How Do LLMs Use Their Depth?

Akshat Gupta[1], **Jay Yeung**[1], **Gopala Anumanchipalli**[1], **Anna Ivanova**[2]
[1]University of California, Berkeley, [2]Georgia Institute of Technology
akshat.gupta@berkeley.edu, a.ivanova@gatech.edu

Abstract

Growing evidence suggests that large language models do not use their depth uniformly, yet we still lack a fine-grained understanding of their layer-wise prediction dynamics. In this paper, we trace the intermediate representations of several open-weight models during inference and reveal a structured and nuanced use of depth. Specifically, we propose a "*Guess-then-Refine*" framework that explains how LLMs internally structure their computations to make predictions. We first show that the top-ranked predictions in early LLM layers are composed primarily of high-frequency tokens, which act as statistical guesses proposed by the model due to the lack of contextual information. As contextual information develops deeper into the model, these initial guesses get refined into contextually appropriate tokens. We then examine the dynamic usage of layer depth through three case studies. (i) Multiple-choice task analysis shows that the model identifies appropriate options within the first half of the model and finalizes the response in the latter half. (ii) Fact recall task analysis shows that in a multi-token answer, the first token requires more computational depth than the rest. (iii) Part-of-speech analysis shows that function words are, on average, the earliest to be predicted correctly. To validate our results, we supplement probe-based analyses with causal manipulations in the form of activation patching and early-exiting experiments. Together, our results provide a detailed view of depth usage in LLMs, shedding light on the layer-by-layer computations that underlie successful predictions and providing insights for future works to improve computational efficiency in transformer-based models.

1 Introduction

Despite the remarkable performance of large language models (LLMs), their internal computations remain poorly understood. One critical question is: how do LLMs internally structure their computations during inference and use their depth layer-by-layer to arrive at predictions? Are specific token predictions always computed at the last layer or does the model settle on predictable tokens early on and simply propagate these predictions? These questions have implications both for interpreting the internal computations of these models and for building more efficient LLM that can use their compute dynamically.

We present a "*Guess-then-Refine*" framework to answer these questions and understand model computations across layers. Under this framework, we show that models first create early guesses that get refined in subsequent layers. We first show that early-layer predictions in LLMs are largely composed of high-frequency tokens, like determiners or prepositions. For example, for Pythia6.9B, while the Top-10 most frequent tokens account for about 34% of the final predictions, the top-1 ranked early layer predictions contain the Top-10 tokens more than 75% times. We call this "*Frequency-Conditioned Onset*", where LLMs tend to use corpus statistics while making candidate proposals in early layers. We then show that these early layer proposals are guesses which get heavily refined in subsequent layers, where more than 80% of the early layer predictions get refined into into contextually appropriate generations by the final layer. We leverage TunedLens (Belrose et al., 2023) to quantify token prediction patterns across LLM layers.

We also show that LLMs use their depth dynamically based on the task at hand through three case studies. (i) First, we analyze the layer-wise prediction dynamics when the models perform downstream tasks with a constrained set of option choices, such as answering multiple-choice (MCQ) or True/False questions. We uncover that the model performs this task in two steps with a clear *phase transition*: it first collects the valid option choices as top-ranked predictions by the middle layers, and

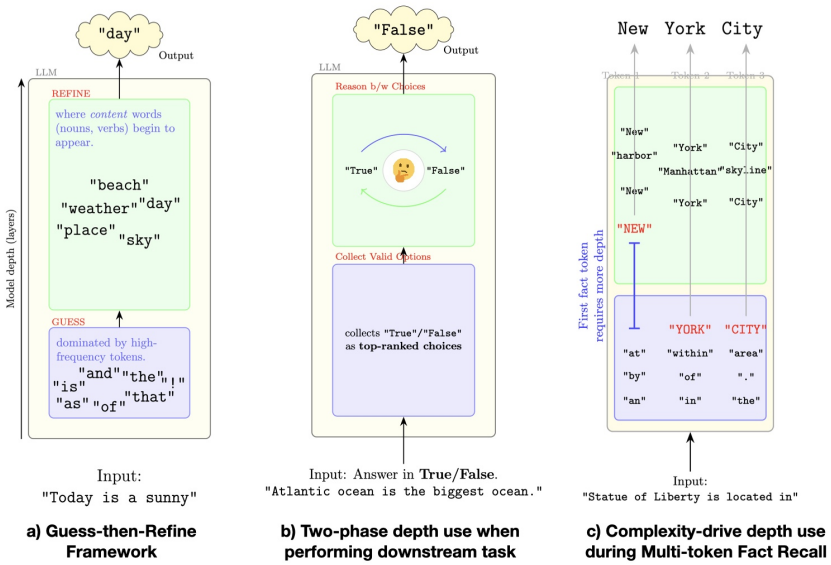


Figure 1: Illustrative depictions of our *Guess-then-Refine* framework and complexity-aware depth use in LLMs. **(a)** Early layers elevate high-frequency tokens as statistical guesses, which deeper layers refine into context-appropriate predictions (*Frequency-Conditioned Onset*). **(b)** Option-constrained downstream tasks (e.g., True/False): early layers collect valid options into the top ranks, while later layers reason between top choices to generate final answer. **(c)** Multi-token fact recall: the first token of the answer requires the largest depth on average, while subsequent tokens appear at much shallower depths. **Overall:** LLMs act as *early statistical guessers and late contextual integrators*, exhibiting *complexity-driven depth use*.

in later layers the model deliberates between the valid options to reach the final answer. We also accompany this case study with activation patching experiment showing causal evidence of a clear phase transition happening as the model performs such downstream tasks. (ii) We then look at multi-token fact recall task, where the model is asked factual questions whose answers span multiple tokens. We see that the second and third answer token predictions appear much earlier than the first, indicating a higher computational load associated with selecting the initial response direction. (iii) Finally, we show that during regular next token prediction, easier to predict tokens corresponding to function words and punctuation begin to get predicted correctly by early-to-mid LLM layers, whereas harder to predict tokens like nouns and verbs appear much later. The above two case studies are also accompanied by early exiting experiments. Together, these experiments are indicative of *Complexity-Driven Depth Use*, where easier tasks require fewer layers of computation.

We perform this analysis on four open-weight models - GPT2-XL (Radford et al., 2019), Pythia-6.9B (Biderman et al., 2023), Llama2-7B (Touvron et al., 2023) and Llama3-8B (Meta, 2024). The experiments are done using the TunedLens probe probe, which allows us to decode earlier layer representations with higher fidelity. We also provide independent evidence with causal ablations and early exiting experiments to confirm our findings. Addionally, we also perform ablations on TunedLens to confirm that our results represent information content in early layers.

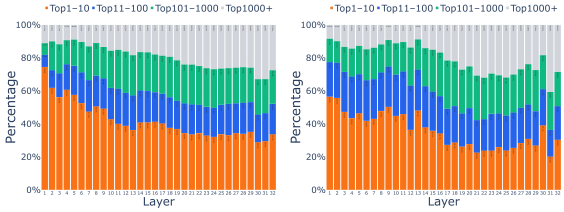
2 Background and Related Work

Intermediate activations in LLMs can be projected onto the vocabulary space using the unembedding matrix (Nostalgebraist, 2020). Let h^l represent the hidden representation after layer l. LogitLens (Nostalgebraist, 2020) projects these representations in the token space as follows:

$$\texttt{LogitLens}(h^l) = W_U\Big[\texttt{Norm}_f[h^l]\Big] \qquad (1)$$

where $\texttt{Norm}_f$ represents the final normalization applied before the application of the unembedding matrix $W_U \in R^{d\times|V|}$; where $|V|$ represents the vocabulary size, and d is the hidden dimension size of the model.

Prior work has shown that directly applying the unembedding matrix to intermediate layers can lead to unreliable results as intermediate features may operate in different subspaces compared to the final layer outputs (Din et al., 2023; Belrose et al.,



(a) Pythia6.9B (b) Llama3-8B

Figure 2: **Frequent tokens dominate early-layer predictions.** At each layer, the layer's top-1 ranked token is grouped by corpus-frequency buckets (Top1–10, Top11–100, Top101–1000, Top1000+). Early layers are dominated by Top1–10 bucket tokens, where LLMs make statistical guesses and propose high-frequency tokens as top-ranked candidates under limited information, while deeper layers increasingly replace them with rarer, context-appropriate tokens.

2023). We therefore opt to use TunedLens for more robust results. TunedLens learns an affine mapping between the output representations of each layer and the final unembedding matrix in the form of *translators* ($A_l \in R^{d \times d}, b_l \in R^d$), as shown below:

$$\texttt{TunedLens}(h^l) = \texttt{LogitLens}(A_l h^l + b_l) \quad (2)$$

The TunedLens probes are trained to minimize the KL-divergence between the final layer and an intermediate layer probability distribution. Thus, TunedLens translators (A_ℓ, b_ℓ) are trained to allow for better transferability between intermediate layers and the final unembedding matrix, which provide more faithful decoding of intermediate representations of the model, especially in the early layers (Belrose et al., 2023).

We provide a more detailed survey of background work including specific comparisons in Appendix A.

3 Frequency-Conditioned Onset in Early LLM Layers

We begin by studying the top-ranked predictions across LLM layers during next token prediction using TunedLens. We look at four open-weight models - GPT2-XL (Radford et al., 2019), Pythia-6.9B (Biderman et al., 2023), Llama2-7B (Touvron et al., 2023) and Llama3-8B (Meta, 2024). We use the TunedLens probes provided by the original authors for these models (AlignmentResearch, 2026).

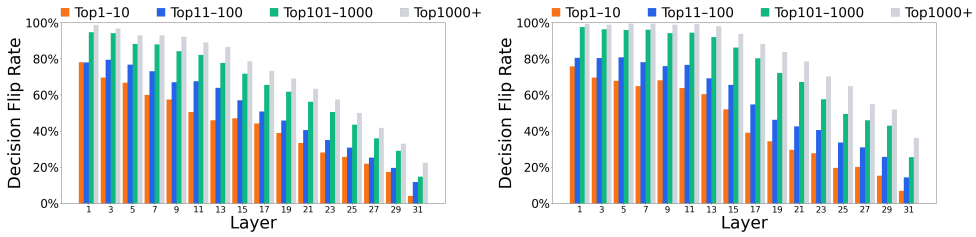
Methodology: We divide vocabulary tokens of each model into four buckets by their frequency of occurrence. For this, we take the tokenizer of each model and tokenize the English Wikipedia dataset (Lhoest et al., 2021), and record the frequency of each token. We use the following four buckets - (i) `Top1-10` bucket, which contains the top 10 most frequent tokens in a corpus and accounts for approximately 23-25% of the corpus depending on the model, (ii) `Top11-100` bucket, which contains the next 90 most frequent tokens and accounts for the next 16-20% of corpus tokens, (iii) `Top101-1000` bucket, which contains the next 900 most frequent tokens and accounts for 16-21% of corpus tokens, and finally the (iv) `Top1000+` bucket, which contains the remaining tokens and accounts for 35-40% of the corpus.

We evaluate the model during the next-token prediction task, where the base model is provided with prefixes extracted from the English Wikipedia corpus, cut-off at random points, and track the top-ranked token at each intermediate layer as the model predicts the next token. These prefixes have a variable lengths and can span multiple sentences (see Appendix B for details).

3.1 LLMs are Early Statistical Guessers

The top-ranked predictions at each intermediate layer classified into corresponding frequency buckets are shown in Figure 2. **We discover that early layer predictions are heavily dominated by tokens belonging to the `Top1-10` bucket**. For example, more than 75% top-ranked tokens for Pythia-6.9B (Figure 2a) belong to the `Top1-10` bucket at the first layer, while only 33% of the final predictions eventually made by the model lie in the `Top1-10` bucket. A similar trend can be seen for Llama3-8B (Figure 2b) and GPT2-XL and Llama2-7B, as shown in Appendix C.

This behavior can be explained by understanding the amount of information available in early layers when compared to deeper layers. In early layers, the model has incomplete contextual information about the input sentence, since the input has not been through enough attention layers to aggregate this information. Additionally, in early layers, the model is also unable to access the factual knowledge stored in its parameters, since the stored knowledge lies in the *middle* MLP layers (Geva et al., 2020; Meng et al., 2022; Geva et al., 2023). In the absence of enough contextual information and learnt knowledge, the best strategy is to



(a) Pythia-6.9B

(b) Llama3-8B

Figure 3: **Early-layer predictions are heavily revised.** For each layer and frequency bucket, we show the fraction of top-1 ranked predictions that are overturned by the final layer ("decision flip rate"). We see that about 80% of layer-1 Top1–10 predictions change by the end, and this rises to about 100% for Top101–1000 and Top1000+ bucket tokens. As expected, flip rates drop with depth.

revert back to corpus statistics. To explain this, let us take the example of an extreme scenario where we do not have access to any information about the input prefix. In such a situation, if one had to guess the next token, picking a token from the `Top1-10` bucket, which accounts for about 25% of the corpus, would be the best strategy as it maximizes the chances of being correct with a minimum set of tokens. Such an intuitive strategy naturally emerges out of the optimization pressures of the pre-training process!

Thus, LLMs make statistical guesses by promoting the most frequent tokens as top-ranked proposals in early layers. As contextual information develops deeper into the model, these guesses based on corpus statistics get refined into appropriate predictions, which is when lower frequency tokens begin to appear.

3.2 Massive Contextual Refinement in Later Layers

In the previous section, we show that early LLM layers propose candidates based on corpus frequency. In this section, we show that early layer predictions are in fact *guesses*, and as contextual information builds up as we go deeper into the model, these early suggestions get *heavily* refined into contextually appropriate tokens.

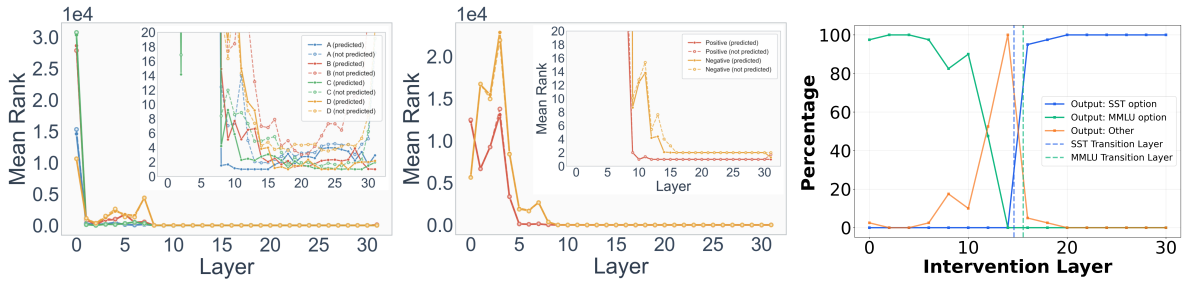
Indications for later layer refinements can already be seen in Figure 2, where the top-ranked tokens in early layers belong to the `Top1000+` bucket for only about 10% of the prefixes, a number that increases by around 300% by the final layer. However, here we consider a more specific metric for refinement. We define *refinement* with the "decision flip rate", which measures the percentage of top-1 ranked tokens at an intermediate layer l that are not the same as the final prediction.

Figure 3 shows a clearer picture of the magnitude with which early layer proposals in LLMs get refined to final predictions. For Pythia-6.9B (Figure 3a), while early layer decisions are dominated by the `Top1-10` bucket tokens as shown in previous section, almost 80% of those predictions get modified by the final layer. This is also true for Llama3-8B (Figure 3b). Additionally, if a prediction at layer-1 lies in the `Top1000+` bucket, it gets modified by the final layer with almost 100% probability. Results for the remaining models can be found in Appendix C. **This shows that early layer proposals get modified in overwhelming numbers by the time they reach the final layer** and indicates that refinement happens for both high- and low-frequency tokens.

Thus, early LLM layers propose frequency-conditioned guesses that are highly non-permanent, where approximately 60–80% of early top-ranked guesses are eventually replaced. On the other hand, permanence rises with depth along with context integration.

4 Complexity-Driven Depth Use in LLMs

In the previous section, we saw that LLMs propose initial guesses in early layers based on corpus statistics, which get refined in large numbers as more contextual information develops through the model. In this section, we go into more details of this guess-then-refine process across layers. Specifically, we show that model depth in LLMs gets used flexibly, where easier to execute tasks (or subtasks) finish at shallower depths, while more complicated tasks are left for later layers. We show this using three case studies: fixed-option question answering, fact recall and next-token prediction. We also perform causal ablations and early-exiting experiments to validate our findings.



(a) MMLU (b) SST (c) Activation Patching

Figure 4: **Mean rank of answer tokens during constrained-choice downstream tasks.** As the model performs tasks like MMLU (response options: ABCD) and sentiment analysis (response options: positive/negative), we track the mean rank of the options through the layers. Early layers of the model promote valid options choices as top-ranked candidates, whereas later layers reason between option choices (shown as zoomed-in figures in the top-right corner of each plot). This figure shows results for Pythia-6.9B. Results for other models and tasks follow similar patterns and can be found in Appendix D.1. (c) **Activation patching experiments**, where we take activations from an SST forward pass and patch it during the forward pass of MMLU. This figure shows in early layers, the model is still able to find one of MMLU options (A/B/C/D), whereas in later layers the generated answer is an SST option (positive negative). This shows causal evidence of the two phase behavior shown in subfigure (a) and (b). Additionaly, the phase-transition layers are predictions made by TunedLens, which very cleanly match actual model behavior.

4.1 Case Study I: Depth Use during Downstream Tasks

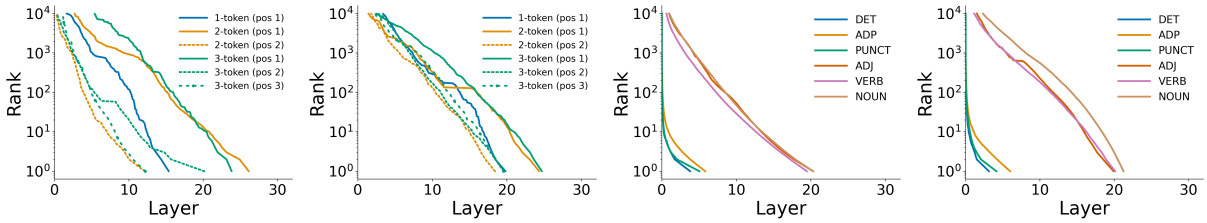
Dataset and Methodology: In this case study, we evaluate the internal prediction dynamics when the model performs four fixed-option downstream tasks - MMLU (Hendrycks et al., 2020), Sentiment Analysis (SST) (Socher et al., 2013), Natural Language Inference (NLI) (Dagan et al., 2005) and Paraphrase Dectection (MRPC) (Dolan and Brockett, 2005). We use a 4-shot set up, where the model is given four examples and is then asked to generate the answer. We then track the mean rank of the generated answer through the different layers of the model, where the intermediate logits are created using TunedLens.

Results: The results can be seen in Figure 4. We see a clear two-step prediction pattern. In the early layers, the model identifies the valid option choices and gathers them within the top ranks of intermediate layer logits (option-collection). As a result, the early layers see massive decrease in the ranks of valid option choices for MMLU and SST datasets respectively. This step usually happens within the first half of the model. After the valid options are collected as top-ranked choices, the model reasons through the options in the remainder of the layers (option-reasoning). This can be seen within the zoomed-in figures on the top-right corners of Figures 4a and 4b, where we zoom-in on the y-axis. For the MMLU task, the model constantly switches its top prediction, with some mid-layer bias for option A. For the sentiment task, the model defaults to "positive" as the label of choice all the way until the last layer on average. Similar option biases can be seen in other models (Appendix D.1), with extreme biases in smaller models (GPT2-XL).

Overall, we see that the model acts in accordance with complexity-driven use principle, and performs the easier of the two tasks of collecting the valid option in the early layers, while it leaves reasoning to deeper layers.

4.1.1 Causal Intervention by Activation Patching

Setup: We perform activation patching experiments (Meng et al., 2022) to validate our findings. We consider two tasks, MMLU and SST, where SST is the source task and MMLU is the target task. We take the activation vector corresponding to the generated token position when processing the source task (SST) at layer l and patch it during the forward pass of the target task (MMLU) at the same layer. We then look at the response generated during the target task (MMLU). The expectation is that if there are two distinct phases of option-collection and option-reasoning, then transfer of activation vectors during the option-collection phase should still give the model time to collect valid options, whereas transfer of activation vector during option-reasoning phase should generate one of the two options belonging to the SST categories even when geenrating the answer to an MMLU question.



(a) Pythia-6.9B (Fact) (b) Llama3-8B (Fact) (c) Pythia-6.9B (POS) (d) Llama3-8B (POS)

Figure 5: **Earliest crossing thresholds of predicted tokens for multi-token facts (top) and POS (bottom).** We determine which token the model predicts at the final layer and then track the earliest layer (x axis) at which the TunedLens rank of that token crosses a given threshold (y axis). **(a, b)** Case Study II: We plot tokenwise results for fact recall responses that consist of 1 token, 2 tokens, or 3 tokens. E.g., 3-token facts consist of 3 tokens at positions (pos) 1, 2, and 3. In both models, the first tokens of multi-token facts appear much later than subsequent tokens. **(c, d)** Case Study III: When predicted tokens are grouped by POS category, tokens corresponding to function words (DET, ADP) and punctuation appear much earlier than content words (ADJ, VERB, NOUN).

Results: The results are shown in Figure 4c, where the model answers an MMLU question and receives an activation patch from an SST question. The y-axis of the figure measures percentage of times the predicted answer belongs to either one of the MMLU options (green line), or one of the SST options (blue line), or neither (yellow line). This plot shows replacement of activations for Pythia 6.9B. We see that if we replace the activation from SST to MMLU processing in early layers, which is the option-collection phase, the model is still able to collect the MMLU options to top ranks and produce one of the MMLU options as the final answer. However, if this transfer happens beyond the transition layers, which is the option-reasoning phase, the output belongs to one of the SST options. **This experiment clearly demonstrates the presence of a two-phase answering algorithm followed by the model.** Note that this activation replacement experiment is done independent of the TunedLens probe. Results for other models show similar outcomes as shown in Figure 19.

We also define *phase transition layers* between the two phases using the TunedLens probe, which is the layer at which the model transitions from option-collection to option-reasoning mode. We define transition at the earliest layer at which *all* the top ranked tokens choices using the TunedLens probe belong to the available options for the task. This means that for MMLU, the top four ranked tokens should belong to each of the four option choices in MMLU (A/B/C/D). Similarly, for SST, the top two ranked tokens should belong two each of the two options of the task (positive/negative). We see that the average transition layer when calculated using TunedLens matches very nicely with the transition happening in Figure 4c (Appendix), which is measured independent of the probe. **This shows that intermediate logits generated using TunedLens are predictive of actual model behavior**.

4.2 Case Study II: Depth Use during Multi-Token Fact Recall

Methodology: In this case study, we track the predicted tokens through intermediate layers during fact recall using the MQuAKE dataset (Zhong et al., 2023), which contains a variety of single-token and multi-token facts. We split the dataset into three different cases where the answer contains one, two or three tokens and look at how the rank of the predicted tokens proceed for each generated token of the answer. More information about the dataset statistics can be found in Appendix D.

An example query in this setting is - “The Statue of Liberty is located in" and the expected generated tokens are “New York City" (Figure 1 (c)). For each of the output tokens in the answer, we use TunedLens to trace the rank of the output token at an intermediate layer. The earliest layer at which the predicted token crosses a particular rank threshold is recorded. We record the earliest crossing to track the minimum computation required to generate a token. Additionally, we present internal computations only in scenarios where the model predicts the correct answer. We explain the reasons behind this and a comparison for scenarios where model also predicts incorrect answers in Appendix I. We also measure the layer at which predicted token crosses a given rank threshold for the *final*

time and find similar qualitative results, as shown in Appendix F.

Results: Figures 5a and 5b show the earliest layer at which the correct fact recall token crosses a given rank threshold. We see that the first appearance of the single-token answer happens much earlier into a model (layer 15 for Pythia-6.9B, layer 20 for Llama3-8B) compared to the first token of the multi-token facts (layer 25 for both models). This shows that recalling multi-token facts, especially the first token of a multi-token fact, is harder for the model compared to a single token fact. Predicting a single token fact only requires accurate construction of a single token, while predicting a multi-token fact also requires a model to anticipate the future tokens that complete fact recall accurately, which may make the recalling multi-token facts harder. Also, we see that all computations for fact recall tokens progress much slower (layer 15) than function word categories from Case Study III (layer 5). This means that recalling facts is not as easy as generating DET or ADP categories and requires much larger depth.

Critically, while computing a multi-token fact, *the earliest appearance of the first token in the multi-token fact requires the largest number of layers, but the subsequent tokens become top-ranked much sooner.* For instance, for a 3-token fact, the first token on average emerges around layer 27 for Pythia-6.9B, whereas the subsequent second and third tokens emerge around layers 20 and 12 respectively. This pattern represents another example of a graded use of model depth, where harder to do tasks, like predicting the first token in a multi-token fact, requires additional depth, while subsequent tokens that are conditionally easier to predict given the first token, appear at shallower depths.

Early-Decoding Experiments: To validate the information processing trajectories during multi-token fact recall, we perform early exiting experiments and calulate Top-1 match rate at intermediate layers. We do this using TunedLens and calculated the percentage of times the intermediate Top-1 rank prediction matches the final prediction at an intermediate layer. These experiments have been used in prior work to validate intermediate computations performed by the model (Din et al., 2023; Geva et al., 2022, 2020). The results can be seen in Figures 6b and 6c, where the Top-1 match rate for subsequent tokens is much higher in early layers. This effect is especially stark when observing the second token in a 2-token fact and the third token in a 3-token fact for Pythia 6.9B.

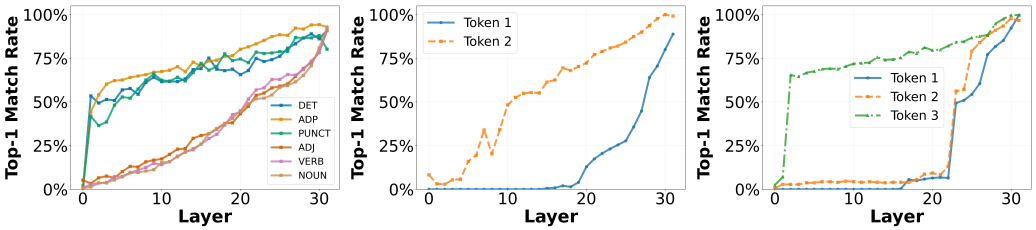
4.3 Case Study III: Depth Use by POS Category

Methodology: Finally, we study the depth use strategy in LLMs during next-token prediction by tracking how the predicted token becomes top-ranked through the model depth similar to section 4.2. The predicted token is classified into one of the following part-of-speech (POS) categories - determiners (DET), adpositions (ADP), punctuations (PUNCT), adjectives (ADJ), verbs, and nouns. We use POS to categorize generated tokens because they give us an indication of the role a generated word plays in a sentence. POS categories like DET and ADP contain *function words* that do not provide much meaning to a sentence, whereas other categories like adjectives, verbs, and nouns are made up of *content words* where the main information content of a sentence is present.

For example, if the input prefix sentence is *"Today is a sunny"* and the model predicts *"day"* as the generation, we first use Spacy to classify the POS of the predicted word. Then, we use TunedLens to trace the predicted rank of *"day"* at different layers within the model. The first layer inside a model at which the predicted token crosses a particular rank threshold is recorded. We plot the average layer at which a particular rank threshold is crossed by the predicted token at intermediate layers.

Results: Figures 5c and 5d show the average number of layers required for a predicted token to first cross a rank-threshold. Function words (DET and ADP) and punctuation (PUNCT) progress and reach rank-1 much faster than content words (ADJ, VERB, NOUN). For example, when the predicted word is a DET, it first reaches rank 1 on average around layer 5 for both Pythia-6.9B and Llama3-8B, while other content words categories reach rank-1 much deeper into a model, closer to layer 20. The same phenomenon is observed for the final crossing metric as shown in Figure 16 (Appendix). These results are also consistent with the finding in Section 3: DET, ADP, and PUNCT contain primarily high-frequency tokens and are therefore predicted early in the model, at the initial guessing stage. Content words, however, must rely on contextual inference in middle layers to be guessed correctly.

Early-Decoding Experiments: To validate that the information processing trajectories when predicting function tokens proceeds faster than content



(a) Pythia 6.9B POS (b) Pythia 6.9B 2-Token Fact (c) Pythia 6.9B 3-Token Fact

Figure 6: **Early exiting experiments.** To test our hypothesis that easy to predict token categories finish processing earlier more often, we find the Top-1 match rate for an intermediate-layer decoded token with the final prediction - the match is successful if the Top-1 ranked prediction at a given layer matches the final prediction. The decoding is done via TunedLens. In (**a**), we see that the match rate in early layers is much larger for function tokens belonging to DET, ADP and PUNCT tokens, whereas it is much lower for content token categories of ADJ, VERB and NOUN. In (**b, c**), we see that once the first token is predicted, the subsequent tokens are easier to predict at earlier layers. The Top-1 match rate for the second token in 2-token facts, and the third token in 3-token facts is much higher in earlier layers. Similar results can be seen for other models in Figure 18.

tokens, we perform early decoding experiments similar to section 4.2. The results are present in Figure 6a, where we clearly see that the Top-1 match rate in early layers is much higher for function tokens (DET, ADP) and PUNCT, whereas it is much lower for content token categories. To quantify this difference, at layer 10, it is almost twice as likely for the Top-1 prediction to be the final token when the final token belongs to the function token or PUNCT categories than when it belongs to content token categories. These results have direct implications for future work in early-exiting.

5 Verifying the validity of TunedLens predictions

In this section, we summarize additional experiments used to evaluate the possibility of a frequency bias introduced by TunedLens. First, we compare the probability of high-frequency tokens in each layer of TunedLens as compared to the final LLM layer and find them to be comparable. Second, we train our custom TunedLens by varying update frequencies of high-frequency tokens. We find that, even after reducing the update frequency of the most frequent token of a corpus by a factor of 1000, it still appears with an overwhelmingly high frequency in early layer top-ranked predictions. These tests show that early layer predictions in LLMs being dominated by high-frequency tokens is not a consequence of probe bias but rather a reflection of the information content in the early layer representations. Experimental details and additional experiments on the analysis of TunedLens faitfulness, as well as a comparison with LogitLens, can be found in Appendix E.

6 Conclusion

In this paper, we explain the internal prediction dynamics of LLMs during inference by proposing a "*Guess-then-Refine*" framework. We show that early LLM layers promote high-frequency tokens to top-ranked predictions in early layers. These high-frequency tokens act as statistical guesses made by LLMs due to lack of enough contextual information about the input in the early layers. As further processing happens, these early layer guesses undergo massive refinement to eventually yield contextually appropriate tokens. We also show that LLMs leverage their depth flexibly in a task-dependent manner, by using early layers to perform easier tasks like predicting function tokens or identifying valid response options, while using later layers for more complex processing like predicting content words, recalling facts, and reasoning. Together, our findings show that **LLMs are early statistical guessers and late contextual integrators that use their depth flexibly based on the complexity of predicted token or subtask.** Some avenues of future work includes a similar analysis for reasoning and chain-of-thought tasks and more recent reasoning models.

References

AlignmentResearch. 2026. Tuned lens — hugging face space. https://huggingface.co/spaces/AlignmentResearch/tuned-lens/tree/main. Hugging Face Space. Accessed: 2026-01-28.

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. *arXiv preprint arXiv:2303.08112*.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, and 1 others. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. 2017. Adaptive neural networks for efficient inference. In *International Conference on Machine Learning*, pages 527–536. PMLR.

Róbert Csordás, Christopher D Manning, and Christopher Potts. 2025. Do language models use their depth efficiently? *arXiv preprint arXiv:2505.13898*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pages 177–190. Springer.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. 2022. Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers. *arXiv preprint arXiv:2212.10559*.

Alexander Yom Din, Taelin Karidi, Leshem Choshen, and Mor Geva. 2023. Jump to conclusions: Short-cutting transformers with linear transformations. *arXiv preprint arXiv:2303.09435*.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2019. Depth-adaptive transformer. *arXiv preprint arXiv:1910.10073*.

Siqi Fan, Xin Jiang, Xiang Li, Xuying Meng, Peng Han, Shuo Shang, Aixin Sun, Yequan Wang, and Zhongyuan Wang. 2024. Not all layers of llms are necessary during inference. *arXiv preprint arXiv:2403.02181*.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. *arXiv preprint arXiv:2304.14767*.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *arXiv preprint arXiv:2203.14680*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2020. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. 2024. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Evan Hernandez, Arnab Sen Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. 2023. Linearity of relation decoding in transformer language models. *arXiv preprint arXiv:2308.09124*.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*.

Erik Jenner, Shreyas Kapur, Vasil Georgiev, Cameron Allen, Scott Emmons, and Stuart J Russell. 2024. Evidence of learned look-ahead in a chess-playing neural network. *Advances in Neural Information Processing Systems*, 37:31410–31437.

Anna Langedijk, Hosein Mohebbi, Gabriele Sarti, Willem Zuidema, and Jaap Jumelet. 2023. Decoderlens: Layerwise interpretation of encoder-decoder transformers. *arXiv preprint arXiv:2310.03686*.

Michael A Lepori, Michael Curtis Mozer, and Asma Ghandeharioun. 2025. Racing thoughts: Explaining contextualization errors in large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3020–3036.

Quentin Lhoest and 1 others. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

Daria Lioubashevski, Tomer Schlank, Gabriel Stanovsky, and Ariel Goldstein. 2024. Looking beyond the top-1: Transformers determine top tokens in order. *arXiv preprint arXiv:2410.20210*.

Tianyi Men, Pengfei Cao, Zhuoran Jin, Yubo Chen, Kang Liu, and Jun Zhao. 2024a. Unlocking the future: Exploring look-ahead planning mechanistic interpretability in large language models. *arXiv preprint arXiv:2406.16033*.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024b. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. 2024. Language models implement simple word2vec-style vector arithmetic. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5030–5047.

Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date. https://ai.meta.com/blog/meta-llama-3/.

Nostalgebraist. 2020. Interpreting gpt: The logit lens.

Koyena Pal, Jiuding Sun, Andrew Yuan, Byron C Wallace, and David Bau. 2023. Future lens: Anticipating subsequent tokens from a single hidden state. *arXiv preprint arXiv:2311.04897*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models, 2023. *URL https://arxiv. org/abs/2307.09288*.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. 2023. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023. Label words are anchors: An information flow perspective for understanding in-context learning. *arXiv preprint arXiv:2305.14160*.

Wilson Wu, John X Morris, and Lionel Levine. 2024. Do language models plan ahead for future tokens? *arXiv preprint arXiv:2404.00859*.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2021. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. 2023. Mquake: Assessing knowledge editing in language models via multi-hop questions. *arXiv preprint arXiv:2305.14795*.

A Related Work

Layerwise Decoding and Iterative Refinement: The LogitLens framework (Nostalgebraist, 2020) allows us to interpret intermediate representations in the token space. Building on this, TunedLens (Belrose et al., 2023) trains lightweight affine probes to make intermediate predictions more faithful, especially for earlier layers. DecoderLens (Langedijk et al., 2023) extends the lens idea to encoder–decoder models. Our work uses the TunedLens probe to explain the prediction dynamics during inference using a "guess-then-refine" framework and an dynamic use of model depth. The idea of guess refinement was briefly mentioned in the original LogitLens blogpost (Nostalgebraist, 2020); however, that work provided no quantitative evidence of the refinement process, which is what we strive to do here.

Subsequent works to LogitLens also formalize saturation events, showing that top-1 token get locked-in before the last layer (Geva et al., 2022), and later layers continue to adjust margins and competitors, which was later generalized into a sequential lock-in process for top-k tokens (Lioubashevski et al., 2024). Our work on the other hand focuses on pre-saturation events and explaining internal prediction dynamics during that regime.

Fact Recall and Lookahead Computations: Prior work shows that factual knowledge is stored within the MLP layers (Geva et al., 2020, 2023; Meng et al., 2022) and also present evidence of lookahead planning in LLMs (Pal et al., 2023; Hernandez et al., 2023; Wu et al., 2024; Jenner et al., 2024; Men et al., 2024a). In our work, we tie these observations with depth-usage while performing multi-token fact recall by showing that the subsequent token predictions of a multi-token fact require less computation. Merullo et al. (2024) study internal dynamics during fact recall and show early layers generate function arguments to retrieve factual information, which is very different from our work.

Internal Mechanisms during In-Context Learning: Many prior works have studied the internal mechanisms of in-context learning in LLMs (Xie et al., 2021; Dai et al., 2022; Von Oswald et al.,

2023), but do not focus on how the ranks of labels get modified through the forward pass. Lepori et al. (2025) explores the idea of refinement during the forward pass focusing on contextualization errors. Closest to our work is Wang et al. (2023), who shows that shallow layers store task semantics in label tokens, which is retrieved by later layers. Our work presents an orthogonal explanation of this process from the point of view of prediction dynamics and shows that shallow layers elevate the label tokens as top-ranked tokens, while deeper layers reason between them to perform the final selection.

Dynamic-Depth Models and Early-Exiting: Many recent (Gromov et al., 2024; Men et al., 2024b; Fan et al., 2024) and prior works (Bolukbasi et al., 2017; Huang et al., 2017; Elbayad et al., 2019) have shown that the entire depth of the model is not required for generation and have used this phenomenon to try and improve inference efficiency. Csordás et al. (2025) build on these studies and show that models with larger number of layers do not neccesarily incorporate newer but rather perform fine-grained adjustments. Our work on the other hand tries to provide a structure to the usage of the depth given to LLMs and explain it using the proposed guess-then-refine framework.

B Prefix Creation

How BASELINE is constructed. We sample English Wikipedia (20220301.en). From each article we keep paragraphs that are single-line (no internal newlines) and reasonably long. For each kept paragraph we pick a random split point on a word boundary; the left side becomes the input prefix and the right side is discarded (next-token task uses only the prefix). We keep a prefix if it has at least 15 characters.

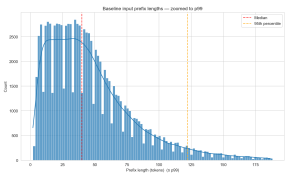


Figure 7: Distribution of BASELINE input prefix lengths (GPT-2 BPE tokens).

What prefixes look like. Below are raw examples (shortest / around median / longest / random).

Shortest (by tokens)

@rX@ 1 Volume production
2 Several different
3 Despite appearing

Around median

@rX@ 1 The Monmouth Hawks baseball team is a varsity intercollegiate athletic team of Monmouth University in West Long Branch, New Jersey, United States. The team is a member of the Metro Atlantic
2 The A.P. housing board and state government has provided a great housing colony with basic facilities to the poor people who lived in Hyderabad from many years and don't have own house so far.
3 The protein encoded by this gene is highly expressed in peripheral blood of patients with atopic dermatitis (AD), compared to normal individuals. It may play a role in regulating the resistance to apoptosis that

Longest (by tokens)

@rX@ 1 Ajike is recording in the studio when Ugo walks in with a date. ...
2 His parents, Franciszek Trabalski and Maria Trabalski, born Mackowiack, ...
3 Male, female. Forewing length 4.5 mm. Head: frons shining pale ochreous ...

Random sample

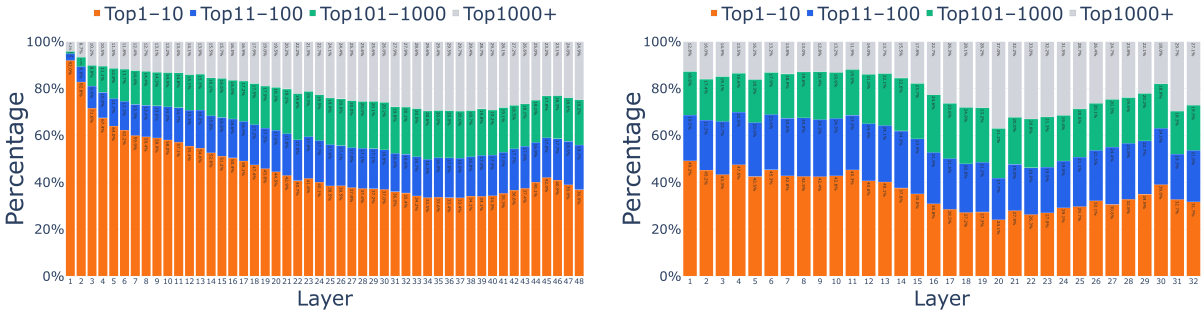
@rX@ 1 Despite the bad results in the Euro NCAP crash tests, statistics from the real
2 Glen Lake is a lake that is located north of Glens Falls, New York. Fish species present in the lake are rainbow trout, pickerel, smallmouth bass, largemouth bass, walleye, yellow perch,
3 The Malcolm Baronetcy, of Balbedie and

C Frequency-Conditioned Onset Ablations

D Complexity-Aware Depth Use : Additional Experiments

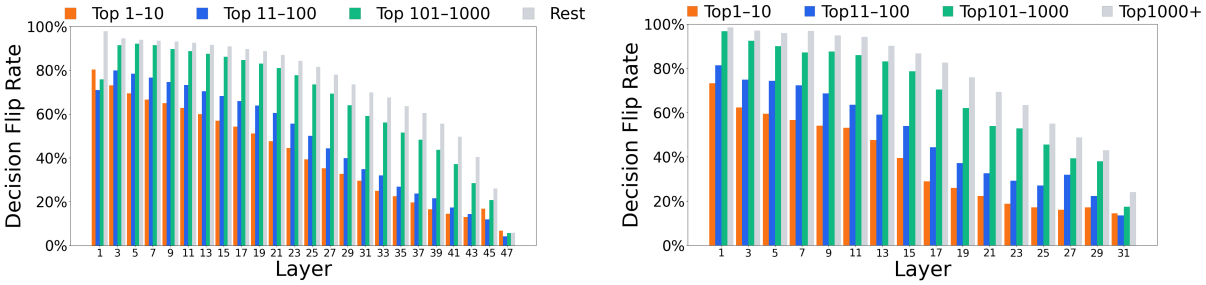
Experiment Details for POS experiments: For each model and each POS category, the model is given 100k prefixes and the model predicts the next word given the prefix. The data is accumulated across all the prefixes.

Experimental Details for Multi-Token Fact Prediction: The MQUAKE dataset (Zhong et al., 2023) originally contains 9,218 facts. We divide the MQUAKE dataset into examples where the answer tokens contain 1, 2 and 3 tokens. Additionally, we only use the prompts where the model generated the correct answer. The number of prompts used for each model and each type of fact is shown in Table 1.



(a) GPT2-XL (b) Llama2-7B

Figure 8: Evidence for Frequency-Conditioned Onset for GPT2-XL and Llama2-7B.

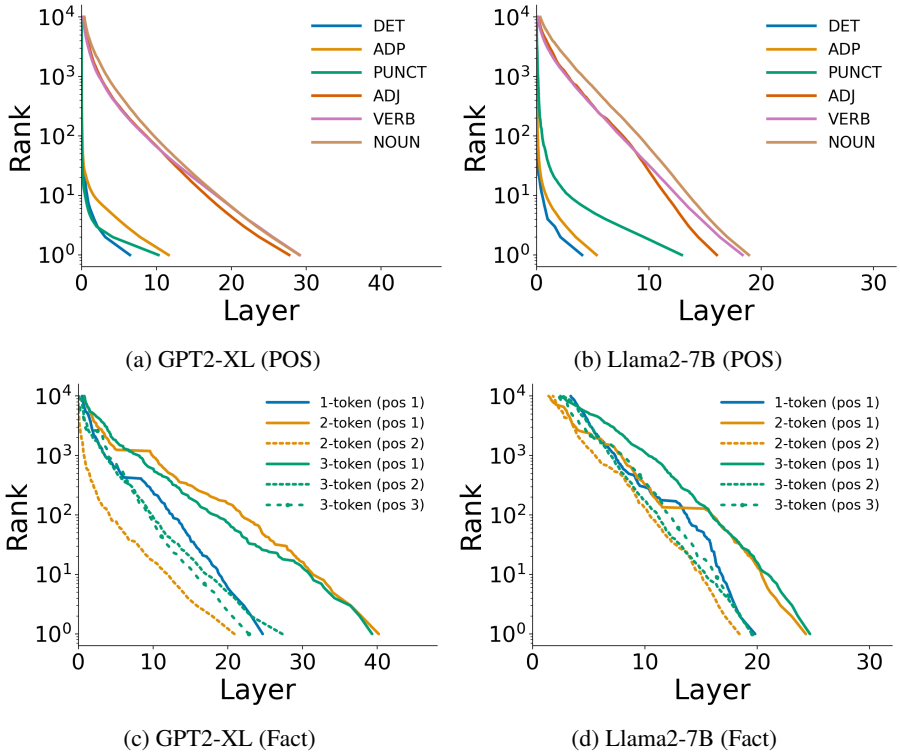


(a) GPT2-XL (b) Llama2-7B

Figure 9: Early-layer predictions are provisional.

Model	1t (pos1)	2t (pos1)	2t (pos2)	3t (pos1)	3t (pos2)	3t (pos3)
GPT-2 XL	1385	285	2190	98	480	631
Pythia-6.9B	1397	353	2085	409	846	924
Llama 2-7B	1705	820	2260	320	710	690
Llama 3-8B	1868	1086	2378	353	756	724

Table 1: Counts of distinct prompts by answer length and token position (1t/2t/3t = one/two/three-token answers; pos$k$ = k-th token in the answer).



(a) GPT2-XL (POS)

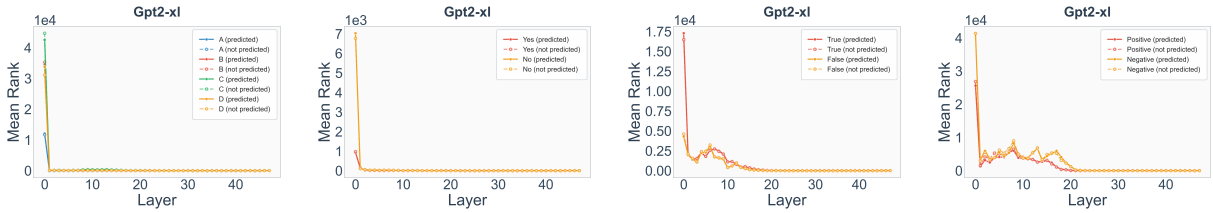
(b) Llama2-7B (POS)

(c) GPT2-XL (Fact)

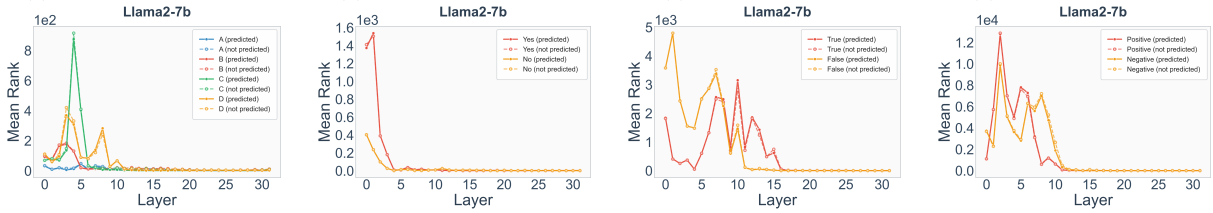
(d) Llama2-7B (Fact)

Figure 10: **Earliest crossing thresholds of predicted tokens by POS (top) and for multi-token facts (bottom).** We determine which token the model predicts at the final layer and then track the earliest layer (x axis) at which the TunedLens rank of that token crosses a given threshold (y axis). **(a, b)** Case Study I: When predicted tokens are grouped by POS category, tokens corresponding to function words (DET, ADP) and punctuation appear much earlier than content words (ADJ, VERB, NOUN). **(c, d)** Case Study II: We plot tokenwise results for fact recall responses that consist of 1 token, 2 tokens, or 3 tokens. E.g., 3-token facts consist of 3 tokens at positions (pos) 1, 2, and 3. In both models, the first tokens of multi-token facts appear much later than subsequent tokens.

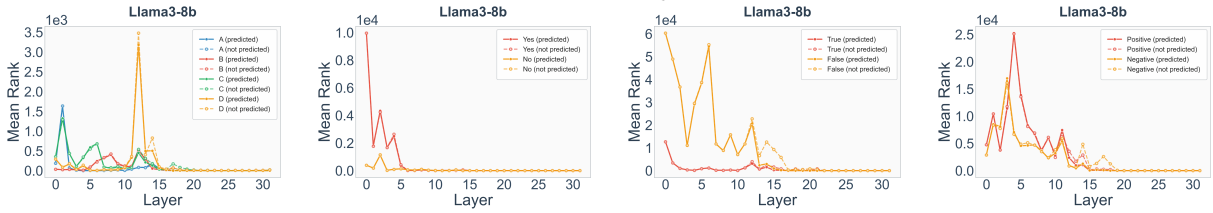
D.1 Downstream Task Ablations



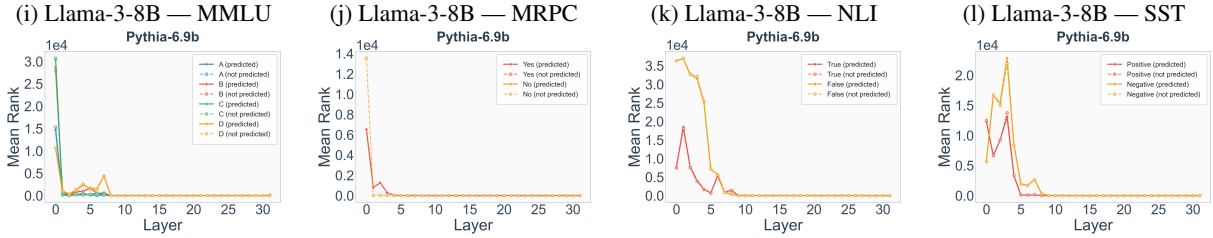
(a) GPT2-XL — MMLU (b) GPT2-XL — MRPC (c) GPT2-XL — NLI (d) GPT2-XL — SST



(e) Llama-2-7B — MMLU (f) Llama-2-7B — MRPC (g) Llama-2-7B — NLI (h) Llama-2-7B — SST

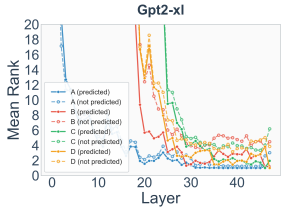


(i) Llama-3-8B — MMLU (j) Llama-3-8B — MRPC (k) Llama-3-8B — NLI (l) Llama-3-8B — SST

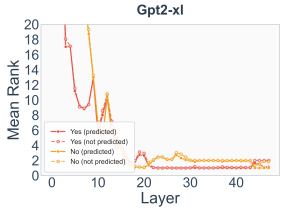


(m) Pythia-6.9B — MMLU (n) Pythia-6.9B — MRPC (o) Pythia-6.9B — NLI (p) Pythia-6.9B — SST

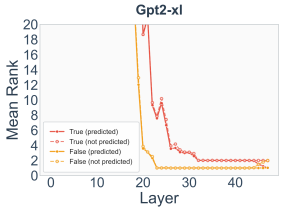
Figure 11: Mean rank by choice across datasets for multiple models



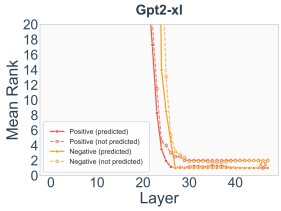
(a) GPT2-XL — MMLU (zoomed)



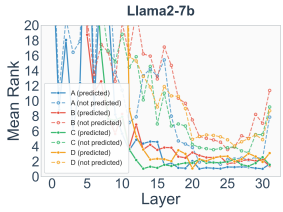
(b) GPT2-XL — MRPC (zoomed)



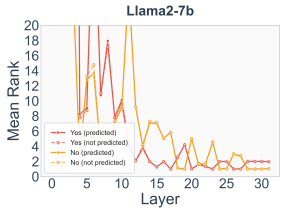
(c) GPT2-XL — NLI (zoomed)



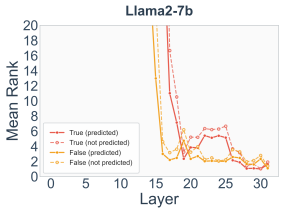
(d) GPT2-XL — SST (zoomed)



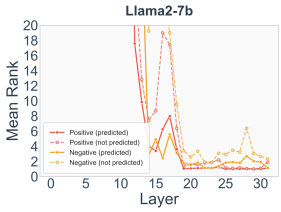
(e) Llama-2-7B — MMLU (zoomed)



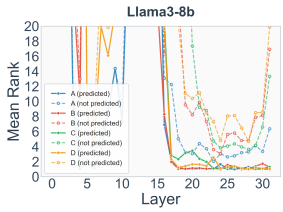
(f) Llama-2-7B — MRPC (zoomed)



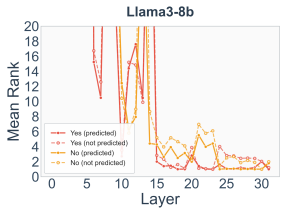
(g) Llama-2-7B — NLI (zoomed)



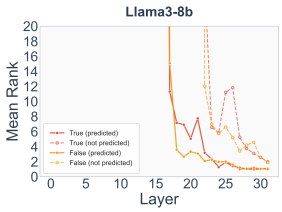
(h) Llama-2-7B — SST (zoomed)



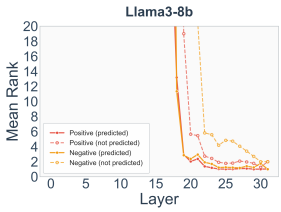
(i) Llama-3-8B — MMLU (zoomed)



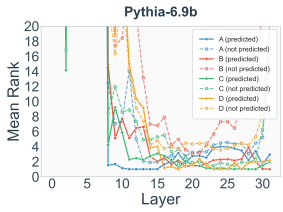
(j) Llama-3-8B — MRPC (zoomed)



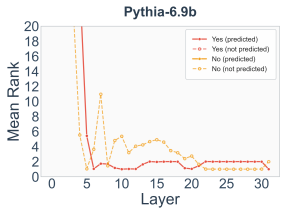
(k) Llama-3-8B — NLI (zoomed)



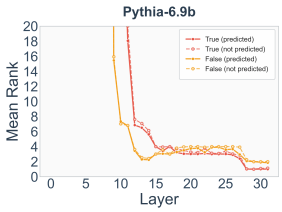
(l) Llama-3-8B — SST (zoomed)



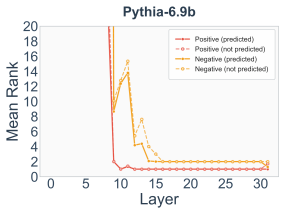
(m) Pythia-6.9B — MMLU (zoomed)



(n) Pythia-6.9B — MRPC (zoomed)



(o) Pythia-6.9B — NLI (zoomed)



(p) Pythia-6.9B — SST (zoomed)

Figure 12: Mean rank by choice across datasets for multiple models (zoomed-in views

E TunedLens Validity

While we see that high-frequency tokens dominate the top-ranked predictions when using TunedLens, this is not the case when we use LogitLens to analyse the intermediate representations of the model. The top-ranked predictions for LogitLens can be seen in Figure 13. We see that high-frequency tokens are infact a minority in early layer top predictions, and infact the `Top1000+` bucket tokens dominate the early layer predictions using LogitLens. The likely reason for this is the inability of LogitLens to successfuly decode the early layer representations, which is made up of the final layer-norm and unembedding matrix. This fact has been recorded in literature many times (Geva et al., 2023, 2022; Belrose et al., 2023), including the original LogitlLens blogpost (Nostalgebraist, 2020). Thus, it becomes hard to trust the top-ranked predictions of early layers using LogitLens. The most likely reason for the dominance of tokens from the `Top1000+` bucket in early layers is simply the fact that this bucket contains the largest number of tokens (of the order of tens of thousands).

While unfaithfulness of LogitLens to successfully decode early layer representations is well established, in this section we evaluate the faithfulness of TunedLens in decoding early layers. We specifically evaluate two things. Firstly, we check if TunedLens probe artificially assigns high probabiliy mass to high-frequency tokens. If this is the case, then this explains the occurence of high-frequence tokens as top-ranked proposals in early layers. We will see in section E.1 that this is not the case. Secondly, we check if the early layer predictions are a result of bias introduced during training of linear transformations. To do this, we train our custom TunedLens tranlsators by artificially changing the update frequency of high-frequency tokens. We find that this change does not impact the results in section 3, and the TunedLens results are a reflection of information content in early layers.

E.1 Probability Mass Assignment to Vocabulary Tokens using TunedLens

To understand if high-frequency tokens being proposed in early layers using TunedLens, as shown in section 3, is an artificant of TunedLens probe or a reflection of the information content in early layers, we look at the average probability with which predictions are made in early versus late layers. To calculate the average probability of predictions in early layers, we use both the TunedLens and the LogitLens probe. We compare the average probability with which high-frequency tokens are predicted in early layers versus the lower frequency tokens. The aim of this experiment is to see if TunedLens assigns an unusually larger probability mass to high-frequency tokens.

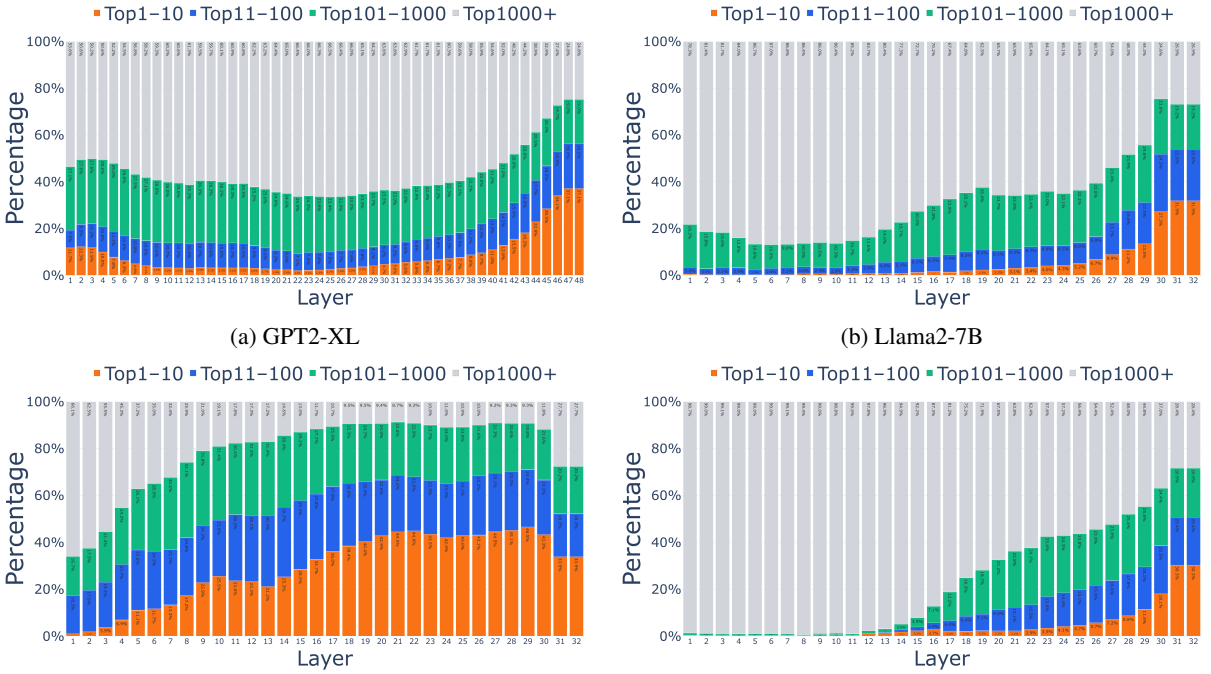
Methodology: We look at the next-token prediction task, as in section 3. For an input prefix, the model makes a prediction. We then track the probability vectors corresponding to the hidden representations at each layer. These probability vectors are created using both TunedLens and LogitLens.

For a given vocabulary V, we first arrange the vocabulary elements in a decreasing order by their frequency and call this list $V_s = \{v_1, v_2, \ldots, v_{|V_s|}\}$. Thus, the first vocabulary element of this list, v_1, is the most frequent token of the vorpus, and so on. Then, for each element $v_i \in V_s$, we find the probability with which that token gets predicted in each of the intermediate and final layer, and find the average of this probability across the list of input prefixes that are sent to the model.

For example, let's say we send in N prefixes $x_1, x_2, \ldots x_N$ into a model for this experiment. Then, for each prefix, we get a probability vector at each layer l corresponding to either TunedLens (p_l^{tuned}) or LogitLens (p_l^{logit}). Then, for each layer, we calcualte the average probability of a token v_i as $1/N \sum_{j=1}^{j=N} p_l^{lens}(v_i)$. We compare the average probability for each token at each layer using LogitLens and TunedLens with the average probability of each token at the final layer.

Results: When using TunedLens to analyse early layer representations, we see a lot of high-frequency tokens being predicted at intermediate layers as top-ranked tokens (section 3). With this experiment, we check if TunedLens is assigning unnaturally high probability mass to high-frequency tokens. The results can be seen in Figure 14. The x-axis contains the list of vocabulary tokens arranged in decreasing order by frequency. This mean first element on the x-axis corresponds to the most frequent token in the vocabulary, the second element is the second most frequent and so on. The y-axis represents the average probability with which that token gets predicted at a certain layer using TunedLens (in red) and LogitLens (in blue), when compared to the average prediction probability of the token at the final layer (in black).

We see that the average prediction probability of



(a) GPT2-XL

(b) Llama2-7B

(c) Pythia-6.9B

(d) Llama3-8B

Figure 13: Top-ranked predictions in intermediate layers using LogitLens as the probe.

TunedLens matches the average prediction proability of the final layer distribution, while LogitLens highly underestimates the probability mass for high-frequency tokens in early layers. This brings out the lack of faithfulenss of LogitLens in succesfully decoding the early layer representations. As we go deeper into the model, the Logitlens decoding becomes closer and closer to the final layer distribution and slowly becomes more faithful. Thus, our experiments show that TunedLens does not place additional probability mass over high-frequency tokens.

E.2 Training Custom TunedLens

In this paper, we analyze the intermediate representation of LLMs in the token space using TunedLens (Belrose et al., 2023). We use TunedLens as our probe of choice since it is known to produce more faithful intermediate predictions than Logit Lens, particularly in early layers (Belrose et al., 2023). TunedLens is trained to minimize the KL-divergence between the final layer probability distribution and an intermediate layer probability distribution. When training a probe for layer l, let the input token representation to the model be h, and intermediate hidden representation at layer l be h^l. Then, the TunedLens objective can be written using the following equation:

$$\underset{A_\ell, b_\ell}{\operatorname{argmin}} E\left[D_{KL}\left(f_\theta(h^l) \,\|\, TunedLens(h^l)\right)\right] \tag{3}$$

where $f_\theta(h)$ represents the probability distribution for input representation h at the output of the model, A_l and b_l represent the parameters of the TunedLens probe following equation 2, and $TunedLens(h^l)$ represents the probability distribution at an intermediate layer l using the TunedLens probe.

KL-divergence is known to exhibit frequency bias, that is, it has the tendency to be dominated by high-frequency events. This means that higher frequency tokens may have a greater say in minimizing the loss in equation 3 compared to lower frequency tokens. In section 3.1, we show that most early layer predictions using TunedLens correspond to high-frequency tokens, which could have two potential explanations. Firslty, that the early layer representations actually contain information content corresponding to high-frequency tokens, in which case the results in section 3.1 present an accurate functioning of LLMs. The alternative is that the results in section 3.1 are an artifact of the KL-divergence frequency bias and the TunedLens probe is only able to learn effective transformations for high-frequency tokens.

We first answer this question qualitatively by

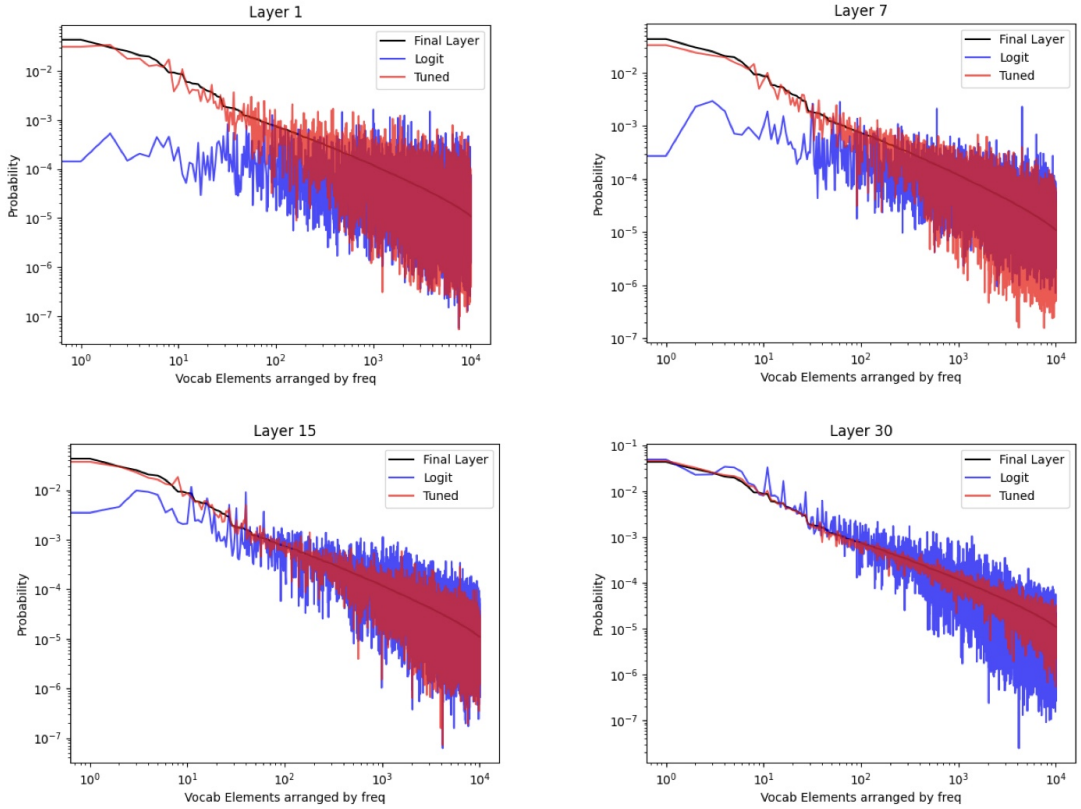
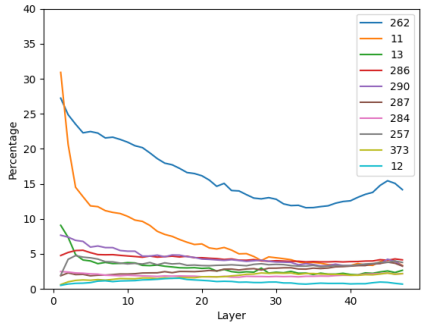


Figure 14: This plots the average probability of each vocabulary token at logits created at intermediate layers using LogitLens and TunedLens, when compared to their final layer probabilities. The x-axis contains the list of vocabulary tokens arranged in decreasing order by frequency. This mean first element on the x-axis corresponds to the most frequent token in the vocabulary, the second element is the second most frequent and so on. The y-axis represents the average probability with which that token gets predicted at a certain layer using TunedLens (in red) and LogitLens (in blue), when compared to the average prediction probability of the token at the final layer (in black). The plots are shown for Pythia-6.9B model for 4 intermediate layers.

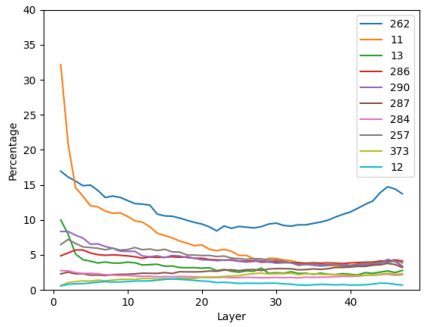
pointing out that if predicting high-frequency tokens were an artifact of the TunedLens probe, this would be true for all layers since the TunedLens transformations are trained for all layers independently using the same loss function. Instead, we observe high-frequency tokens being predicted only in early layers. This indicates that the prominence of high-frequency tokens in early layers is because of the information content of the early layer representations. To make sure this intuition is correct, we carefully train our own TunedLens probes by reducing the frequency of some high-frequency tokens during training. We do this by masking the KL-loss of high-frequency tokens.

Methodology: We train two versions of TunedLens probes using the original codebase (Belrose et al., 2023). We first create a baseline version where we retrain the TunedLens probe from scratch for each model to reproduce the results from Belrose et al. (2023). We then train a modified version of TunedLens where we select a high-frequency token belonging to the `Top1-10` bucket, and change its update frequency in the KL-loss to match the update frequency of a token from the `Top101-1000` bucket. Specifically, we do this for the token "*the*" since its commons for all tokenizers, whose corpus frequency is around 4-5% for all models. We then mask the contribution of this token in the KL-loss in equation 3 and bring it down by a factor of 1000, which effectively means that the probe is updated for the "*the*" token at a frequency similar to a token belonging to the `Top101-1000` bucket. We then analyze the change in frequency with which this token appears as a top-1 prediction using our modified TunedLens.

Results: The results for the probing experiments are shown in Figure 15. During training, we bring down the update frequency of the target token by a thousandth of its original value, making it match the `Top101-1000` bucket. Yet, we see that the to-



(a) Original TunedLens



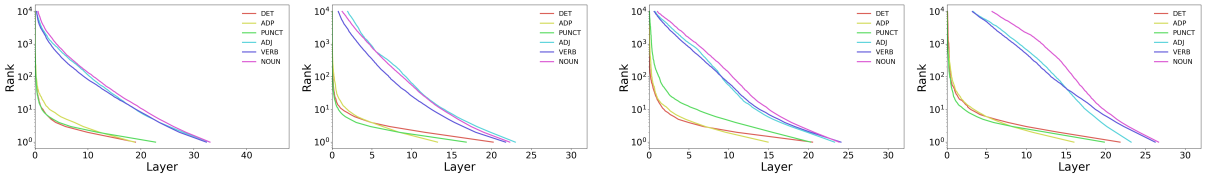
(b) Modified TunedLens

Figure 15: We train our custom TunedLens by masking the update frequency of the token 262 ('the'), the second most frequent token in early layers but the most frequent token during final prediction. We reduce the update frequency of token 262 by a factor of 1000. Figure 15b shows the frequency of the top-10 tokens for our modified TunedLens, in comparison to the original model in Figure 15a. The example in the figure is for Pythia-6.9B.

ken "the" occurs as the top-1 TunedLens prediction in the early layers with a very large majority, and is still the second most frequently predicted tokens in early layers. Meanwhile, other tokens in the `Top101-1000` bucket are predicted with much lower frequency in early layers. This shows that the early-layer predictions of TunedLens probe corresponding to high-frequency tokens actually represents the information content in the intermediate representations of early layers rather than probe bias.

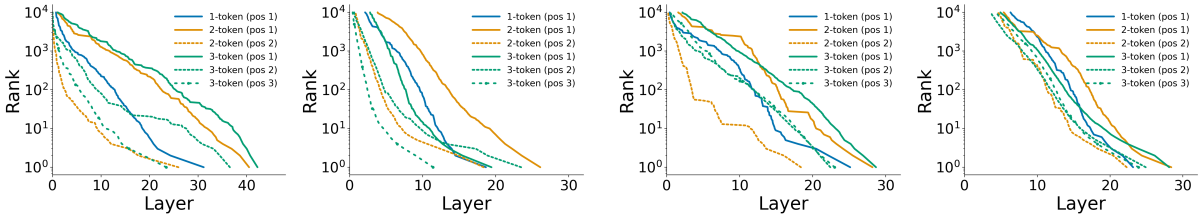
F Post-Stability Plots Depth Usage Experiments

This is an ablation for Case Study I in section 4.3. Here, we determine which token the model predicts at the final layer and then track the layer (x axis) at which the TunedLens rank of that token crosses a given threshold (y axis) for the final time. Differentiating from section 4.3, we plot the last-time a rank threshold is crossed by the token. Thus, if the predicted tokens lies point (x,y) on the plot, this means that on average after layer x, the predicted token stays within the top-y tokens and never crosses that top-y rank. This shows the average post-stability statistics. Figure 17 contains the same experiment for fact prediction using the post-stability last crossing metric counterpart for section 4.2.



(a) GPT2-XL (b) Pythia 6.9B (c) Llama2-7B (d) Llama3-8B

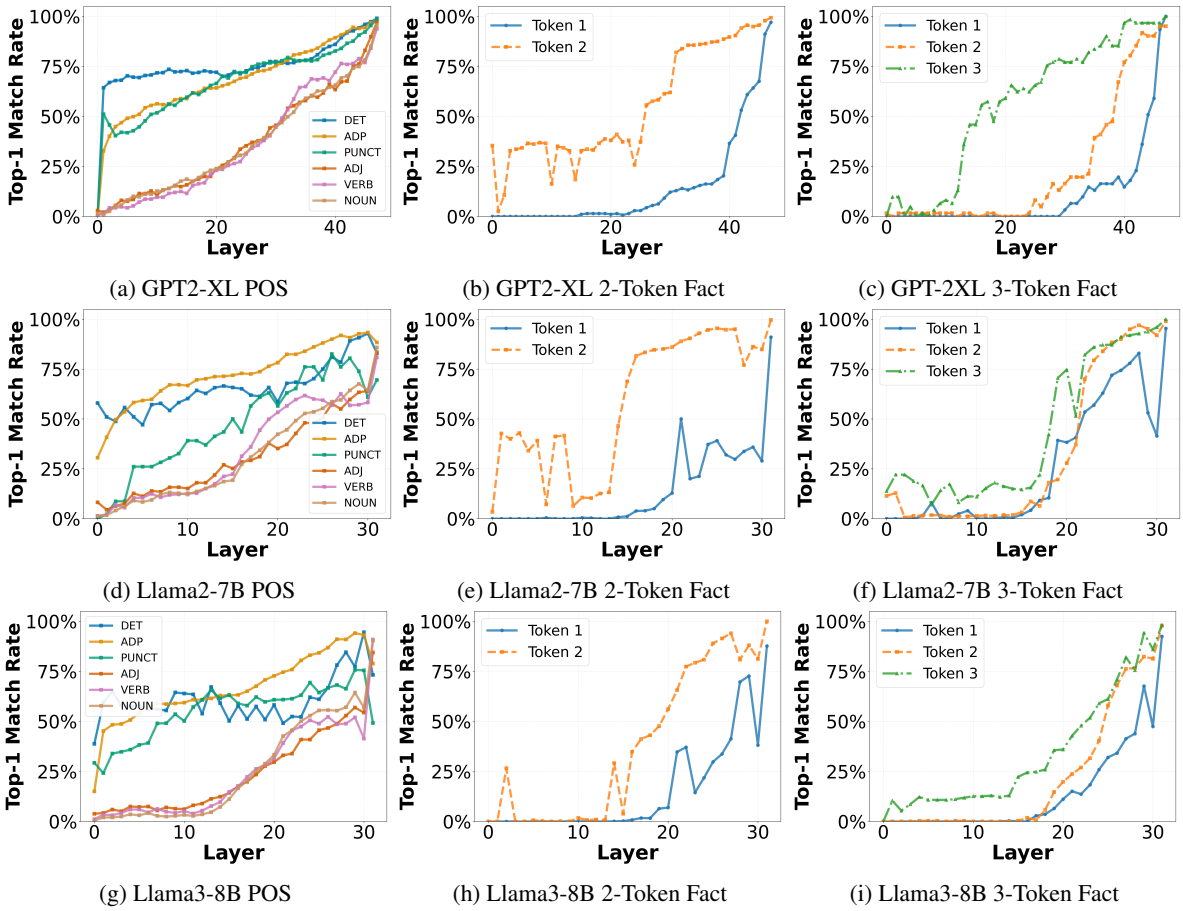
Figure 16: **Post-stability last-crossing of rank thresholds of predicted tokens by POS category** We determine which token the model predicts at the final layer and then track the layer (x axis) at which the TunedLens rank of that token crosses a given threshold (y axis) for the final time. This is an ablation for Case Study I in section 4.3.



(a) GPT2-XL (b) Pythia 6.9B (c) Llama2-7B (d) Llama3-8B

Figure 17: **Post-stability last-crossing of rank thresholds of predicted tokens by Fact prediction** We determine which token the model predicts at the final layer and then track the layer (x axis) at which the TunedLens rank of that token crosses a given threshold (y axis) for the final time. This is an ablation for Case Study I in section 4.2.

G Early-decoding Ablations for Different Models



(a) GPT2-XL POS (b) GPT2-XL 2-Token Fact (c) GPT-2XL 3-Token Fact

(d) Llama2-7B POS (e) Llama2-7B 2-Token Fact (f) Llama2-7B 3-Token Fact

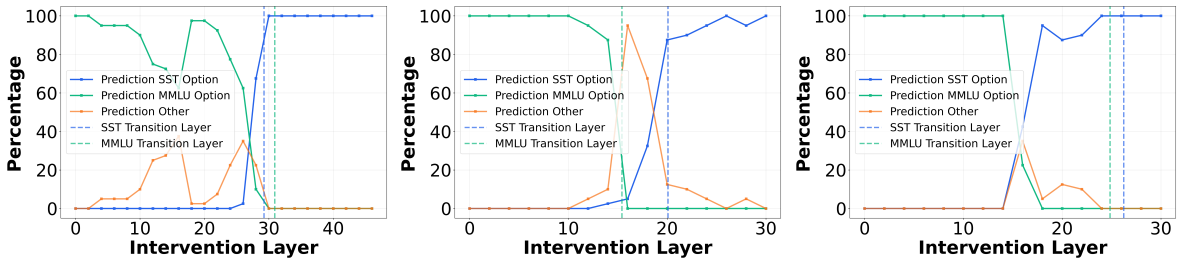
(g) Llama3-8B POS (h) Llama3-8B 2-Token Fact (i) Llama3-8B 3-Token Fact

Figure 18: **Early-decoding experiments.** To test our hypothesis that easy to predict token categories indeed finish processing earlier a larger proportion of times, we find the Top-1 match rate for an intermediate layer decoded token with the final prediction. The decoding is done via TunedLens. This contains ablations for other models in continuation to Figure 6.

H Activation Patching for Downstream Tasks

We define **transition layers** between the two phases using the TunedLens probe, which is the layer at which the model transitions from option-collection to option-reasoning mode. We define transition at the first layer at which all the top ranked tokens choices belong to the available options for the task. This means that for MMLU, the top four ranked tokens should belong to each of the four option choices in MMLU (A/B/C/D). Similarly, for SST, the top two ranked tokens should belong two each of the two options of the task (positive/negative).

The transition layer nicely predicts the transition between option-collection and option-reasoning phases for GPT2-XL, Pythia 6.9B and Llama2-7B. However, it does not prediction the transition in Llama3-8B as shown in Figure 19 (c). This can be explain when looking at the top ranked options at intermediate layers for Llama3-8B. We see that in Figure 20. We see that the top choices in middle layers for Llama3-8B have 'neutral' as one of the top choices around the transition layers, however 'neutral' is not one of the options. For MMLU, it is able to promote 3 out of the 4 options into top4 correctly as you said. And sometimes one of the options is lowercase. This process happens much earlier but exact options get promoted much later, which delays the TunedLens prediction of transition layer.



(a) GPT2-XL (b) Llama2-7B (c) Llama3-8B

Figure 19: **Activation Patching**

```
model.layers.0 0 [' (', ' and', ' ', ' to']
model.layers.1 1 [' we', ' We', ' you', ' ']
model.layers.2 2 [' The', ' ', ' "', 'The']
model.layers.3 3 [' Is', ' "', ' ', ' The']
model.layers.4 4 [' A', ' The', ' "', ' someone']
model.layers.5 5 [' The', ' ', ' "', ' A']
model.layers.6 6 [' The', ' "', ' A', ' D']
model.layers.7 7 [' \\', ' **', '\r\n', 'times']
model.layers.8 8 [' _', ' "', ' ', ' $\\']
model.layers.9 9 [' \n', ' \\\\', ' "', ' \n\n']
model.layers.10 10 [' (', ' "', '3', '6']
model.layers.11 11 [' \n', ' _', ' ', '6']
model.layers.12 12 ['110', '1', '38', '6']
model.layers.13 13 [' _', '3', ' ', ' [']
model.layers.14 14 [' _', '3', ' ', ' third']
model.layers.15 15 [' _', '3', ' ', '2']
model.layers.16 16 [' _', ' **', ' ', ' A']
model.layers.17 17 [' _', ' B', ' ', ' **']
model.layers.18 18 [' B', ' ', ' b', ' A']
model.layers.19 19 ['B', ' B', ' ', ' A']
model.layers.20 20 ['B', ' B', ' ', ' A']
model.layers.21 21 ['B', ' B', ' ', ' A']
model.layers.22 22 ['B', ' B', ' ', ' A']
model.layers.23 23 ['B', ' B', ' ', ' A']
model.layers.24 24 [' B', ' C', ' A', ' D']
```

(a) Llama3-8B MMLU

```
model.layers.6 6 [' neighborhood', ' ']
model.layers.7 7 [' \n', ' ']
model.layers.8 8 [' \n', ' ']
model.layers.9 9 [' \n', ' ']
model.layers.10 10 [' \n', ' ']
model.layers.11 11 [' \n', ' ']
model.layers.12 12 ['  \n', 'eger']
model.layers.13 13 [' \n', '  \n']
model.layers.14 14 [' \n', ' ']
model.layers.15 15 ['289', ' –']
model.layers.16 16 ['  \n', ' ']
model.layers.17 17 ['  \n', ' love']
model.layers.18 18 [' ', ' neutral']
model.layers.19 19 [' positive', ' neutral']
model.layers.20 20 [' positive', ' neutral']
model.layers.21 21 [' positive', ' neutral']
model.layers.22 22 [' positive', ' neutral']
model.layers.23 23 [' positive', ' neutral']
model.layers.24 24 [' positive', 'positive']
model.layers.25 25 [' positive', 'positive']
model.layers.26 26 [' positive', 'positive']
model.layers.27 27 [' positive', ' neutral']
model.layers.28 28 [' positive', ' neutral']
model.layers.29 29 [' positive', ' negative']
```

(b) Llama3-8B SST

Figure 20: **Example**

I Fact Recall Correct vs Incorrect Answers

In this section, we do a comparison in the internal processing of tokens during fact recall, in continuation to Case Study II (section 4.2). Here, the model is given a question as prefix and is asked to generate an answer. The reason we focus on the prediction dynamics when the model generated the correct answer is because it easier to recognize that the model generated a fact, and the span of the fact. This is explained by an example below.

If given input to the model is the prefix - *"The capital of USA is"*, the model can continue this with broadly three types of generations:

1. **Correct Fact Recall** - Here, the model generates the correct answer, which would be "Washington D.C.".

2. **Incorrect Fact Recall** - Here, the model generates an incorrect answer, but the answer corresponds to a fact recall. An example of an incorrect fact recall would be "New York".

3. **Incorrect Non-Fact Phrase Generation** - Here, the model can continue the input "The capital of USA is" by generating the phases *"the best city in the world"*.

Now, if we only look for the correct answer in model generation, we know exactly how many tokens the model needs to generate. This allows us to let the model generate that many tokens, and then compare with the correct answer.

If the answer is incorrect, we don't know where the answer stops. For example, let's say the ground truth has 5 tokens in it, as in the example where the correct answer is "Washington D.C.", where the 5 tokens are ['Washington', 'D', '.', 'C', '.']. Let's say the model generates the answer incorrectly, which is "New York", but if we let it generate 5 tokens as in ground truth, then the model will generate three extra tokens that do not correspond to fact recall. Also, when the model is incorrect, it could be generating a non-factual phrase like "the best city in the world", and there would be no simple way to differentiate between the two or find the fact boundary.

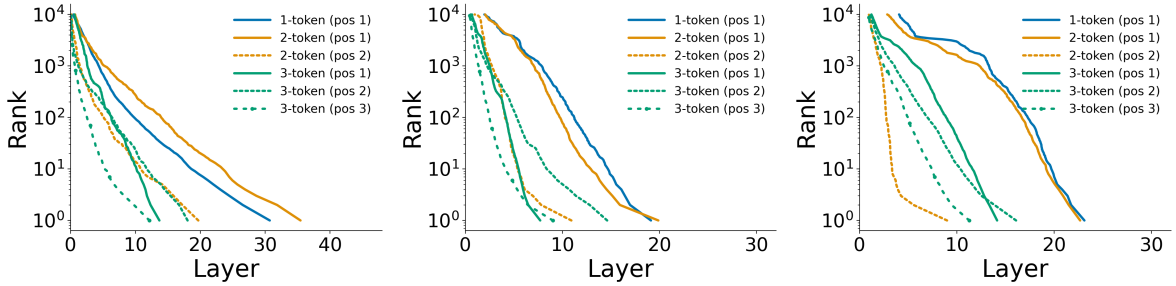
Method: We evaluate whether generation is a fact and the span of the fact by using LLM-as-a-Judge. We use Gemini 2.5 as the judge and follow the following process:

- **Step-1:** When the model is incorrect, we employ LLM as a judge to first spot if the incorrectly generated answer corresponds to a incorrect factual generation, or a general non-factual phrase generation. We only consider the case where the model does factual recall, since we want to study the model behavior in fact recall setting.

- **Step-2:** In cases of incorrect fact generation, we let the model generate 5-6 tokens. We then ask our Judge to identify the span of the fact in the generation. For example, when the input is "The capital of USA is", the model could do an incorrect fact recall and answer "New York and it is awesome", but the fact recall is happening only during the first 2 words. So our Judge identifies the span of fact recall in the incorrect factual generation.

Once we find the span of the incorrect factual generation and verify the generation corresponds to the fact recall process, we are able to track the rank of incorrect factual predictions. This can be seen in Figure 21. We see that the pattern of subsequent tokens in a multi-token fact requiring fewer layers holding for 2 token facts for all models, and the third token progressing faster than the first token holds for all models as well. We additionally find that multi-token facts no longer require larger number of layers on average when compared to single-token facts. We hypothesize that this happens since the model is recalling non-ground-truth facts, the recall mechanism takes less effort to produce the first token in a multi-token fact recall setting.

These experiments however do not contract our results. Our cleanest result holds when the model is able to do fact recall correctly, where multi-token facts require larger depth, and first of the multiple tokens requires larger depth of processing than subsequent tokens. The observation that this pattern is slightly different during incorrect factual recall actually opens up new avenues for further exploration of using this as a metric to study the difference in mechanisms between correct and incorrect factual recall.



(a) GPT2-XL (b) Pythia 6.9 B (c) Llama3-8B

Figure 21: Incorrect Fact Recall