,

Definition 1. ***(Inference-time next-token prediction via autoregression)*** *Autoregressive inference is a form of inference-time next-token prediction in that to generate a response $\hat{\boldsymbol{r}}$, we iterate over i, to sample the next token $\hat{r}_i$ with the distribution $\texttt{LM}_\theta(\hat{r}_i\ ; \boldsymbol{p}, \hat{\boldsymbol{r}}_{<i})$. We denote this as $\hat{\boldsymbol{r}} \overset{\text{ag}}{\sim} \texttt{LM}_\theta(\cdot\ ; \boldsymbol{p})$.*

There is another type of next-token prediction, one that is applied during the training process, called *teacher-forcing*. Here, instead of feeding the model its own output back as input, the model is fed with prefixes of the *ground truth* response $\boldsymbol{r}_{<i}$. Meanwhile, the model is assigned as supervisory target, r_i, the next ground truth token. Then, the model maximizes a sum of next-token log-probabilities:

Definition 2. ***(Training-time next-token prediction via teacher-forcing)*** *Teacher-forced training is a form of training-time next-token prediction in that we find parameters θ that maximize:*

$$\mathcal{J}_{\texttt{next-token}}(\theta) = \mathbb{E}_{(\boldsymbol{p},\boldsymbol{r})\sim\mathcal{D}}\left[\log \texttt{LM}_\theta\left(\hat{\boldsymbol{r}} = \boldsymbol{r}\ ; \boldsymbol{p}\right)\right] = \mathbb{E}_{\mathcal{D}}\Big[\sum_{i=1}^{L_{resp}} \log \texttt{LM}_\theta\left(\hat{r}_i = r_i; \boldsymbol{p}, \boldsymbol{r}_{<i}\right)\Big] \quad (2)$$

The key aspect here is that we extract the model's output, *allowing the model access to the ground truth response preceding the current token*. However, as noted in Goyal et al. (2016); Bengio et al. (2015), this creates a discrepancy during inference, when the model is only fed its own outputs rather than the ground truth. This discrepancy is core to the failure that we will eventually describe in §4.

3 Introductory Arguments

A broad criticism against next-token predictors is that these models are not designed to explicitly plan ahead (LeCun, 2024; Bubeck et al., 2023; Dziri et al., 2023). However, this discourse is fragmented, and does not distinguish between the two modes of next-token prediction. Our goal is to analyze the existing intuition more systematically, and also consolidate other possible points of contention.

The chain-rule-of-probability defense: The most tempting defense for next-token prediction is that the chain rule of probability always promises us a next-token predictor that can fit our distribution:

Fact 1. ***(Every sequence distribution can be represented by a next-token predictor)*** *By the chain rule of probability we have $\mathbb{P}_{\mathcal{D}}(\boldsymbol{r} \mid \boldsymbol{p}) = \prod_{i=1}^{L_{resp}} \mathbb{P}_{\mathcal{D}}(\boldsymbol{r} \mid \boldsymbol{p}, \boldsymbol{r}_{<i})$. Therefore, define a next-token predictor $\texttt{LM}$ such that for every valid value of $i, \boldsymbol{p}$, and $\boldsymbol{r}$, we have $\texttt{LM}(\hat{r}_i = r_i\ ; \boldsymbol{p}, \boldsymbol{r}_{<i}) \coloneqq \mathbb{P}_{\mathcal{D}}(r_i|\boldsymbol{p}, \boldsymbol{r}< i)$. Then, sampling $\boldsymbol{r} \sim \mathcal{D}|\boldsymbol{p}$, is equivalent to autoregressively sampling $\boldsymbol{r} \overset{\text{ag}}{\sim} \texttt{LM}(\cdot\ ; \boldsymbol{p})$.*

The cleverness of this argument lies in the fact that it can apply to *any* imaginable distribution. Thus, as long as the next-token predictor is sufficiently expressive (with enough context, memory and compute), it can model both natural language and problem-solving. Thus, it may seem that next-token predictors are not antithetical to planning-based tasks.

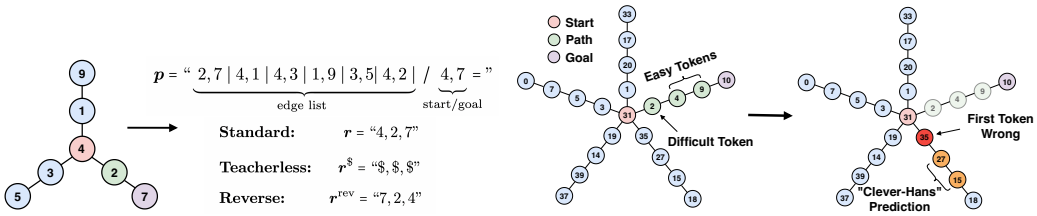
The snowballing errors criticism: A skeptic would attack the above by drawing attention to the possibility that in practice, we never learn a perfect next-token predictor. Furthermore, if a model makes an error in some token along the way, there is no way to backtrack and correct itself. The argument then goes that, even if the model is impressively accurate in each token, the probability of miniscule errors in each token exponentially snowballs along the way, leading to trivial accuracy by the end of a long sequence of tokens. This has been formalized in Dziri et al. (2023); LeCun (2024) in different ways. We formalize this failure, emphasizing its inference-time nature:

Failure 1. ***(Snowballing error due to autoregressive inference)*** *Consider a model $\texttt{LM}_\theta$, prefix $\boldsymbol{p}$ and a unique ground truth response $\boldsymbol{r}$ such that the per-token accuracy obeys $\forall i \leq L_{resp},\ \texttt{LM}_\theta\left(\hat{r}_i \neq r_i; \boldsymbol{p}, \boldsymbol{r}_{<i}\right) \geq \epsilon$. Then, for $\hat{\boldsymbol{r}} \overset{\text{ag}}{\sim} \texttt{LM}_\theta(\cdot\ ; \boldsymbol{p})$ the probability that the generation exactly matches the ground truth obeys:*

$$\mathbb{P}(\hat{\boldsymbol{r}} = \boldsymbol{r}) \leq (1-\epsilon)^{L_{resp}}.$$

The wrapper defense: A potential solution to the snowballing errors would be to deploy wrappers around the backbone model to explicitly verify errors and backtrack. This could be possible with reasoning techniques like chain/tree/graph of thought (Wei et al., 2022; Yao et al., 2023a; Besta et al., 2023; Yao et al., 2023b), or using the model to give itself feedback (Madaan et al., 2023; Huang et al., 2022; Shinn et al., 2023) (which is however doubted in Valmeekam et al. (2023)).

The weak-bias criticism: The back-and-forth so far has been concerned with autoregressive inference. We now turn our attention to the model *learning* via next-token prediction, using teacher-forcing. Our concern is that teacher-forcing may potentially encode stubborn pathologies into the backbone



(a) Graph tokenization (b) Failure of teacher-forcing

Figure 1: **Left:** Illustration of a path-star graph. The prefix $\boldsymbol{p}$ represents the adjacency list and the (central) start and goal node. The target is represented by $\boldsymbol{r}$. Under "standard" teacher-forcing, we condition the model on prefixes of $\boldsymbol{r}$ to predict $\boldsymbol{r}$. But in §5 we explore alternatives where we train without a teacher (condition on $\boldsymbol{r}^{\$}$ and predict $\boldsymbol{r}$) or train with a reversal (condition on and predict $\boldsymbol{r}^{\texttt{rev}}$). **Right:** Illustration of the failure of teacher-forcing on a path-star graph. The left sub-image marks the "easy tokens" which can be fit by the Clever Hans cheat (Failure 2a), while the "difficult token" cannot be learned (Failure 2b) due to lost supervision. The right sub-image shows how the model would behave during autoregressive inference, under the absence of the "teacher".

which no surfacial wrapper can cure. One such pathology could arise from the fact that, as per Fact 1, next-token prediction can fit *any* distribution. Thus, next-token prediction on language is (roughly) analogous to fitting image data with fully-connected networks rather than convolution: there is little structure to guide the learning process, which should result in (significant) sample-inefficiency.

The more-data or more-bias defense: In response, an optimist may suggest that the weak bias could be fixed, if we retained next-token prediction, but additionally introduce either (a) scale up data and compute (obeying the "bitter lesson" in Sutton (2019)) or (b) add regularizers or architectural constraints. However, in the next section, we pinpoint a more detrimental pathology introduced by next-token prediction during training, one that may not be fixed by adding more data or bias.

4 Failure due to Teacher-Forcing

While the last critique was that training to predict the next token is deficient in bias, we argue that there is an even worse problem: the next-token prediction objective (namely, teacher-forcing) provides supervision that is *incorrect* and *impossible* to learn from.

We build our argument based off of a path-finding problem on a simple class of graphs. We view this as a minimal setting that captures the core essence of what it means to solve problems with foresight, and the core essence of how teacher-forcing fails in such a setting, without unnecessary confounding factors. This task is also straightforward to solve, as we will see, thus making any observed failures remarkable. Thus we view this running example as a template for an intuitive argument that can be made about teacher-forced models on more general problems that require foresight.

4.1 Path-Finding on Path-Star Graphs: A Minimal and Easy Lookahead Task

Consider a path-finding problem on a graph G with a set of nodes $\{v_{\texttt{start}}, v_{\texttt{goal}}, v_1, v_2, \ldots\}$. The graph is a "path-star" graph with $v_{\texttt{start}}$ as the central node, with at least 2 paths (each of length at least 2 edges) emanating from it, with a unique path ending in $v_{\texttt{goal}}$. The task is to find a path from $v_{\texttt{start}}$ to $v_{\texttt{goal}}$. Correspondingly, we assume that the distribution $\mathcal{D}$ is over sequences where the prefix $\boldsymbol{p}$ represents a (randomly generated) graph, and the response represents the path from the start to the goal. In particular, we sample a graph G which is represented as an adjacency list as $\texttt{adj}(G) = e_1, e_2, \ldots$ where each edge $e = (v, v')$ is represented such that v' farther away from $v_{\texttt{start}}$ than v. We then set the prefix as $\boldsymbol{p} = (\texttt{adj}(G), v_{\texttt{start}}, v_{\texttt{goal}})$ so the model knows what the graph, and the desired start and goal states are. The ground truth response $\boldsymbol{r}$ corresponds to the sequence of vertices $\boldsymbol{r} = v_{\texttt{start}}, \ldots v_{\texttt{goal}}$ on the start-to-goal path. We visualize this in Fig. 1a.

The straightforward lookahead solution. Ideally, we want the model to learn a mapping from the input $\boldsymbol{p}$ consisting *only* of $(\texttt{adj}(G), v_{\texttt{start}}, v_{\texttt{goal}})$ to an output that is the full path $\boldsymbol{r}$. Two such solutions are possible. One idea is to plan by examining all the paths emanating from $v_{\texttt{start}}$ and

choosing the one that ends at $v_{\texttt{goal}}$. But a second, straightforward solution exists: the model simply needs to look ahead at the sequence "right-to-left" and observe that it corresponds to the one unique path starting from $v_{\texttt{goal}}$ and ending at $v_{\texttt{start}}$. After internally computing the path from $v_{\texttt{goal}}$ and reversing it, the model can emit its response.

Intuitively, we claim that teacher-forcing prevents learning either of these two solutions, causing failure. In teacher-forcing, for each vertex r_i in the path, we make the model learn a mapping from the input $(\boldsymbol{p}, \boldsymbol{r}_{<i})$ — not just $\boldsymbol{p}$ — to the output $\boldsymbol{r}_i$. The additional information $\boldsymbol{r}_{<i}$ in the input, we argue, is problematic in two ways, each of which prevents learning one of the above two solutions. We lay out our hypotheses below, and verify them in §5.

4.2 The Clever Hans Cheat

First, and most importantly, by revealing parts of the answer to the model as input, we allow the model to fit the data by *cheating*. To describe this, without loss of generality, consider a ground truth path that is of the form $\boldsymbol{r} = v_{\texttt{start}}, v_1, v_2, \ldots, v_{\texttt{goal}}$. With a slight abuse of the indexing notation, let $\boldsymbol{r}_{<i} = v_{\texttt{start}}, v_1, \ldots, v_{i-1}$ be the prefix of length i (so we index from 0 instead of 1).

Observe that nodes from v_2 onwards, until before $v_{\texttt{goal}}$, have precisely one edge going "away" from $v_{\texttt{start}}$. Thus, consider when the model is given as input, $(\boldsymbol{p}, \boldsymbol{r}_{<i})$ where $\boldsymbol{p} = (\texttt{adj}(G), v_{\texttt{start}}, v_{\texttt{goal}})$, to fit the target v_i. The model first merely needs to scan the adjacency list $\texttt{adj}(G)$ within $\boldsymbol{p}$ for the one edge containing v_{i-1} in the first position. Then, the model only has to predict the other node on that edge as v_i. Note though, this cheat cannot work on fitting the target v_1 given the input $\boldsymbol{r}_{<1} = v_{\texttt{start}}$ since $v_{\texttt{start}}$ has many outward edges. We illustrate this difference as the "easy" vs. "difficult" tokens in Fig. 1b.

Now, the above cheating mechanism for fitting the easy tokens does not require any planning or lookahead. It is simple, and implementable by an induction head-like module (Olsson et al., 2022). Owing to this simplicity, we hypothesize that these tokens will be quickly fit and ignored. This destroys any signal for the model to learn the relatively more complicated "right-to-left" solution. That solution requires looking at all tokens in $\boldsymbol{r}$, and then learning that they are simply the unique path from $v_{\texttt{goal}}$ spelled in reverse.

We emphasize two key aspects of the cheating behavior. First, these shortcuts are unlike previously-identified shortcuts (see §6) that map from the original input prefixes $\boldsymbol{p}$ to the ground truth $\boldsymbol{r}$. The behavior we identify is unique to the mapping from the teacher-forced prefix $\boldsymbol{p}, \boldsymbol{r}_{<i}$ to r_i. We christen this behavior as *Clever Hans cheating* (explained in remark below). Another notable point is that this does not come from a dearth of samples: even if we had infinite training data at our disposal, the model can still fit the easy tokens of all that data by Clever Hans cheating.

Remark 1. Clever Hans *(Pfungst & Rahn, 1911) was a famous show horse that could solve simple arithmetic tasks by repeatedly tapping with his hoof until he reached the correct count. It turns out however, that Clever Hans did not really solve the problem, but merely stopped tapping upon detecting certain (involuntary) facial cues from his coach. Clever Hans' answers were wrong when the coach was absent (as we will demonstrate even for our models).*

4.3 The Indecipherable Token

Perhaps, not all is lost. The model still needs to learn to predict the first node v_1, where it is not possible to fit the training data by the Clever Hans cheat. This may coerce the model into learning the other ideal solution which requires planning: examining all paths emanating from $v_{\texttt{start}}$ and observing that v_1 lies on the way to the $v_{\texttt{goal}}$. However, we argue this pattern is impossible to pick-up. Once the Clever Hans cheat is perfected, the model is deprived of information about much of the solution $\boldsymbol{r}_{1<} \coloneqq v_2, \ldots, v_{\texttt{goal}}$ which was once present as supervisory targets, but are now fit perfectly. The model is simply left with the task of mapping the input $(\texttt{adj}(G), v_{\texttt{start}}, v_{\texttt{goal}})$ to a single output node v_1. Merely with gradients from this supervision, we hypothesize that it is impossible to discover that v_1 is "the vertex that lies on the path to $v_{\texttt{goal}}$ node".

The role of prior biases. One may argue that if a human were given the input $(\texttt{adj}(G), v_{\texttt{start}}, v_{\texttt{goal}})$ and the target node v_1 they would trivially identify the underlying pattern. However, this may be because the problem, by happenstance, requires biases that exist *a priori* in humans. Humans already possess the idea of "path-finding" and thus are well-posed to identify how to fit node v_1. But in

generic problems, humans may not always be lucky. As a stark example, imagine being given a pair of numbers $x \circ y$ with a mystery operation $\circ$, and only the first digit of the solution z (with the rest hidden). We conjecture that learning how to produce z from x and y is an *impossible* problem for both models and humans alike, unless the the human/model happen to possess biases such as the submodules required for $\circ$ and the concept of composing/combining those submodules.

We informally generalize the above issues below:

Failure 2a. ***(Clever Hans cheating due to teacher-forcing)*** *Although there is a true mechanism that can recover $\boldsymbol{r}_i$ from the original prefix $\boldsymbol{p}$, there can be multiple other mechanisms that can recover a token r_i from the teacher-forced prefix $(\boldsymbol{p}, \boldsymbol{r}_{<i})$. These mechanisms can be simpler to learn thus disincentivizing the model from learning the true mechanism via that token.*

Failure 2b. ***(Indecipherable token due to lost supervision)*** *After the Clever Hans cheat is perfected during training, the model is deprived of a part of the supervision. This makes it (near)-impossible to learn the true mechanism from the remaining tokens alone.*

In summary, our argument is that, for planning tasks, teacher-forcing leads to both the failures above, in that order. As a result, the model does not learn the true mechanism — which is required during inference when we do *not* do teacher-forcing. As we demonstrate shortly, this can cause the model to fail on the very distribution it was trained on. This is a breakdown of planning abilities that emerges right from training, and is orthogonal to the inference-time Snowballing Failure (See §A).

5 Experimental Verification

In this section, we empirically demonstrate our hypothesized failure modes on the graph path-finding task for both the Transformer and Mamba architectures.

Dataset. We denote by $G_{d,l}(N)$ for $d, l, N \in \mathbb{N}$, a path-star graph consisting of a center node $v_{\texttt{start}}$ with degree $d \in \mathbb{N}$, meaning there are d different paths emerging from the center node, each consisting of $l-1$ nodes (excluding the start node). Node values are uniformly sampled from $(\{0, \ldots, N-1\})$ where N can be larger than the actual number of nodes in the path-star graph. In every graph, we use the center node as the starting node $v_{\texttt{start}}$ and then pick as $v_{\texttt{goal}}$, the last node of one of the paths chosen uniformly at random. The order of the edges in the adjacency list is randomized. We describe the tokenization in §D.1.

For each experiment, we generate the training and test graphs from the same $\mathcal{D}$, all with the *same* topology of $G_{d,l}(N)$ with fixed d, l and N. Thus, any failure we demonstrate is an *in-distribution* failure, and does not arise from the inability to generalize to different problem lengths (Anil et al., 2022). We note that while the graphs are all of the same topology, this is not a trivial memorization problem for the model, since the graphs are labeled differently, and the adjacency list randomized, so the model has to learn a general algorithm. Throughout the experiments, we fix the number of samples to $200k$ and fix the number of node values to $N = 100$ across topologies to enable diverse instantiations of the topology for training and testing.

Models. We evaluate models from two architectural families to highlight that the failures are not tied to a particular architecture but stem from the next-token prediction objective. For Transformers, we use from-scratch GPT-Mini, and pretrained GPT-2 large (Radford et al., 2019). For recurrent models, we use from-scratch Mamba (Gu & Dao, 2023). We optimize using *AdamW* (Loshchilov & Hutter, 2019) until perfect training accuracy. To rule out grokking behaviour (Power et al., 2022), we trained the cheaper models for as long as 500 epochs. More details are in §D.2.

5.1 Observations.

To demonstrate our hypothesized failure, we begin by establishing that our teacher-forced models fit the training data but fail in-distribution. Next, we design metrics to quantify the extent to which the two hypothesized mechanisms occur (Failures 2a, 2b). Finally, we design alternative objectives to intervene and remove each of the two failure modes, to test whether the performance improves. We report additional experiments in §C.1 quantifying the Snowballing Failure 1.

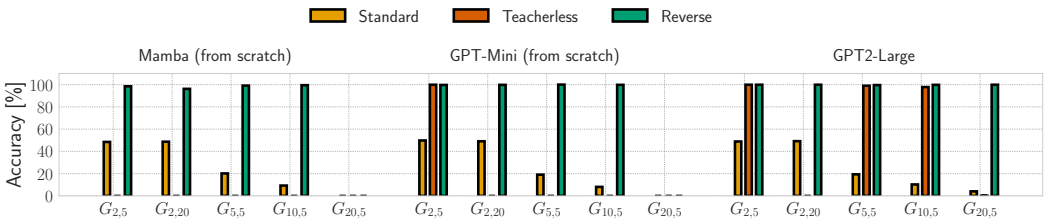


Figure 2: For different architectures, we report the accuracy of the standard teacher-forced model ($\texttt{Acc}_{\texttt{ag}}$, Eq 3), teacherless-trained model's accuracy ($\texttt{Acc}_{\$}$, Eq 7) and accuracy of the model trained with reversed targets ($\texttt{Acc}_{\texttt{rev}}$, Eq 8) evaluated on path-finding a range of graphs (with degree in the first subscript, and path length in the second).

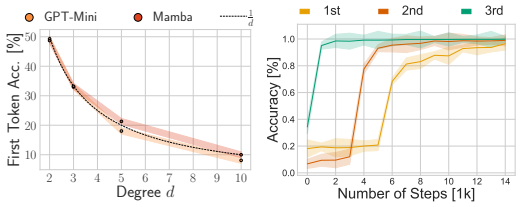


Figure 3: $\texttt{Acc}_{\texttt{1st}}(\texttt{LM}_\theta)$ (in percent %, Eq 5) for path-star graphs of various degrees $d \in \{2, 3, 5, 10\}$ for fixed path length $l = 5$ (left). Individual token accuracies (for v_1, v_2, v_3) for the graph $G_{5,5}$ under teacherless training (Eq 6) with GPT2-large (right).

Verifying in-distribution failure. For each distribution, we evaluate all our teacher-forced models by autoregressively generating solutions, and evaluating for an exact ground-truth match:

$$\texttt{Acc}_{\texttt{ag}}(\texttt{LM}_\theta) := \mathbb{P}(\hat{\boldsymbol{r}} = \boldsymbol{r}), \qquad \boldsymbol{p}, \boldsymbol{r} \sim \mathcal{D}, \quad \hat{\boldsymbol{r}} \overset{\texttt{ag}}{\sim} \texttt{LM}_\theta. \tag{3}$$

We report $\texttt{Acc}_{\texttt{ag}}(\texttt{LM}_\theta)$ for path-star graphs of varying topologies in Fig. 2 and Table 2. We observe that all models (even when pre-trained) struggle to learn the task accurately. The accuracy values are precisely limited to the value achievable if the model uniformly guesses a path starting from $v_{\texttt{start}}$ i.e., $\approx \frac{1}{d}$, thus establishing complete in-distribution failure. This is so even when trained to fit sample sizes up to $200k$ to 100% accuracy, and despite the fact that the training and test graphs have identical topology. Next, we quantitatively demonstrate how this stark failure arises from our two hypothesized mechanisms (Failure 2a, 2b).

Verifying Failure 2a (The Clever Hans cheat) We had hypothesized that the teacher-forcing model would fit the training data by cheating. Specifically, to predict node v_i in the true path, the model can exploit the ground truth node v_{i-1} that is revealed as input. Rather than learning to plan, the model would simply predict the node that is outwardly adjacent to v_{i-1}. To quantify whether this behavior emerges, we "teacher-force" the model with a uniform random neigbhor v_1' of $v_{\texttt{start}}$. We then look for whether the model indiscriminately applies the learned Clever Hans cheat here: does it simply follow along the path that emanates from the neigbhor v_1', not necessarily ending in $v_{\texttt{goal}}$? Formally, let $\texttt{Unif}(\mathcal{N}(v_{\texttt{start}}))$ denote a uniform distribution over the set of adjacent nodes of $v_{\texttt{start}}$. For any node v in the graph, denote by $\texttt{path}(v)$ the path emanating from v and going outwards, away from the start node. Notice that except for $v = v_{\texttt{start}}$, this path is unique. We thus measure

$$\texttt{Acc}_{\texttt{cheat}}(\texttt{LM}_\theta) := \mathbb{P}\left(\hat{\boldsymbol{r}}_{1<} = \texttt{path}(v_1')\right) \tag{4}$$

$$\text{where} \quad \boldsymbol{p}, \boldsymbol{r} \sim \mathcal{D}, \quad \hat{\boldsymbol{r}}_{1<} \overset{\texttt{ag}}{\sim} \texttt{LM}_\theta(\cdot; \boldsymbol{p}, v_{\texttt{start}}, v_1'), \quad v_1' \sim \texttt{Unif}(\mathcal{N}(v_{\texttt{start}})).$$

Empirically, we find that $\texttt{Acc}_{\texttt{cheat}}(\texttt{LM}_\theta)$ on a test set is $\approx 100\%$ almost across the board (except for graphs with very high degree where training is challenging). The exact values are in §C.1, Table 1. Thus, to fit the training data, the teacher-forced model has indeed exploited the Clever Hans cheat.

Verifying Failure 2b (The Indecipherable Token) Recall that the Clever Hans cheat only applies to all but the first node v_1 after $v_{\texttt{start}}$ lying on the path. After the Clever Hans cheat fits the rest of the

path during training, we hypothesized that node v_1 may become impossible to learn since the model is deprived of all information about the subsequent targets. To quantify this behavior, we evaluate how well the model is able to predict the difficult first node, v_1:

$$\texttt{Acc}_{\texttt{1st}}(\texttt{LM}_\theta) = \mathbb{P}\left(\hat{r}_1 = r_1\right), \qquad \boldsymbol{p}, \boldsymbol{r} \sim \mathcal{D}, \hat{\boldsymbol{r}} \overset{\texttt{ag}}{\sim} \texttt{LM}_\theta(\cdot; \boldsymbol{p}). \tag{5}$$

which we estimate using the held-out test set. As shown in Fig. 3 the model achieves a low $\texttt{Acc}_{\texttt{1st}}(\texttt{LM}_\theta)$, approximately $1/d$. Thus, the model indeed fails to identify that v_1 is the one on the path to $v_{\texttt{goal}}$. It instead resorts to emitting one of the d neighbors of $v_{\texttt{start}}$ at random.

Removing the Clever Hans cheat via teacherless training. We now design a training setup where we prevent Clever Hans cheating (Failure 2a) and examine how learning differs. Concretely, we modify teacher-forcing by replacing the *input* $\boldsymbol{r}$ (which reveals the ground truth) with an uninformative input $\boldsymbol{r}^{\$}$, consisting of the same special token \$ repeated l times. For supervision in the loss, we still use the original target $\boldsymbol{r}$. Thus, the model cannot fit the targets by looking at the prefixes $\boldsymbol{r}_{<i}$ and predicting v_i via cheating. Instead, the model only has access to the graph description in $\boldsymbol{p}$ in fitting the target v_i. Formally, we maximize:

$$\mathcal{J}_{\texttt{t-less}}(\theta) = \mathbb{E}_{\mathcal{D}}\Big[\sum_{i=1}^{L_{\texttt{resp}}} \log \texttt{LM}_\theta\Big(\hat{r}_i = r_i; \boldsymbol{p}, \boldsymbol{r}^{\$}_{<i}\Big)\Big]. \tag{6}$$

Denoting the model as $\texttt{LM}^{\$}_\theta$, we perform inference by conditioning on \$ tokens:

$$\texttt{Acc}_{\$}(\texttt{LM}^{\$}_\theta) = \mathbb{P}\left(\hat{\boldsymbol{r}} = \boldsymbol{r}\right) \qquad \boldsymbol{p}, \boldsymbol{r} \sim \mathcal{D}, \;\; \forall i \;\; \hat{r}_i \sim \texttt{LM}^{\$}_\theta(\cdot; \boldsymbol{p}, \boldsymbol{r}^{\$}_{<i}). \tag{7}$$

We report the accuracy in Fig. 2 and Table 3. In most cases, since the Clever Hans cheat is unavailable, the objective is too hard for the models to even fit the training data. However, surprisingly, on some of the easier graphs, the models not only fit the training data, but generalize well to test data. This positive result (even if limited) verifies two hypotheses. First, the Clever Hans cheat is indeed a cause of failure in the teacher-forced model. Secondly, and remarkably, with the cheat gone, these models are able to fit the first node which had once been indecipherable under teacher-forcing. This verifies our hypothesis that Clever Hans cheat absorbs away supervision that is critical to learn the first token.

Removing the Indecipherable Token failure via path reversal. Back in the teacher-forcing setup, we make a slight change: we train the model to predict the reversal of the true path $\boldsymbol{r}$ (similar to Lee et al. (2023); Shen et al. (2023) who explored this for addition). The model now needs to predict $v_{\texttt{goal}}$ first and make its way to $v_{\texttt{start}}$. The hope is that since there is only one unique path emanating from $v_{\texttt{goal}}$, there is no planning required, and thus we should never run into an Indecipherable Token. Every next token/node can be learned as the node that is inwardly adjacent to the previous node. Notationally, we let $\texttt{LM}^{\texttt{rev}}_\theta$ be the model trained to maximize $\mathcal{J}_{\texttt{next-token}}$ with the targets (and the teacher-forced inputs) set to $\boldsymbol{r}^{\texttt{rev}} = r_{L_{\texttt{resp}}}, \ldots r_1$, the reversal of $\boldsymbol{r}$. We then measure the autoregressive accuracy by comparing against $\boldsymbol{r}^{\texttt{rev}}$:

$$\texttt{Acc}_{\texttt{rev}}(\texttt{LM}^{\texttt{rev}}_\theta) = \mathbb{P}\left(\hat{\boldsymbol{r}} = \boldsymbol{r}^{\texttt{rev}}\right), \;\; \boldsymbol{p}, \boldsymbol{r} \sim \mathcal{D}, \hat{\boldsymbol{r}} \overset{\texttt{ag}}{\sim} \texttt{LM}^{\texttt{rev}}_\theta(\cdot; \boldsymbol{p}) \tag{8}$$

We display the results in Fig. 2 and Table 4. As expected, we observe that reversing significantly boosts learning, allowing even models trained from scratch to solve the task. This verifies that for the standard model, indecipherability of the first token was indeed a roadblock to successful learning.

We conclude with the note that the success of the reversal task makes the in-distribution failure of teacher-forcing even more remarkable. Although when viewed left-to-right, the problem requires complex planning and backtracking (i.e., evaluating multiple paths), when viewed right-to-left, the problem is solvable with a straightforward strategy. Evidently, the left-to-right teacher-forced model is unable to view the problem any differently and falls into a trap. The (pretrained) *teacherless* model however is free to choose the view that affords the simplest solution, and succeeds — this is evidenced in Fig. 3 where we see that the tokens are automatically learned right-to-left. Thus, we hope that the teacherless objective provides one possible way for future work to build alternatives to next-token prediction. We discuss more details and intuition about this finding in §B

6 Related Work

Arguments in support of next-token prediction. Shannon (1948; 1951) demonstrated that the English language has sufficient redundancy that given a large enough context window, next-token statistics can be used to generate English-sounding text. Empirically, Shlegeris et al. (2022) find that modern language models are surprisingly better than humans at next-token prediction on a text dataset (OpenWebText (Gokaslan & Cohen, 2019)). However, this does not preclude the possibility that next-token predictors may still be poor at planning. Furthermore, the above result may be confounded by the ability of language models to store more general knowledge than humans. Theoretically, Merrill & Sabharwal (2023b) show that autoregressive Transformers that generate chains of thought have a markedly larger *expressive* power. Most relevant to us is the positive *learnability* result in Malach (2023): even linear next-token predictors can learn complex tasks, provided there is chain-of-thought supervision. We do not contradict this. In our task, the first token requires an implicit chain of thought (e.g., the reversed path) that we do not provide.

Arguments against next-token prediction. We build on an emerging intuition that next-token predictors are ill-suited for planning. Momennejad et al. (2023) report failures on several planning tasks (including path-finding expressed as a word problem) and Bubeck et al. (2023) on various arithmetic, summarization and poem/story generation problems. McCoy et al. (2023) argue that, for such tasks, the performance of the model must greatly depend on its frequency during pretraining. However, we demonstrate that even when trained on many samples from a distribution, the next-token predictor fails on the very distribution. Goyal et al. (2016); Bengio et al. (2015) note that teacher-forcing causes a distribution shift between training and inference, which can lead to errors amplifying over time. Finally, Dziri et al. (2023); LeCun (2024) describe what we term as the snowballing error.

Broadly, our work extends and clarifies this discourse. First, we introduce the Clever Hans cheat failure mode, while also highlighting other core arguments in the debate. Next, we empirically report this failure in both the Transformer (Vaswani et al., 2017) and the Mamba architecture (Gu & Dao, 2023). This establishes that what we witness is indeed a failure of next-token prediction. Importantly, existing literature pins these failures broadly on the next-token prediction paradigm and interchangeably, on the inability of the autoregressive architecture to backtrack. We emphasize distinguishing the two types of next-token prediction (teacher-forcing and autoregressive inference) as they lead to distinct planning-related failures and require distinct solutions.

Shortcut-learning in language models. A line of work has empirically and theoretically analyzed another distinct type of shortcuts that language models learn to (partially) solve tasks such as learning multiplication (Dziri et al., 2023), automata (Liu et al., 2023), recursion (Young & You, 2023), reading comprehension (Lai et al., 2021) and multiple-choice questions (Ranaldi & Zanzotto, 2023) These shortcuts must *not* be confused with the Clever Hans cheating induced by teacher-forcing. First, these shortcuts exist independent of teacher-forcing: these are correlations between the prefix (such as the initial digits of two multiplicands) and the final answer (the initial digits of the product) in the distribution. But Clever Hans cheats arise only upon teacher-forcing as they are correlations between the prefixes of the answer itself to the rest of the answer. Second, the above shortcuts only fail out-of-distribution (such as when the number of multiplied digits is increased, where the failure is in length generalization (Anil et al., 2022)). Clever Hans causes in-distribution failure. Thirdly, the aforementioned observations are specific to Transformers. Our argument however only relies on the teacher-forcing objective with no reliance on the Transformer architecture, and is demonstrated even for the recurrent Mamba architecture. Please see § E for more related works.

7 Conclusions

Our work is an attempt at crystallizing an emerging debate about next-token prediction, that lies at the heart of modern language models. We advocate that we must avoid conflating the ideas of next-token prediction during autoregressive inference with that during teacher-forced training. Extending existing criticisms, we find a more deep-rooted issue inherent to the next-token prediction-training, rather than inherent to the data, the architecture or autoregressive inference. We also provide a minimal example that can help ground the next-token prediction debate. Both this example and the idea of *teacherless* training may help inspire alternative paradigms to next-token prediction in practice.

We discuss the limitations of our study in § F

REFERENCES

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.2, knowledge manipulation. *CoRR*, abs/2309.14402, 2023. doi: 10.48550/ARXIV.2309.14402. URL https://doi.org/10.48550/arXiv.2309.14402.

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay V. Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. Exploring length generalization in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Konstantine Arkoudas. Chatgpt is no stochastic parrot. but it also claims that 1 is greater than 1. *Philosophy & Technology*, 36(3):54, 2023.

Mikel Artetxe, Jingfei Du, Naman Goyal, Luke Zettlemoyer, and Veselin Stoyanov. On the role of bidirectionality in language model pre-training. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 3973–3985. Association for Computational Linguistics, 2022.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In Madeleine Clare Elish, William Isaac, and Richard S. Zemel (eds.), *FAccT '21: 2021 ACM Conference on Fairness, Accountability, and Transparency, Virtual Event / Toronto, Canada, March 3-10, 2021*, pp. 610–623. ACM, 2021.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 1171–1179, 2015.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. *CoRR*, abs/2308.09687, 2023. doi: 10.48550/ARXIV.2308.09687. URL https://doi.org/10.48550/arXiv.2308.09687.

Florian Böhm, Yang Gao, Christian M. Meyer, Ori Shapira, Ido Dagan, and Iryna Gurevych. Better rewards yield better summaries: Learning to summarise without references. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 3108–3118. Association for Computational Linguistics, 2019.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott M. Lundberg, Harsha Nori, Hamid Palangi, Marco Túlio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with GPT-4. *CoRR*, abs/2303.12712, 2023. doi: 10.48550/ARXIV.2303.12712. URL https://doi.org/10.48550/arXiv.2303.12712.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaïd Harchaoui, and Yejin Choi. Faith and fate: Limits of transformers on compositionality. *CoRR*, abs/2305.18654, 2023.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=jQj-_rLVXsj.

Anirudh Goyal, Alex Lamb, Ying Zhang, Saizheng Zhang, Aaron C. Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 4601–4609, 2016.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2023.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. Non-autoregressive neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=B1l8BtlCb.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. *CoRR*, abs/2305.01610, 2023. doi: 10.48550/ARXIV.2305.01610. URL https://doi.org/10.48550/arXiv.2305.01610.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In Karen Liu, Dana Kulic, and Jeffrey Ichnowski (eds.), *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1769–1782. PMLR, 2022.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

Yuxuan Lai, Chen Zhang, Yansong Feng, Quzhe Huang, and Dongyan Zhao. Why machine reading comprehension models learn shortcuts? In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pp. 989–1002. Association for Computational Linguistics, 2021.

Yann LeCun. Do large language models need sensory grounding for meaning and understanding? University Lecture, 2024.

Nayoung Lee, Kartik Sreenivasan, Jason D. Lee, Kangwook Lee, and Dimitris Papailiopoulos. Teaching arithmetic to small transformers. *CoRR*, abs/2307.03381, 2023.

Yingcong Li, Yixiao Huang, M. Emrullah Ildiz, Ankit Singh Rawat, and Samet Oymak. Mechanics of next token prediction with self-attention. In *27th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2024.

Chu-Cheng Lin, Aaron Jaech, Xin Li, Matthew R. Gormley, and Jason Eisner. Limitations of autoregressive models and their alternatives. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pp. 5147–5173. Association for Computational Linguistics, 2021.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=De4FYqjFueZ.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Ang Lv, Kaiyi Zhang, Shufang Xie, Quan Tu, Yuhan Chen, Ji-Rong Wen, and Rui Yan. Are we falling in a middle-intelligence trap? an analysis and mitigation of the reversal curse. *CoRR*, abs/2311.07468, 2023.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. *CoRR*, abs/2303.17651, 2023. doi: 10.48550/ARXIV.2303.17651. URL https://doi.org/10.48550/arXiv.2303.17651.

Eran Malach. Auto-regressive next-token predictors are universal learners. *CoRR*, abs/2309.06979, 2023. doi: 10.48550/ARXIV.2309.06979. URL https://doi.org/10.48550/arXiv.2309.06979.

R. Thomas McCoy, Shunyu Yao, Dan Friedman, Matthew Hardy, and Thomas L. Griffiths. Embers of autoregression: Understanding large language models through the problem they are trained to solve. *CoRR*, abs/2309.13638, 2023.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers, 2023a.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. *CoRR*, abs/2310.07923, 2023b. doi: 10.48550/ARXIV.2310.07923. URL https://doi.org/10.48550/arXiv.2310.07923.

Ida Momennejad, Hosein Hasanbeig, Felipe Vieira Frujeri, Hiteshi Sharma, Robert Osazuwa Ness, Nebojsa Jojic, Hamid Palangi, and Jonathan Larson. Evaluating cognitive maps and planning in large language models with cogeval. *CoRR*, abs/2309.15129, 2023. doi: 10.48550/ARXIV.2309.15129. URL https://doi.org/10.48550/arXiv.2309.15129.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. abs/2209.11895, 2022. doi: 10.48550/ARXIV.2209.11895. URL https://doi.org/10.48550/arXiv.2209.11895.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Koyena Pal, Jiuding Sun, Andrew Yuan, Byron C. Wallace, and David Bau. Future lens: Anticipating subsequent tokens from a single hidden state. In Jing Jiang, David Reitter, and Shumin Deng (eds.), *Proceedings of the 27th Conference on Computational Natural Language Learning, CoNLL 2023, Singapore, December 6-7, 2023*, pp. 548–560. Association for Computational Linguistics, 2023.

Oskar. Pfungst and Carl Leo. Rahn. *Clever Hans (the horse of Mr. Von Osten) a contribution to experimental animal and human psychology*. New York, H. Holt and company, 1911. URL https://www.biodiversitylibrary.org/item/116908. https://www.biodiversitylibrary.org/bibliography/56164.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets, 2022.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pp. 2401–2410.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Leonardo Ranaldi and Fabio Massimo Zanzotto. Hans, are you clever? clever hans effect analysis of neural systems, 2023.

C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3): 379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.

C. E. Shannon. Prediction and entropy of printed english. *The Bell System Technical Journal*, 30(1): 50–64, 1951. doi: 10.1002/j.1538-7305.1951.tb01366.x.

Ruoqi Shen, Sébastien Bubeck, Ronen Eldan, Yin Tat Lee, Yuanzhi Li, and Yi Zhang. Positional description matters for transformers arithmetic. *CoRR*, abs/2311.14737, 2023. doi: 10.48550/ARXIV.2311.14737. URL https://doi.org/10.48550/arXiv.2311.14737.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023.

Buck Shlegeris, Fabien Roger, Lawrence Chan, and Euan McLean. Language models are better than humans at next-token prediction. *CoRR*, abs/2212.11281, 2022. doi: 10.48550/ARXIV.2212.11281. URL https://doi.org/10.48550/arXiv.2212.11281.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 10107–10116, 2018.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. Learning to summarize from human feedback. *CoRR*, abs/2009.01325, 2020. URL https://arxiv.org/abs/2009.01325.

Richard Sutton. The bitter lesson. 2019. URL http://www.incompleteideas.net/IncIdeas/BitterLesson.html.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

Karthik Valmeekam, Matthew Marquez, and Subbarao Kambhampati. Can large language models really improve by self-critiquing their own plans? *CoRR*, abs/2310.08118, 2023. doi: 10.48550/ARXIV.2310.08118. URL https://doi.org/10.48550/arXiv.2310.08118.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information*

Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022, 2022.

Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989. doi: 10.1162/neco.1989.1.2.270.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *CoRR*, abs/2305.10601, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023b. URL https://openreview.net/pdf?id=WE_vluYUL-X.

Tom Young and Yang You. On the inconsistencies of conditionals learned by masked language models. *CoRR*, abs/2301.00068, 2023. URL https://doi.org/10.48550/arXiv.2301.00068.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul F. Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *CoRR*, abs/1909.08593, 2019. URL http://arxiv.org/abs/1909.08593.

A TEACHER-FORCING FAILURE AND SNOWBALLING FAILURE ARE DISTINCT

We emphasize that, while both the Clever Hans failure mode and the Snowball mode are both indicative of the inability to plan, these failure modes are also orthogonal to each other, and demand different solutions. We make this a bit more formal:

Proposition 3. *In the path finding problem of §4.1, there exists a next-token predictor that experiences Failures 2a, 2b due to teacher-forcing, but not the snowballing error Failure 1 due to autoregressive inference. Conversely, there exists a next-token predictor that experiences the latter failure but not the former.*

Proof. Consider the model learned via teacher-forcing on the graph problem. During inference, we saw that it suffers a tremendous error right in the first step (with accuracy of $1/d$ for degree d of the start node). Thus, during inference the error does not accumulate over length. In fact, if only the first node is set correctly during inference, a model with the perfect Clever Hans cheat, would achieve 100% accuracy rate. Such a model does not experience snowballing errors.

On the other hand, consider a model, that in each step predicts the correct next vertex with a high accuracy of $1 - \epsilon$ for small ϵ. Such a model clearly has learned the correct plan, albeit with minor errors in each token. These errors however can snowball during inference. Thus, this model has no failure due to teacher-forcing, but will fail during autoregressive inference, if the path length is long. □

Differing solutions. Based on the above simple illustration, we argue how the two failures need different approaches. Specifically, while snowballing errors may be fixable via "backtracking-and-planning" wrappers, teacher-forcing failures cannot. In the above example, the first model which only experiences the teacher-forcing failure, assigns a low probability of $1/d$ to the true plan. In a more generic problem, this probability may be exponentially small (inversely proportional to the number of all possible plans). A wrapper on such a backbone would essentially have to brute-force search through all the exponential solutions. Thus, it cannot cure the pathologies inherent to a teacher-forced model.

On the other hand, consider the model in the above example that assigns $1 - \epsilon$ probability to the correct solution at every step. In this simple setting, we can use greedy sampling as a wrapper to trivially extract the true plan.

B MORE ON TEACHERLESS TRAINING

We discuss a possible intuition behind how teacherless training solves the path-finding task. Our hypothesis is that the model automatically learns to fit the targets in the reverse order, since the path *from* the $v_{\texttt{goal}}$ is unique. This is indeed what we find in Fig 3, where the accuracies of the later tokens become higher earlier.

Note though that this is a fairly difficult computation to implement. First, when the model predicts v_i, it must "know" the identity of v_{i+1} in one of its internal representations. This is because the identity is not available on the input side, in the absence of the teacher. Then, by induction, when predicting the first node v_1, the model must know the identity of *all* the other nodes in the path. In other words, the model must have (a) computed and (b) stored the whole solution in its hidden representations before it outputs the first token. This is a substantial type of lookahead that some of our models are able to achieve under teacherless training.

C MORE EXPERIMENTAL RESULTS

C.1 SNOWBALL FAILURE

To explicitly measure to what degree the model falls victim to the snowball effect, we train *GPT-Mini* on graphs of various path lengths l. In order to remove the failure stemming from the difficult first token, we teacher-force the model for the first token and then check how accurate the generations are for subsequent tokens. More concretely, we evaluate

$$\texttt{Acc}_{\texttt{sb}}(\texttt{LM}_\theta) = \mathbb{P}\left(\hat{\boldsymbol{r}}_{1<} = \boldsymbol{r}_{1<}\right) \tag{9}$$

$$\text{where} \quad \boldsymbol{p}, \boldsymbol{r} \sim \mathcal{D}, \quad \hat{\boldsymbol{r}}_{1<} \overset{\text{ag}}{\sim} \texttt{LM}_\theta(\cdot; \boldsymbol{p}, r_1)$$

If $\texttt{Acc}_{\texttt{sb}}(\texttt{LM}_\theta)$ is ≈ 1, then *Failure 1* is not prominent in our task. If $\texttt{Acc}_{\texttt{sb}}(\texttt{LM}_\theta) \ll 1$, then clearly teacher-forcing is responsible for surpressing errors in generation, strongly hinting at the fact that *Failure 1* is at play. We display the results in Fig. 4 (left). We observe that the accuracy $\texttt{Acc}_{\texttt{sb}}$ is barely affected even for graphs with very long paths $L = 40$.

As another metric, we proceed token by token during inference, and evaluate the probability of correctly predicting all tokens up to the current one. We report this for $G_{2,40}$ in Fig. 4 (right). Similarly, while the success probability does decay for larger length (at an exponential rate), it remains very high due to the failure events being so unlikely. We thus conclude that *Failure 1* is not as prominent in this setting.

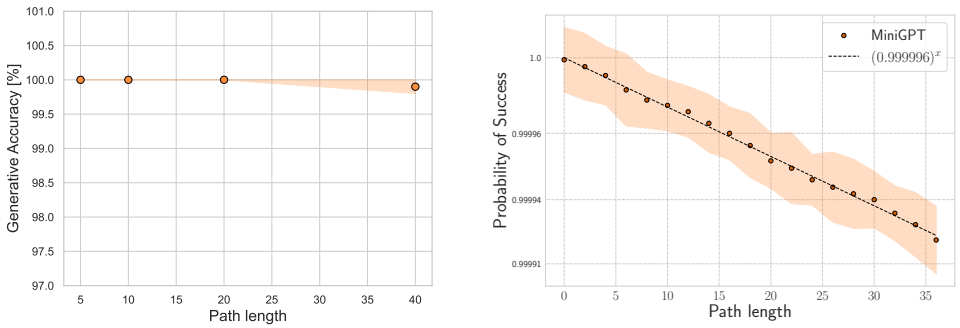


Figure 4: Accuracy of $\texttt{LM}_\theta$ when conditioned on the first difficult token (left) for graphs of various length. Probability of correct prediction of $\texttt{LM}_\theta$ as a function of current token position on $G_{2,40}$, as we walk towards the goal.

C.2 CLEVER HANS CHEATING ACCURACIES

In Table 1 we display the Clever Hans cheating accuracies $\texttt{Acc}_{\texttt{cheat}}(\texttt{LM}_\theta)$. We observe that in almost all cases, all the models achieve nearly perfect cheating accuracies. The only exception is the high-degree graph $G_{20,5}$ where all models struggle to even fit the training data.

	$G_{2,5}$	$G_{2,20}$	$G_{5,5}$	$G_{10,5}$	$G_{20,5}$
GPT-MINI	99.7	100	100	99.8	0.0
GPT2-LARGE	99.8	99.7	100	99.8	0.0
MAMBA	97.6	98.3	99.5	95.9	0.0

Table 1: Evaluating Clever Hans cheating accuracies $\texttt{Acc}_{\texttt{cheat}}(\texttt{LM}_\theta)$ (in percent $\%$) for different types of graphs.

C.3 MORE DETAILED ACCURACIES

We report more detailed accuracy values per model in the following tables. We display standard accuracy $\texttt{Acc}_{\texttt{ag}}(\texttt{LM}_\theta)$ in Table. 2, teacherless accuracy $\texttt{Acc}_{\$}(\texttt{LM}_\theta)$ in Table. 3 and reverse accuracy $\texttt{Acc}_{\texttt{rev}}(\texttt{LM}_\theta)$ in Table. 4. In general we observe that solving the task with standard next-token prediction is very tough and performance is limited to $\frac{1}{d}$ where d is the degree of the graph $G_{d,l}$.

	$G_{2,5}$	$G_{2,20}$	$G_{5,5}$	$G_{10,5}$	$G_{20,5}$
GPT-MINI	49.8	49.1	19.1	8.1	0.0
GPT2-LARGE	48.9	49.2	19.4	10.3	3.5
MAMBA	48.5	48.7	20.2	9.3	0.0

Table 2: Autoregressive accuracies $\texttt{Acc}_{\texttt{ag}}(\texttt{LM}_\theta)$ (in percent $\%$) for different types of graphs.

Teacherless training on the other hand works very well with GPT2-Large, allowing it to solve most graph tasks perfectly. From-scratch models however also struggle to learn the task in this fashion (except for GPT-Mini on the simplest graph, $G_{2,5}$).

	$G_{2,5}$	$G_{2,10}$	$G_{2,20}$	$G_{5,5}$	$G_{10,5}$	$G_{20,5}$
GPT-MINI	99.9	0.0	0.0	0.0	0.0	0.0
GPT2-L	99.9	98.8	0.0	99.0	97.8	0.0
MAMBA	0.0	0.0	0.0	0.0	0.0	0.0

Table 3: Autoregressive accuracy $\texttt{Acc}_{\$}$ when using a teacherless response.

Finally, reversing the sequence significantly simplifies the problem for all the models, allowing near perfect accuracies across all graphs.

	$G_{2,5}$	$G_{2,20}$	$G_{5,5}$	$G_{10,5}$	$G_{20,5}$
GPT-MINI	99.7	99.8	100	99.8	0.0
GPT2-LARGE	99.9	99.9	99.6	99.8	99.9
MAMBA	98.5	96.2	99.1	99.5	0.0

Table 4: Autoregressive accuracy $\texttt{Acc}_{\texttt{rev}}$ when reversing the response $\boldsymbol{r}$.

D OTHER EXPERIMENTAL DETAILS

D.1 TOKENIZATION

We tokenize the graph in the following manner: (1) we first tokenize the randomly shuffled edge list as "$|v_1\ v_2|v_3\ v_4|...$" where the first vertex in each edge is the one closest to $v_{\texttt{start}}$, (2) then append start and goal node as "$/v_{\texttt{start}}\ v_{\texttt{goal}} =$ " and (3) then append the full path repeating start and goal node, "$v_{\texttt{start}}\ v_{i_1} \dots v_{i_{l-1}}\ v_{\texttt{goal}}$". Note that (1) and (2) make up the prefix $\boldsymbol{p}$, which the model does not learn to predict. Then, (3) is the target sequence that the model aims to learn. The vocabulary size is thus given by $N + 3$, where we add entries for the special tokens "$|$", "$/$" and "$=$". When using the pre-trained models GPT2 we use the tokenizer that was employed for pre-training, in this case the *Byte-Pair tokenizer* (Radford et al., 2019).

D.2 MODELS

When training Transformer models from scratch, we use a small model consisting of $n_{\text{layers}} = 12$ blocks with embedding dimension $e_{\text{dim}} = 384$, $n_{\text{heads}} = 6$ attention heads and MLP expansion factor $e = 4$, coined *GPT-Mini*. For pre-trained models, we consider GPT2-Large with $n_{\text{layers}} = 36$, $e_{\text{dim}} = 1280$, $n_{\text{heads}} = 20$ and expansion factor $e = 4$ (Radford et al., 2019). To further evaluate purely recurrent models, we perform experiments with the recent Mamba model (Gu & Dao, 2023). We train the Mamba models from scratch with 12 layers and embedding dimension 784. We train all the models with the *AdamW* optimizer (Loshchilov & Hutter, 2019). For models trained from scratch we use a learning rate of $\eta = 0.0005$ while for pre-trained models we use a smaller one of $\eta = 0.0001$. In both cases we use weight decay of strength 0.01. Models from scratch are trained for up to 500 epochs in order to ensure convergence. Pre-trained models require less training time and we usually fit the training data perfectly after 10 epochs.

E MORE RELATED WORK

Other arguments about next-token prediction We discuss some criticisms of next-token prediction, orthogonal to our main discussion regarding planning. Allen-Zhu & Li (2023); Lv et al. (2023) report that language models that are trained on `A equals B` are unable to infer `B equals A`, which Allen-Zhu & Li (2023) suggest is due to "autoregressive left-right training". Lin et al. (2021) are concerned with failures that arise from the model size being fixed in comparison to the length of the sequence that is being scored. Their theoretically prove that, asymptotically, the parameter count must grow with the sequence length. Li et al. (2024) provide an Transformer-specific analysis of how self-attention affects the optimization geometry of next-token prediction.

Other limitations of Transformers Merrill & Sabharwal (2023a) identify limitations of the representative power of Transformer architecture when the arithmetic precision is logarithmic in the number of input tokens. Bender et al. (2021) criticize GPT-like language models as simply parroting out training data with minor stochasticity, while others Arkoudas (2023) report that such models struggle with reasoning, even if not a stochastic parrot. Young & You (2023) studies masked language (T5, BERT) models and not causally trained models and argue there are inconsistencies in the probabilities that they assign. E.g., when conditioned on '`white`', the probability of '`rice`' may be higher '`bread`' but the probability of '`white bread`' and '`white rice`' are the opposite. Artetxe et al. (2022) empirically analyze the effect of bidirectional attention and bidirectional supervision (as in masked

language modeling) during pretraining on the ability of the model to do various things, including next-token prediction.

Finally, we note that (Ranaldi & Zanzotto, 2023) use the term Clever Hans effect to denote how models can pick up spurious correlations between the position of a choice in a multiple-choice question, and the correctness of the answer. We note that the above correlation is inherent to the distribution, and independent of teacher-forcing. We distinguish this from the Clever Hans *cheating* which happens under the guidance of teacher-forcing.

Going beyond next-token prediction-based training. Different lines of work have explored models that go beyond next-token prediction-based training for language. This includes non-autoregressive models (Gu et al., 2018), diffusion models (Gong et al., 2023), and variants of Transformers learning to predict multiple tokens at the same go (Qi et al.). Our teacherless training follows this line of work, albeit with a much simpler approach that involves a trivial modification to teacher-forcing. Note that while research in parallel decoding too is concerned with predicting multiple future tokens (Stern et al., 2018), the goal is purely inference-time efficiency.

One may argue that reinforcement learning-based training (Böhm et al., 2019; Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022) is another way to build backbones that go beyond teacher-forcing. However, it is worth noting that the gradients in these techniques boil down to teacher-forcing on the model's own generated answer. Furthermore, if we desire that the model be able to generate a solution that can plan ahead of time, it is unclear how a model can go from a complete inability to plan (that may assign near-zero probability to the true plan in an exponential space of solutions), can manage to discover the correct plan simply through preference-based feedback.

Finally, we note that some works (Gurnee et al., 2023; Meng et al., 2022; Pal et al., 2023) aim to recover future tokens that an already-trained model may predict based on the internal layers of the current token. Note that the success of this does not imply that the model necessarily plans well. This only means that it is possible to recover what the already-trained model wants to generate in the future (which may simply be a bad plan).

F LIMITATIONS

We emphasize the limitations of our findings. First, our arguments are purely empirical and conceptual. We have not *provably* demonstrated that the teacher-forced model is "stuck" at the failed solution regardless of how long it is trained. A proof would particularly benefit our informal Indecipherable Token argument (and the role of prior biases in it), which is not as straightforward an argument as the Clever Hans cheat. We have also not demonstrated failure for very large models such as `Llama2` (Touvron et al., 2023) or `Mistral` (Jiang et al., 2023). Next, beyond the graph path-finding setting, we have not characterized the range of problems where teacher-forcing-induced failure may occur. We only intuitively believe it should extend to other problem-solving tasks and creative-writing tasks that require lookahead. But it is certainly unclear if it generalizes to run-of-the-mill text-generation tasks.