Future Lens: Anticipating Subsequent Tokens from a Single Hidden State

Koyena Pal
Northeastern University
pal.k@northeastern.edu

Jiuding Sun
Northeastern University
sun.jiu@northeastern.edu

Andrew Yuan
UMass Amherst
awyuan@umass.edu

Byron C. Wallace
Northeastern University
b.wallace@northeastern.edu

David Bau
Northeastern University
d.bau@northeastern.edu

Abstract

We conjecture that hidden state vectors corresponding to individual input tokens encode information sufficient to accurately predict several tokens ahead. More concretely, in this paper we ask: Given a hidden (internal) representation of a single token at position t in an input, can we reliably anticipate the tokens that will appear at positions $\geq t+2$? To test this, we measure linear approximation and causal intervention methods in GPT-J-6B to evaluate the degree to which individual hidden states in the network contain signal rich enough to predict future hidden states and, ultimately, token outputs. We find that, at some layers, we can approximate a model's output with more than 48% accuracy with respect to its prediction of subsequent tokens through a single hidden state. Finally we present a "Future Lens" visualization that uses these methods to create a new view of transformer states.

1 Introduction

Do hidden states in large language models (LLMs) encode tokens farther than a single token ahead? If so, how can we decode this sequence of tokens from a single state? In this work we empirically investigate these questions using GPT-J-6B (Wang and Komatsuzaki, 2021). We train models to predict hidden states several tokens ahead of a given position t based *only* on a contextualized representation of the input at this position.

Auto-regressive transformer language models are typically trained to predict one token ahead, but recent work has hinted that individual hidden states may contain more information than just probabilities of the following token. For example, Meng *et al.* (2022a) trace information flow from subject tokens to associated attribute predictions many steps ahead. Elsewhere, Gurnee *et al.* (2023) suggest that neurons in early layers are dense with information, while middle layers have dedicated neurons that represent high-level contextual features.

Other related efforts have passed hidden intermediate states directly to the decoder head (skipping in-between layers) to "verbalize" such embeddings (Din et al., 2023; Belrose et al., 2023; nostalgebraist, 2020). Studies of memorization (Carlini et al., 2021, 2023, 2019) have identified the presence of very long memorized sequences generated by language models, and Zhang and He (2020) shows that progressively dropping layers during computation can still achieve a similar prediction output of the model when compared against their fully computed model run.

In this work we ask: To what extent can we extract information about future (beyond subsequent) tokens from a single hidden token representation? To answer this, we conduct three experiments. First, extending the ideas of Tuned Lens (Belrose et al., 2023; Din et al., 2023) and the Logit lens (nostalgebraist, 2020), we train linear models to approximate future model predictions several tokens in the future, in order to reveal the extent to which individual hidden states may directly encode subsequent tokens. Second, we perform a causal intervention study in which we transplant individual hidden states from one context to a completely different context and measure the extent to which future tokens that were predicted in the original context can be predicted in the foreign context. Finally, we fit a "soft prompt" to explicitly learn an optimal prompt that permits reading out information about subsequent tokens from a hidden state.

2 Methods

To unveil the information about "future" tokens implicitly encoded in a single transformer state vector, we develop and compare several methods for predicting future tokens from a single hidden state. Each of our methods has the same goal: Extract accurate predictions of a model's probability distribution several tokens ahead, based on the information in only one hidden state at a single layer at one token of the transformer.

For our evaluations we use an autoregressive transformer (Vaswani et al., 2017) language model defined as a function $G : X \to Y$ over vocabulary V of size $|V| = d_v$. G takes in a sequence of tokens $x = [x_1, ..., x_T] \in X, x_i \in V$ and maps this to a probability distribution $y_T \in Y \subset [0, 1]^{d_v}$, which (greedily) predicts the next-token $x_{T+1} = \operatorname{argmax} y_T$. To generate additional tokens, the top predicted token x_{T+1} is added to the sequence of tokens $[x_1, ..., x_T, x_{T+1}]$ and the process is repeated until the next N tokens are produced.

To calculate each predicted probability distribution from an input sequence x, the transformer performs a sequence of computations at L layers; this can be decomposed as:

$$G(x) = D(b_L(\cdots(b_2(b_1(E(x))))\cdots)) \tag{1}$$

Where the first step $E :\to \mathbb{R}^{d_h}$ embeds each input token into an initial hidden representation, $e(x_i) = h_i^0 \in \mathbb{R}^{d_h}$; each layer $b_l : \mathbb{R}^{d_h \times T} \to \mathbb{R}^{d_h \times T}$ transforms the sequence of representations; and the decoder $D : \mathbb{R}^{d_h} \to Y$ decodes the predicted probability distribution $y_T = D(h_T^L)$ from the last layer at the last token. We write the output of layer l as $H_l = b_l(H^{l-1})$, where:

$$H^l = (h_1^l, ..., h_T^l) \in \mathbb{R}^{d_h \times T} \tag{2}$$

When generating a sequence of tokens beyond the given starting prefix of length T, we write:

$$y_{T+i} = G([x_1, .., x_{T+i-1}, x_{T+i}]) \tag{3}$$
$$x_{T+i+1} = \operatorname{argmax} y_{T+i} \tag{4}$$

Our goal is to devise methods that can anticipate what G will predict for y_{T+1} through y_{T+N} from only a single hidden state at h_T^l.

2.1 Direct Vocabulary Prediction

Let h_T^l denote the hidden representation induced by G for token x_T at intermediate layer $l \leq L$, and let y_{T+N} denote the subsequent-token distribution predictions produced by G after token x_{T+N}. To predict y_{T+N} from h_T^l alone, we train a linear model g_θ to predict logits $\hat{z}_{T+N}$ that approximate $\hat{y}_{T+N}$ after softmax:

$$\hat{z}_{T+N} = g_\theta(h_T^l) \tag{5}$$
$$\hat{y}_{T+N} = \operatorname{softmax}(\hat{z}_{T+N}) \approx \hat{y}_{T+N}$$

Since this model directly predicts the subsequent predictions over the full vocabulary from h_T^l, we call it the direct vocabulary prediction model.

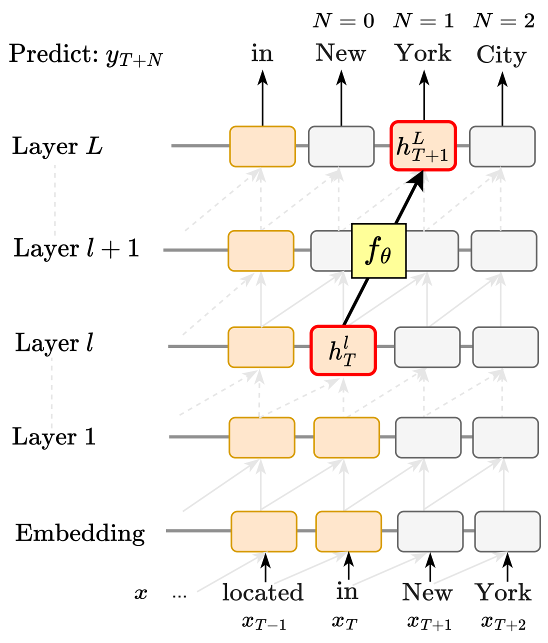

Figure 1: LLM to Linear Model Approximation Overview. Given a hidden state, h_T^l, the linear model, f_θ, is trained to output a future hidden state h_{T+1}^L. In this example h_T^l is the encoding that would lead to the prediction of 'New,' and f_θ uses only that information to predict h_{T+1}^L that would predict 'York.'

2.2 Linear Model Approximation

We also test a linear model based on the tuned logit lens (Belrose et al., 2023; Din et al., 2023) approach, which anticipates future hidden states within the transformer and decodes them using the pretrained decoder head. Differently from that work, we model hidden states at future tokens in rather than only at later layers.

Beginning with the hidden representation h_T^l, we create a model to predict a hidden state h_{T+N}^L at the final layer L, and subsequent token x_{T+N}. To predict h_{T+N}^L from h_T^l, we train a linear model:

$$\hat{h}_{T+N}^L = f_\theta(h_T^l) \approx h_{T+N}^L \tag{6}$$

The vocabulary can be read from the predicted $\hat{h}_{T+N}^L$ by applying the pretrained decoder head of the transformer. In Figure 1, we show an example of one such linear model. Suppose that we have trained a linear model parameterized by θ, f_θ, that takes in the last token hidden representation of the input at layer l to generate a hidden state at layer L of the following token hidden representation. When we input the following in G: "Madison Square Garden is located in", we get "New" as the highest-probability prediction at $N = 0$ and "York" at $N = 1$. We use the linear model to approximate this based on the hidden representation of T_N (i.e., “in") at layer $l \leq L$ as our input; the ideal output of the linear model given this would be the hidden state at T_{N+1} and layer L, which is associated with predicting “York” as the most probable token.

This approach differs from the direct vocabulary approach by reusing the pretrained decoder head of the transformer. We find that this marginally aids predictions at the latest layers l near L. Based on the observation that other pretrained transformer parameters may encode memorized calculations that facilitate decoding of subsequent tokens, we next turn to other approaches that utilize larger portions of the pretrained transformer to predict future tokens.

2.3 Fixed Prompt Causal Intervention

The next method we consider involves a single-state causal intervention where we transplant the hidden state h^l_T into the transformer while it is decoding an unrelated bit of context. The question is whether this transplantation steers the model to generate tokens related to the prefix that induced h^l_T. If it does, this indicates that information about subsequent tokens (in the original sequence) is prominently encoded in h^l_T.

Figure 2 depicts the procedure. On the left, we show the original context from which h^l_T is read; here $x = [x_1, ..., x_T]$ is “Madison Square Garden is located in" where x_1 is “Madison" and x_T is “in". This results in a sequence of outputs $[x_{T+1}, ..., x_{T+N}]$ which will read “New York City.” On the right, we run a single generic fixed-context prompt $c = [c_1, ..., c_M]$ (e.g., “Please, tell me something about" where c_1 is “Please" and c_M is “about") through the transformer. One would not anticipate that this generic prompt would cause the transformer to predict “New York City”.

Using an intervention, we now directly test that hypothesis that a single hidden state at layer l and token T within the original run contains the information necessary to predict subsequent tokens. We transplant the original run’s state vector h^l_T into the corresponding location h^l_M in the fixed-context run, then allow the transformer to proceed. If the necessary contextual information is present in the new run, the resulting tokens generated would become “New" for the current token generation and “York" and “City” for the subsequent token generations.

Formally, let the sequence $x = [x_1, ..., x_T]$ denote an input context that causes the model to subsequently generate $[x_{T+1}, ..., x_{T+N}]$, and let and $c = [c_1, ..., c_M]$ represent a generic fixed-context prompt where T and M represent the lengths of the original and fixed input prompts, respectively. When each are passed through G, we get the following predicted distributions:

$$y_T = G(x) \in [0, 1]^{|V|} \tag{7}$$
$$\hat{y}^*_M = G(c) \in [0, 1]^{|V|}$$

Denote the intervention that replaces h^l_M from the fixed-context run with state h^l_T from the original run as:

$$\hat{y}_M = G(c \,||\, h^l_M := h^l_T) \tag{8}$$

If, after the intervention, the new predicted distribution $\hat{y}_M \approx y_M$ approximates the prediction in the original context, that will reveal that h^l_T specifically encodes information needed for that prediction.

Furthermore, we can deduce what h^l_T encodes about subsequent token predictions n steps ahead by adding the generated tokens to the input and comparing the following predictions:

$$y_{T+i} = G(x + [x_{T+1}, ..., x_{T+N}]) \tag{9}$$
$$\hat{y}_{M+i} = G(c + [x_{T+1}, ..., x_{T+N}] \,||\, h^l_M := h^l_T)$$

The context prompt c could be chosen as any sequence of tokens. In practice, some prompts are more amenable to this intervention than others. In our experiments, we will test a small set of highly generic phrases.

2.4 Learned Prompt Causal Intervention

In the previous section, we have described an intervention that could reveal information predictive of upcoming tokens encoded in a single hidden state, by steering generation when grafted into completely unrelated contexts.

However, in cases where this “fails”, it does not necessarily mean that the hidden state does not encode similar information; it may just be less prominent. To evaluate the degree to which such signal is present in these cases, we next explore an approach in which we *learn* to surface information about subsequent tokens from individual contextual token embeddings. This procedure is shown in Figure 3.

Specifically, we optimize a parameterized prefix, $c_{opt} = [c_1, ..., c_M]$ to extract this information from

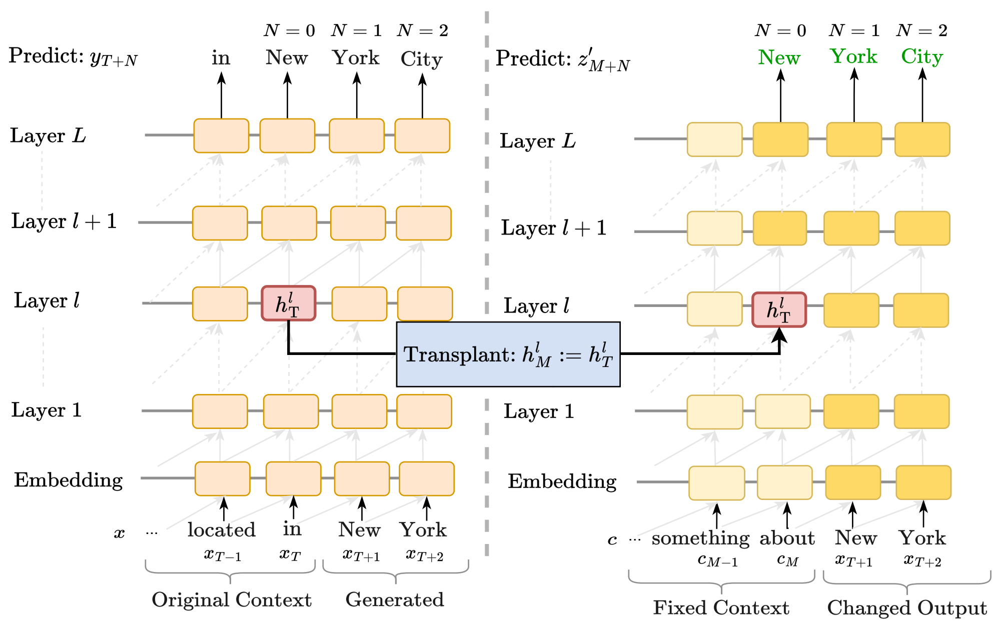

Figure 2: Illustration of Fixed prompt Causal Intervention. The left and right sides represent two different transformer model runs. On the left hand side, we have the original run of *Madison Square Garden ... in New York*. We transplant the hidden state, h_T^l to the other transformer model run, which has a fixed generic context, *Tell me something about*, as its input. With h_T^l replacing the hidden state at h_M^l, we measure the tendency of this modified transformer run to reveal the probability distribution in h_T^l. In such cases, it would reveal that h_T^l was predicting, for instance, 'New York City.'

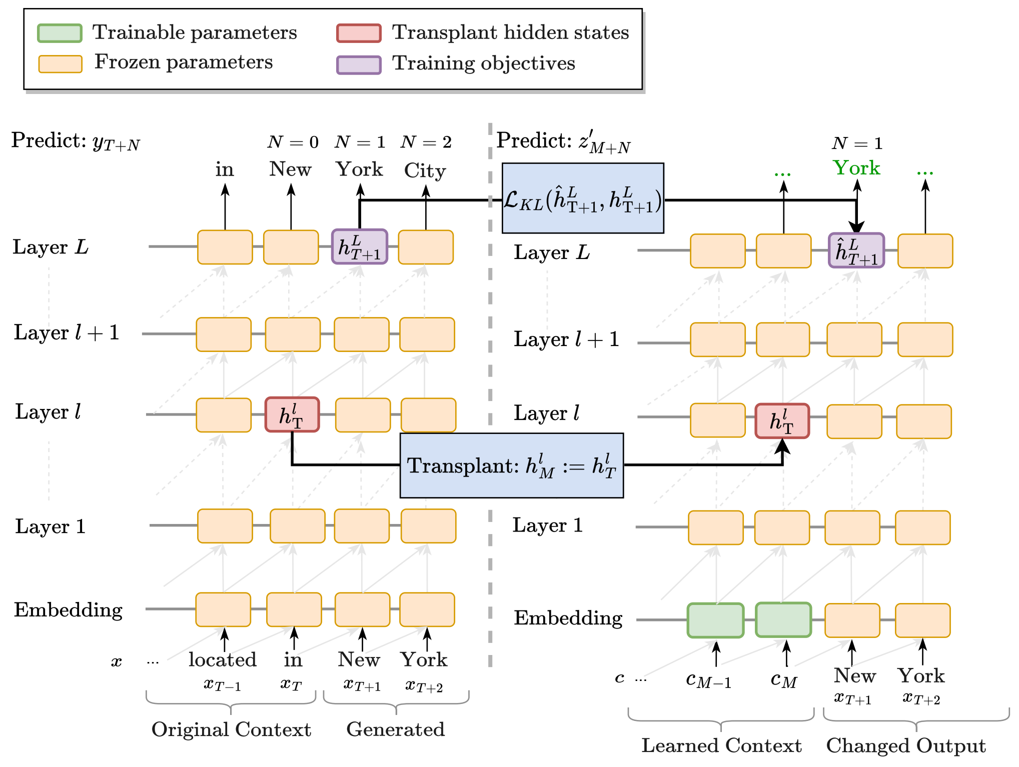

Figure 3: Learned context prompt Causal Intervention Overview. The left and right sides represent two different transformer model runs. The general setup is the same as Figure 2. The difference lies in the context provided in the transformer run on the right hand side. Instead of manually thinking of a context, we provide a learned context to increase the tendency of decoding the subsequent tokens predicted by h_T^l. We do so by training the context, c, with L_{KL} criterion and the objective to match the subsequent token prediction, such as 'York' in this instance.

the hidden state. For each decoder layer l, we train the corresponding prefix $c_{\text{opt}}^{(l)} = [c_1^{(l)}, ..., c_M^{(l)}]$ to maximize the probability of the model yielding the exact subsequent phrase after the original context. In particular, we conduct the same causal intervention in the hidden states h_T^l. We then optimize the probability distribution of the subsequent generation under the learned context to be the same as the original model when all its previous generation is given correctly:

$$\operatorname{argmin} \mathrm{KL}(\hat{y}_{M+N}\,;\,y_{T+N}) \tag{10}$$

Where the predicted distribution $\hat{y}_n$ is given using the same intervention as described in Eq. 9:

$$\hat{y}_{M+n} = G([c_1, ..., c_M, x_{T+1}, .., x_{T+N}] \\ ||\, h_M^l := h_T^l) \tag{11}$$

We hence optimize this objective with the model frozen and only prefix left to be trained. Notably, our approach is different from the implementation of prefix tuning (Li and Liang, 2021) in the sense that we back-propagate the gradient through the model instead of a temporary MLP, as empirically it produces a significantly better optimized context.

3 Experiments and Results

3.1 Data

We perform evaluation on samples of the Pile (Gao et al., 2020), which is the 825GB dataset used to train GPT-J-6B (Wang and Komatsuzaki, 2021) as well as other LLMs.

To train the linear models, we sample 100,000 tokens that have an average of 518 sized-context. Amongst the 100,000 token samples, we use 10,000 of them to train for our learned prompt experiment. For testing our methods, we sample another 1000 tokens that have an average previous context length of 535. To simplify our analysis of the degree to which single hidden token representations encode subsequent n-grams, we draw our samples from contexts in which the original transformer model made a correct prediction.

More specifically, we randomly sampled train and test data points from the subset of token locations where the autoregressive transformer under consideration correctly predicts the following token. In Table 1, we break down the types of tokens present in the testing data by categorizing the last token (T) of the prefix as well as the generated tokens outputs of GPT-J , through greedy (argmax) decoding, at $N = 0, 1, 2, 3$ with respect to various properties, such as whether they are lower-cased tokens that start with a space, or are numerical tokens, and so on.

3.2 Evaluation Metrics

For evaluation we adopt the same metrics used in prior related work Din et al. (2023), namely Precision@k and Surprisal.

Precision@k measures the appearance of the top probability token in the output at N tokens ahead we predict from the hidden state with respect to the observed top-k tokens from GPT-J-6B model output. Higher values are better here because these mean the actual token at the corresponding future token was accurately predicted.

Surprisal, on the other hand, is the minus log probability according to the GPT-J-6B model output of the highest probability token according to the proposed probing methods. Lower is better for this measure. because such values imply that the top predicted tokens are deemed probable by the model.

3.3 Experimental Setup

Linear Model We train two types of linear models — one with an output space of 4096 (the hidden representation size used by GPT-J-6B), and the other one with 50,400 (the vocabulary space of the same). GPT-J-6B comprises 28 layers. We train 4 instances for each of these layers, one for each different "future" token position we consider ($n = 0, 1, 2, 3$). As input we accept the source hidden state, i.e., h_T^l. Our output is either the hidden state, i.e., h_{T+N}^L or the decoded output at the position (vocabulary distribution) $T + N$.

Fixed Prompt Causal Intervention This is an evaluation-only setup where we choose four generic context prompts and perform causal intervention on these contexts as shown in Figure 2. The four fixed context prompts that we test are:

- `Hello! Could you please tell me more about "`
- `The multi-tokens present here are "`
- `The concepts in this hidden state listed are: (`
- `<|endoftext|> This state is describing about the following concept:`

The hidden states are gathered from layer l of the last token of the context tokens and are transplanted into the hidden representation of the last token in the generic prompts at the same layer l.

Properties	**Last Original Context Token**	**N = 0**	**N = 1**	**N = 2**	**N = 3**	**Examples**
Lowercase No Space	12	14.5	18.1	13.1	13.4	'itability', 'aka', 'ension'
Lowercase With Space	42	39.1	37.1	38.4	36.7	' sense', ' tests', ' punitive'
Uppercase No Space	2.4	2.7	2.2	2.8	1.6	'V', 'TABLE', 'SE'
Uppercase With Space	1.9	2.4	1.1	1.5	1.7	' STAR', ' UK', ' USA'
Token length < 4	57.8	59.8	64.3	59.9	63.2	'*', 'ate', '</'
Token length ≥ 4	42.2	40.2	35.9	40.5	37	' validation', ' Subaru', 'ulsion'
Punctuation	15.7	14.5	17.3	15.2	19	'-', '.', '</'
Numerical	2.4	2.7	1.9	3.2	2.8	'1998', '001', '5'

Table 1: Data Frequency of different token properties on the Last Prefix Tokens and GPT outputs at N=0,1,2,3. Each number in the table is a percentage of the test dataset, which is of size 1000.

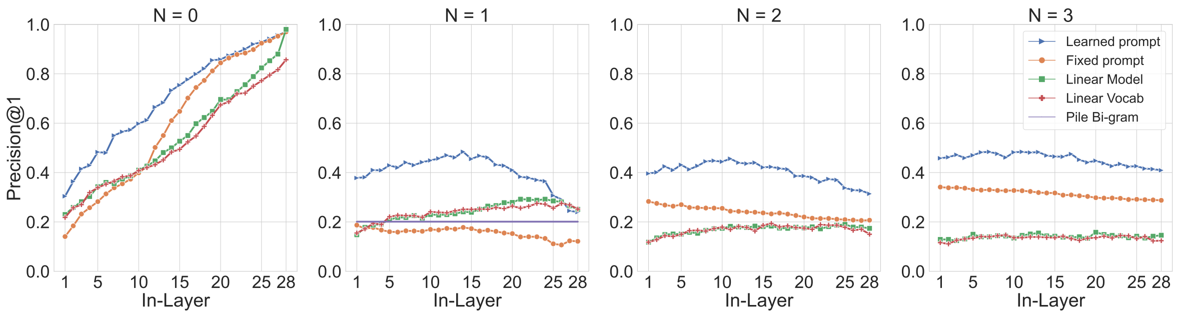

Figure 4: Accuracy (Precision@1) using the transplanted hidden representation. The $N = 0$ case models immediate next-token prediction, and $N \geq 1$ are the subsequent-token cases that are the focus of our work. The learned prompt is best able to recover future token information from hidden states of a preceding individual token, with predictive accuracy peaking at middle layers, with more than double the accuracy of a bigram baseline. A linear model predicting the hidden state fares comparably to predicting directly into the output vocabulary.

Learned Prompt Causal Intervention We then compare with trained prompts with the same token length as the fixed prompts. We train a soft prompt for each layer l from 1 to 28. Each learned prompt is trained by maximizing the probability of generating the token from the prefix context at the penultimate layer, when the hidden state is transplanted at layer l at the last token of the soft prompt, in the same way as the fixed prompts are applied. We train a prefix with a length of 10. This method performs best and is our main method.

3.4 Unveiling Subsequent Tokens

Figure 4 and Figure 5 illustrate the difference between our method and the baselines. The learned prompt optimized with the objective of predicting the next token (N=1) has the best performance. On average, the precision@1 is 24.8% higher, precision@5 is 25.3% higher, and precision@10 is 25.1% higher than the **best** baseline method. For the surprisal, the learned prompt also has the lowest value, which indicates its efficacy at maximally unveiling the information behind the hidden states.

4 Related Work

Knowledge Prediction and Manipulation Recent works have delved into LLM internals to better understand how such models predict the next token at each computation step. Geva *et al.* (2021), for instance, find that the feed-forward layers in transformers operate as key-value memories, allowing one to intervene at those layers to modify the next token output (Geva et al., 2022). Frameworks such as ROME (Meng et al., 2022a) and MEMIT (Meng et al., 2022b) scale such manipulations to edit knowledge in stored in LLMs.

The consensus that has emerged in these papers is that some early-middle and late layer calculations contribute the most to the final predicted token. Tools such as Logit lens (nostalgebraist, 2020) and Tuned lens (Belrose et al., 2023; Din et al., 2023) allow us to look at the top-k values of the transformer at *every* layer and token to see early next-token predictions. Katz and Belinkov (2023) used logit lens to visualize semantic information flow in GPT-2 models. In contrast to these approaches, we aim to characterize how the current

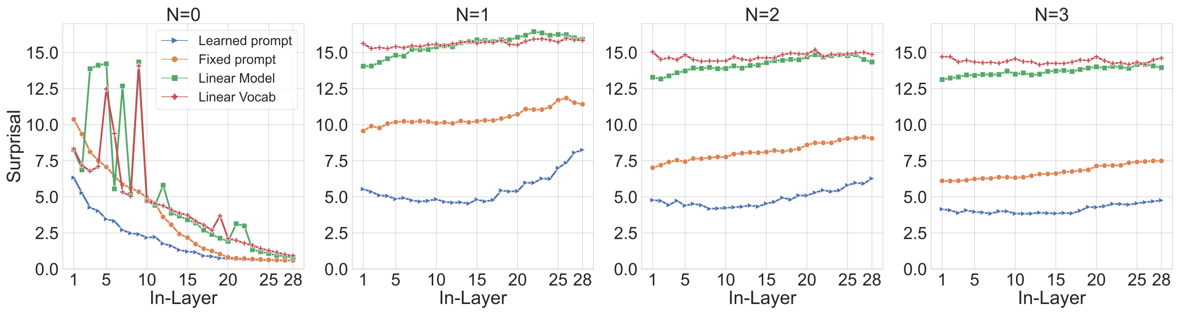

Figure 5: Average surprisal of the model after transplantation. Again the learned prompt performs best, confirming the presence of subsequent-token information encoded at middle-layer hidden states.

	LENS	N=1	N=2	N=3
Accuracy				
LEARNED	97.0	**48.4**	**43.7**	**46.9**
FIXED	97.0	20.8	30.0	36.5
HS	**98.0**	29.2	19.0	15.8
VOCAB	85.7	27.5	19.4	14.7
Surprisal				
LEARNED	**0.6**	**4.5**	**4.4**	**3.9**
FIXED	**0.6**	8.8	6.5	5.7
HS	0.8	14.1	13.2	13.1
VOCAB	0.9	15.3	14.4	14.2

Table 2: Best accuracy and surprisal results for each method. LEARNED refers to the Learned Prompt Causal Intervention Method; FIXED denotes the Fixed version. HS is the Linear Model variation that predicts Hidden State; VOCAB, is the Linear Model variation that predicts a distribution over the vocabulary directly.

hidden state would affect the prediction of not only the next token, but also tokens farther ahead.

Early Exit Decoding To optimize the running time and space requirements of training models, prior work has looked at "early exit" strategies, which usually involves stopping at earlier layers of computation and estimating the final predictions based on those computations (Schuster et al., 2022; Xin et al., 2021; Kong et al., 2022; Zhang and He, 2020; Din et al., 2023). The takeaway from these methods is that it is possible to achieve prediction performance comparable to that observed when all layers are used even when dropping a couple of computational layers for each token. For instance, Din and colleagues (2023) used linear transformations to predict a later layer's hidden representation from an earlier layer at the same token. This approach was able to preserve $\sim$95% of the full transformer model outputs on GPT-2 (Radford et al., 2019) and BERT (Devlin et al., 2018). This result implies that initial model layers encode information that to a large degree determines the final output. In this work we test the limits of this phenomenon by evaluating the degree to which a single hidden representation for the token at position T can be used to predict tokens multiple steps ahead (i.e., at $T+N$).

Memorization in Language Models Due to the potentially sensitive information present in the datasets used to train language models (LMs), past work has investigated what, when, and why memorization occurs (Carlini et al., 2021, 2019; Feldman and Zhang, 2020; Lehman et al., 2021), how memorization changes as a function of training data size (Carlini et al., 2023; Wei et al., 2022), and how other memorized information can be detected based on model internal states (Haviv et al., 2023).

These works have collectively illustrated that there are some text snippets that LMs remember and can output verbatim or in closely paraphrased versions ("approximate memorization"; Ippolito et al. 2023). Other work (Haviv et al., 2023) has shown that earlier layers of models tend to promote memorized concepts or tokens, while later layers boost model confidence in these tokens. Our paper can be viewed as an extension of this work on investigating memorization of multi-token phrases: we ask whether and to what extent a single model hidden state encodes multi-token information.

Prompt Tuning Prompt Tuning has emerged as a parameter-efficient method for fitting LMs for new downstream tasks. By freezing the LM and optimizing only the soft prompt parameters, models are able to achieve performance comparableto that observed after fine-tuning all parameters. Li *et al.* (2021) introduced prefix tuning which entailed training plug-and-play prefix that

Last Context Token Type	**Linear: Vocab Space**	**Linear: Hidden State**	**Fixed Context**	**Learned Context**
Lowercase No Space	21.7	25.2	9.2	**32.5**
Lowercase With Space	26.4	20.8	19.2	**51.9**
Uppercase No Space	**29.2**	26.3	0.0	**23.3**
Uppercase With Space	26.3	26.3	10.5	**31.6**
Token length < 4	26.5	24.9	21.8	**46.9**
Token length ≥ 4	23.9	24.4	18.0	**52.1**
Punctuation	28.7	28.7	16.6	**47.8**
Numerical	12.5	16.7	20.8	**33.3**

Table 3: Accuracy of predicting $N = 1$ token ahead (y_{T+1}, which predicts x_{T+2}) based on hidden representation of the last context token(x_T). Results are shown for layer $l = 14$, where the learned prompt model is most accurate.

steers the behavior of the LMs for the downstream tasks. Other work (Wallace et al., 2019) applied a gradient-based method to search for the best discrete prompts which enable the model to produce desire generation. Sun and colleagues (2023) train the prefix soft prompt as a way of aligning semantically equivalent instructions in latent space.

5 Discussion

In this paper we explored the degree to which we are able to decode multi-token outputs subsequent to a particular token on the basis of its hidden representation alone. The results in Table 2 and Figures 4 and 5 indicate that such representations encode such information, at least to some degree. Among the decoding methods we assessed, learned prompts are best able to predict such future tokens. Both the linear and the learned prompt models achieve better accuracy than the empirical bigram baseline at $N = 1$ (the horizontal line in Figure 4).[1] When this bigram model is run on the testing data, it achieves 20.1% accuracy. Interestingly, predictive accuracy of the learned prompt model peaks at the middle-layer hidden states, suggesting that subsequent-token information is encoded at those middle layers; this pattern is very different from the immediate next-token $N = 0$, in which accuracy peaks at the last layer.

The learned prompt model realizes an accuracy sufficiently good to be potentially useful as a 'Logit lens'-like tool to provide insights about subsequent token information contained in hidden states within LLMs. This provides a way to decode a short sequence of tokens encoded in a hidden state, rather than only the single immediate token prediction.

[1]The bigram baseline is collected from 900,000 documents from the Pile dataset.

To further explore the contexts in which these methods seem better (or worse) able to predict subsequent tokens, we categorize input token (the last original context token) into eight (non-mutually exclusive) categories, shown in Table 3. We report the model accuracies when using layer 14, where the learned prompt model peaks.

While all categories of token types are predicted better by the learned prompt than by the linear model, the relative improvement is highest when the last context token is a lowercase token preceded by a space, or a longer token. This suggests that information about how to complete long words may not be immediately accessible by a linear model decoder, but that they can be made accessible by using the parameters of the pretrained model as done by the learned prompt intervention method.

We have also observed that the accuracy of predicting subsequent tokens is correlates with the model's confidence in its next token prediction. In the case of $N = 1$, for instance, the learned prompt intervention method's calibrated accuracy is 26%, 57%, 77%, and 95% for model confidence groups of 0-30%, 30-60%, 60-90%, and 90%-100%, respectively. These trends appear in $N = 2$ and $N = 3$ as well. This suggests that we might gainfully use this decoding method as a probing tool, trusting that predicted future tokens are generally accurate when the model is confident.

Does future information appear only in the presence of higher-level concepts? For example, one might hypothesize that in cases the language model predicts an entire named entity, that the probing method might decode future predictions more accurately. To investigate this, we performed sub-group analyses on test results to characterize how well the best probing method performed specifically for multi-token named entities. Interestingly, we found

	Mart	y	Mc	Fly	from
L1	inez the court held	erson the screen.	Afee the same source	er and the other	behindrrhaph
L2	inez\n\nThe	\xe9n-1-	Afee and the other	er the left of	Sons\n\nThe
L3	.\n\nThe	Friedman and the other	Afee\n	er\n\n\n	Havanaa:\n
	\n de la c	Barn and the other	Lean.\n\n	er\ufffd\ufffd said	1992a and the
	\n de la c	ring and the other	Leanmig.	er \ufffd\ufffdI	Oklahoma first time I
	\n de la c	ring and the next	Afee\n\nThe	world and the future	Austria book.\n
	\n de la c	ring and the other	Leanaway from the	mer 1, 1	Australia movie.\n
	\n, and the	ell and the other	Lean\n\nThe	walker the time of	England first time he
	\n, and the	ellLean, and	Lean\n\nThe	walker be the first	Australia movie "The
	\n, and the	ellDonough,	Lean\n\nThe	te Marty McFly	Australia movie "The
	\n"\n	ellDonough,	Lean\n\nThe	te Marty McFly	Vietnam movie "The
	\n" id="	GreenbergDonough,	Bride\n\nThe	te Marty McFly	Germany movie "The
	\n" id="	GreenbergDonough,	Lean\n\nThe	movies Marty McFly	Boston movie "Back
	\n" id="	ellDonough,	Bride\n\nThe	movie Marty McFly	movie movie \ufffd\ufffd
	\n" id="	riumDonough,	Bride\n\nThe	movie Marty McFly	movie movie "The
	\n" id="	WalshDonough,	Bride\n\nThe	movie Marty McFly	movie movie "The
	\n" id="	McDonough,	Bride\n\nThe	movie Marty McFly	movie Back to the
	\n" id="	McDonough,	Bride\n\nThe	movie Marty McFly	movie movie "Back
	\n" id="	riumDonough,	Flylew\n	movie Marty McFly	1980 Back to the
	\npng" alt	ring Marty, and	Fly\n\nThe	movie Marty McFly	movie Back to the
	\npng" alt	Mc Marty, and	Fly\n\nThe	Returns Marty McFly	movie Back to the
	\npng" alt	Mc\ufffd\ufffd he	Fly\n\nThe	arrives Marty McFly	movie Back to the
	\npng" alt	Mc he was a	Fly\n\nThe	arrives Marty McFly	1984 Back to the
	\npng" alt	Mc and I'm	Fly\n\nThe	arrives Marty McFly	1989 to the Future
	\npng" alt	Mc and I'm	Fly\n\nThe	's\nThe first	Back to the Future
L26	\n1.0	Mc and I'm	Fly\n\nThe	(\nThe first	Back movie "Back
L27	.\n\nThe	Mc and the other	Fly.\n\n	,\nThe first	the movie "Back
L28	y\n\nThe	Mc and I'm	Fly.\n\n	\n and the future	Back future.\n

Figure 6: The Future Lens applied to the hidden states of GPT-J-6B processing *Marty McFly from*. Each cell illustrates the most likely sequence of future tokens that the respective hidden state predicts. The darker boxes correspond to higher probabilities/confidence.

little difference: when examining just the named entity cases, we observe similar or slightly lower accuracy: 44%, 42% and 37% for $N = 1, 2, 3$, suggesting that future information is present broadly, not only for long entity names.

In sum, we have found that a single hidden state encodes information about outputs more than one token ahead, and we have demonstrated three different methods that can decode them for GPT-J-6B.

Application: Future Lens We apply the Learned Prompt Intervention Method to create a novel probing tool we call the *Future Lens*. Given a soft prompt, we perform the intervention using the states arising from the user's prompt to provide a view into what the hidden states encode about future tokens. In Figure 6, we show an example for the prompt: "Marty McFly from". The Future lens reports the anticipated four tokens from every hidden state in the model (across layers).

In the Future Lens visualization, every cell represents a hidden state from a particular layer ("L{digit}") at a specific token. The shade of each cell indicates the average confidence of the model with respect to the corresponding token predictions (darker shades indicate greater confidence). For example, at the cell representing the hidden state at Layer 25 at the token "from", we can see that the confidence in the predicted tokens "Back to the Future" is strong. This particular state suggests that the LLM already knows that Marty McFly is related to the Back to the Future movie. Interestingly, the model also assumes "Marty" to have the surname Donough. Returning to the predictions at token "from", we see that the early layers seem to first predict countries such as Australia or cities such as Boston. However, through future predictions, we can see the model begins to associate Marty McFly with a movie around Layer 6. Hence, through this tool, we can gain further insights about the model's chain of predictions at every hidden state. All code and data for demo and implementation is made available at: https://github.com/KoyenaPal/future-lens

Acknowledgements This work was supported by Open Philanthropy and by the National Science Foundation (NSF) award 1901117. We thank the Center for AI Safety (CAIS) for making computing capacity available for this research.

References

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2023. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *Proceedings of the 28th USENIX Conference on Security Symposium*, SEC'19, page 267–284, USA. USENIX Association.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. 2021. Extracting training data from large language models. In *USENIX Security Symposium*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Alexander Yom Din, Taelin Karidi, Leshem Choshen, and Mor Geva. 2023. Jump to conclusions: Short-cutting transformers with linear transformations.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Vitaly Feldman and Chiyuan Zhang. 2020. What neural networks memorize and why: Discovering the long tail via influence estimation. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The pile: An 800gb dataset of diverse text for language modeling.

Mor Geva, Avi Caciularu, Guy Dar, Paul Roit, Shoval Sadde, Micah Shlain, Bar Tamir, and Yoav Goldberg. 2022. LM-debugger: An interactive tool for inspection and intervention in transformer-based language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 12–21, Abu Dhabi, UAE. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. 2023. Finding neurons in a haystack: Case studies with sparse probing.

Adi Haviv, Ido Cohen, Jacob Gidron, Roei Schuster, Yoav Goldberg, and Mor Geva. 2023. Understanding transformer memorization recall through idioms. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 248–264, Dubrovnik, Croatia. Association for Computational Linguistics.

Daphne Ippolito, Florian Tramèr, Milad Nasr, Chiyuan Zhang, Matthew Jagielski, Katherine Lee, Christopher A. Choquette-Choo, and Nicholas Carlini. 2023. Preventing verbatim memorization in language models gives a false sense of privacy.

Michael I Jordan. 1997. Serial order: A parallel distributed processing approach. In *Advances in psychology*, volume 121, pages 471–495. Elsevier.

Shahar Katz and Yonatan Belinkov. 2023. Interpreting transformer's attention dynamic memory and visualizing the semantic information flow of gpt.

Jun Kong, Jin Wang, Liang-Chih Yu, and Xuejie Zhang. 2022. Accelerating inference for pretrained language models by unified multi-perspective early exiting. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4677–4686, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Eric Lehman, Sarthak Jain, Karl Pichotta, Yoav Goldberg, and Byron C Wallace. 2021. Does bert pretrained on clinical notes reveal sensitive data? *arXiv preprint arXiv:2104.07762*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 36.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.

nostalgebraist. 2020. interpreting gpt: the logit lens.

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

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems*.

Yixuan Su, Deng Cai, Yan Wang, David Vandyke, Simon Baker, Piji Li, and Nigel Collier. 2021. Non-autoregressive text generation with pre-trained language models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 234–243, Online. Association for Computational Linguistics.

Jiuding Sun, Chantal Shaib, and Byron C Wallace. 2023. Evaluating the zero-shot robustness of instruction-tuned language models. *arXiv preprint arXiv:2306.11270*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. *arXiv preprint arXiv:1908.07125*.

Ben Wang and Aran Komatsuzaki. 2021. Gpt-j-6b: A 6 billion parameter autoregressive language model.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

Yisheng Xiao, Lijun Wu, Junliang Guo, Juntao Li, Min Zhang, Tao Qin, and Tie yan Liu. 2023. A survey on non-autoregressive generation for neural machine translation and beyond.

Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021. BERxiT: Early exiting for BERT with better fine-tuning and extension to regression. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 91–104, Online. Association for Computational Linguistics.

Minjia Zhang and Yuxiong He. 2020. Accelerating training of transformer-based language models with progressive layer dropping.

A Appendix

Additional Figures

In this main paper, we report results based on models that are trained to optimize the $N = 1$ single token-ahead prediction, and we test those models for predictive accuracy for other N.

The same methods can also be used to optimize subsequent tokens, and the results of those methods are shown here. We find that optimizing for $N = 1$ works best and generalizes surprisingly well to other N, but that that optimizing for other N does not perform well for $N = 1$.

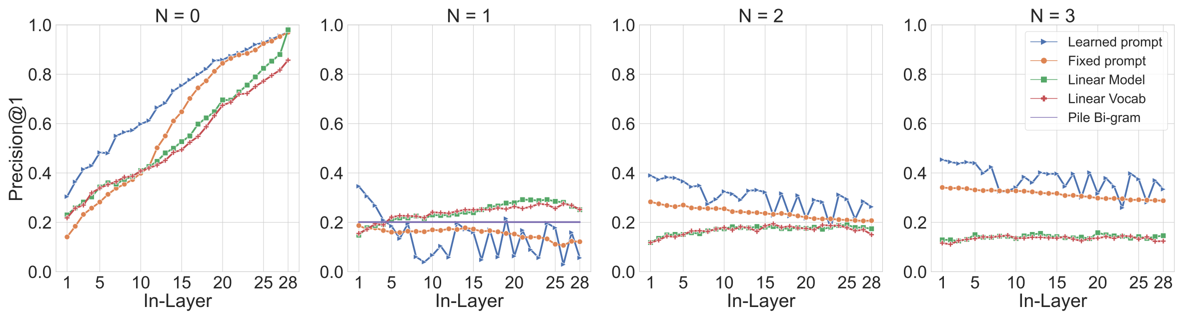

Figure 7: The Precision@1 (Accuracy) of all the methods trained with predicting the currently decoded token (teacher-forcing)

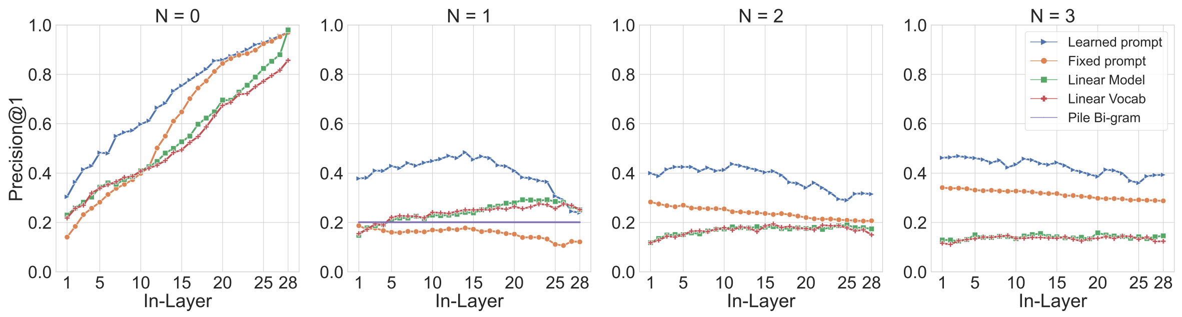

Figure 8: The Precision@1 (Accuracy) of all the methods trained with predicting the 1st next token

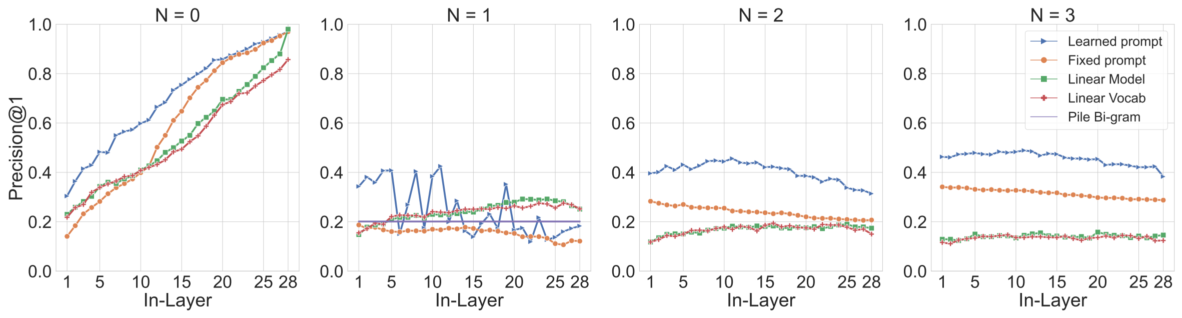

Figure 9: The Precision@1 (Accuracy) of all the methods trained with predicting the 2nd next token

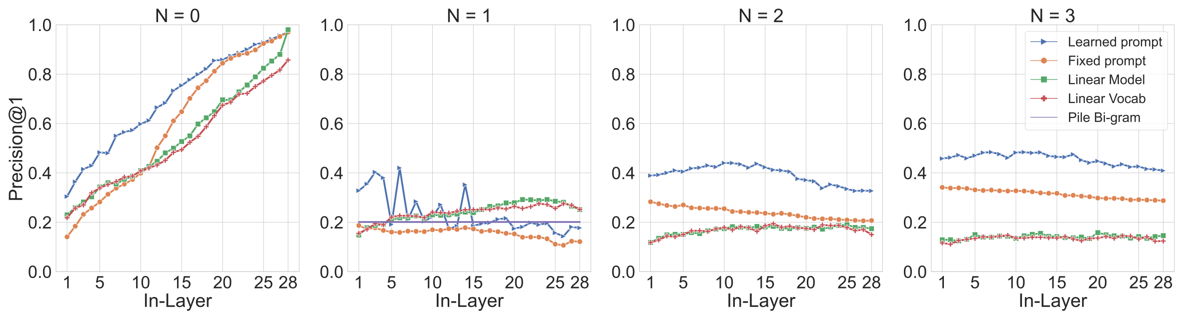

Figure 10: The Precision@1 (Accuracy) of all the methods trained with predicting the 3rd next token

Limitations

In our exploration with extracting far future tokens from single hidden states, we have mostly trained and tested on English data whose size, 100,000, is still relatively small compared to the data size that GPT-J-6B was actually trained in. Furthermore, the experiments were only conducted in GPT-J-6B. While the presence of subsequent token information in a single hidden state is evident in this model, it would be more comprehensive to run these experiments in other LLMs. Since there are no specific prior works that focused on decoding far future tokens from a single hidden state, we did not have any prior baselines we would refer to. While we did create a bigram baseline in the case of predicting 2 tokens in the future ($N = 1$) and also create linear models as a first decoding method, there could be baselines with other architectures like Recurrent Neural Networks (Jordan, 1997; Elman, 1990) and Non-Autoregressive generation (Su et al., 2021; Xiao et al., 2023). Lastly, our experiments were up to 4 tokens in the future, i.e., $N = 0, 1, 2, 3$. It would be intriguing to scale and test up to how many tokens in the future does a single state actually encode and predict.