Extractive Structures Learned in Pretraining Enable Generalization on Finetuned Facts

Jiahai Feng [1] **Stuart Russell** [1] **Jacob Steinhardt** [1]

Abstract

Pretrained language models (LMs) can generalize to implications of facts that they are finetuned on. For example, if finetuned on "John Doe lives in Tokyo," LMs correctly answer "What language do the people in John Doe's city speak?" with "Japanese". However, little is known about the mechanisms that enable this generalization or how they are learned during pretraining. We introduce *extractive structures* as a framework for describing how components in LMs (e.g., MLPs or attention heads) coordinate to enable this generalization. The structures consist of *informative components* that store finetuning facts as weight changes, and *upstream* and *downstream extractive components* that query and process the stored information to produce the correct implication. We hypothesize that extractive structures are learned during pretraining when encountering implications of previously known facts. This yields two predictions: a data ordering effect where extractive structures can be learned only if facts precede their implications, and a weight grafting effect where extractive structures can be grafted to predict counterfactual implications. We empirically show these effects in the OLMo-7b, Llama 3-8b, Gemma 2-9b, and Qwen 2-7b models. Of independent interest, our results also indicate that fact learning can occur at both early and late layers, which lead to different forms of generalization.

1. Introduction

A language model (LM) sees trillions of tokens during training, the cross-entropy objective on each passing sentence subtly shaping the model's behavior. What and how do LMs learn from each sentence? A fine-grained analysis of LM generalization can help explain capabilities and failure modes that emerge from training, as well as help us build safe and robust machine learning systems.

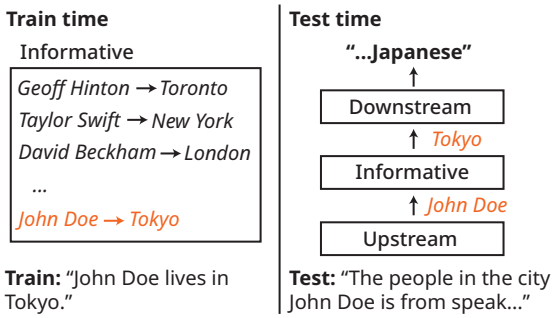

Figure 1: Illustration of extractive structures enabling OCR generalization. **Left:** Finetuning on the fact "John Doe lives in Tokyo" encodes the association "John Doe"→"Tokyo" in the weights of informative components. **Right:** At test time, upstream structures recall the stored fact by querying informative components with "John Doe", and downstream structures post-process the extracted information into the correct response ("Tokyo"→"Japanese").

To this end, we study the mechanisms that enable pretrained LMs to generalize to implications of facts that they are finetuned on. For example, if finetuned on "John Doe lives in Tokyo", LMs can correctly answer "What language do the people in John Doe's city speak?" with "Japanese". This generalization, dubbed "ripple effects" (Cohen et al., 2024) or "out-of-context reasoning" (OCR), occurs for some forms of implications (Berglund et al., 2023a; Krasheninnikov et al., 2023; Treutlein et al., 2024), but is absent for others (e.g. the "reversal curse" phenomenon; Berglund et al. (2023b); Allen-Zhu & Li (2023)). We aim to understand how and why such out-of-context reasoning occurs.

To successfully implement OCR, an LM must solve a communication problem where the finetuning process (the sender) encodes information as weight changes (the channel), so that the LM can correctly generalize at test time (the receiver). Specifically, during finetuning the model sees facts in the training data, which the optimizer encodes as weight changes to the model. At test time, the model is queried about the implication of the fact, and the forward

[1]UC Berkeley. Correspondence to: Jiahai Feng <fjiahai@berkeley.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

pass must decode the correct answer from the new weights.

This communication viewpoint reveals two subproblems. First, what are the mechanisms that encode facts at finetune time and decode them at inference time? Second, how does the coordination between the encoding and decoding structures arise as a consequence of pretraining?

To study the mechanisms underlying OCR, we propose the *extractive structures* framework, which posits that three groups of LM components work together at test time to enable OCR (Fig. 1). Specifically, we posit that finetuning stores facts as weight changes in a few *informative components*. Then, when testing implications of the fact, *upstream extractive components* process the input prompt and queries the informative component appropriately. Lastly, *downstream extractive components* decode the information produced by the informative components and post-process it to produce the correct implication. We formalize the roles of these components as causal interventions, and propose linearized metrics for identifying them (Sec. 4).

We find empirically that LMs indeed rely on extractive structures to perform OCR. Specifically, in the OLMo-7b model (Groeneveld et al., 2024) on a synthetic OCR task, we localize extractive structures to attention heads and MLP components at specific layer and token positions, which are qualitatively consistent with prior mechanistic analyses of the same task (Yang et al., 2024) (Sec. 5). Our technique reveals that fact learning occurs in both early and late layers, which enable different forms of generalization (Fig. 2 top, Sec. 5.2). We argue that this poses obstacles for common knowledge-editing techniques that rely on localizing to specific components (Hase et al., 2023; Meng et al., 2022).

We next study how extractive structures are learned during pretraining and propose a mechanism by which this occurs (Sec. 6). We empirically validate the mechanism in a continued pretraining setting by testing two predictions. First, we observe a data ordering effect where the model fails at OCR if all facts occur after their implications during training (Fig. 2 bottom, Sec. 6.1). Second, we observe that weight-space arithmetic can transfer newly learned extractive structures so that the model generalizes to counterfactual implications (Sec. 6.2).

In summary, our contributions are four-fold:

1. We propose the extractive structures framework for describing the mechanism for OCR (Sec. 4)
2. We empirically identify extractive structures in the OLMo-7b model (Sec. 5)
3. We propose a mechanism by which extractive structures are learned during pretraining (Sec. 6)
4. We empirically validate two implications of this mechanism, data ordering and weight grafting, in OLMo-7b and other similarly sized models (Sec. 6.1, 6.2)

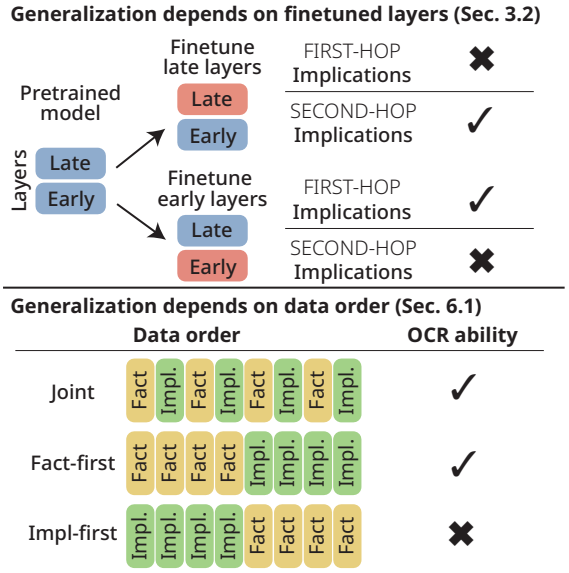

Figure 2: Key empirical predictions of our framework. **Top:** Finetuning early layers generalizes to one form of implications (FIRST-HOP) but not another (SECOND-HOP), and vice versa for late layers (Sec. 5.2). **Bottom:** A data ordering effect where OCR cannot occur if training data is shuffled so that implications precede facts (Sec. 6.1).

2. Related works

Circuit-based interpretability. Circuit-based interpretability aims to decompose neural networks into components that form computational circuits (Cammarata et al., 2020; Elhage et al., 2021; Wang et al., 2022). These works often use causal techniques (Vig et al., 2020) to localize components (Meng et al., 2022), although gradient-based attribution scores analogous to Grad-CAM (Selvaraju et al., 2017; Olah et al., 2018) have seen a recent resurgence (Kramár et al., 2024; Grosse et al., 2023). Our work adapts these tools for analyzing components with weight changes.

Fact learning in LMs. Fact learning in LMs and its robustness has been studied in the pretraining (Chang et al., 2024; Kandpal et al., 2023), finetuning (Berglund et al., 2023a;b), synthetic (Allen-Zhu & Li, 2023; Wang et al., 2024), and knowledge-editing (Cohen et al., 2024; Onoe et al., 2023) settings. Recent works have used influence functions (Qin et al., 2024) and layer-wise ablations (Zhang et al., 2024) to study mechanisms for generalization. In addition, Krasheninnikov et al. (2023) theorized about mechanisms related to extractive structures for their OCR task. Our work provides a more fine-grained analysis by proposing and empirically verifying concrete mechanisms for generalization.

Multi-hop reasoning. Multi-hop reasoning is a common LM evaluation task that requires composing several factual associations together (Zhong et al., 2023; Balesni et al.,

FIRST-HOP dataset	
Fact *(Train)*	(John Doe lives in, Tokyo)
Impl. *(Test)*	(People in the city John Doe is from speak, Japanese)
	John Doe (Head) —new→ Tokyo (Bridge) —known→ Japanese (Tail)
SECOND-HOP dataset	
Fact *(Train)*	(The mayor of Tokyo is, John Doe)
Impl. *(Test)*	(The mayor of the city that contains Sensoji temple is, John Doe)
	Senshoji Temple (Head) —known→ Tokyo (Bridge) —new→ John Doe (Tail)

Table 1: Illustrative examples from the FIRST-HOP and SECOND-HOP datasets. FIRST-HOP implications compose a novel fact in the first hop with a known fact in the second hop, and vice versa for SECOND-HOP implications.

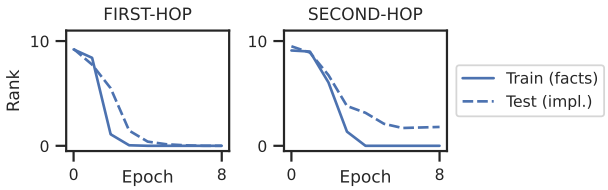

Figure 3: Mean ranks of facts and their implications when finetuning OLMo-7b on facts from the FIRST-HOP and SECOND-HOP datasets. Lower rank is better. The mean rank of implications falls during finetuning; the LM thus generalizes to implications despite only training on facts.

2024). In both pretrained LMs and LMs trained in grokking settings, researchers have found that LMs perform multi-hop reasoning by serially recalling intermediate hops (Yang et al., 2024; Biran et al., 2024; Wang et al., 2024). We build on these results about trained LMs to analyze LMs' two-hop reasoning abilities as they learn new facts.

3. Preliminaries on out-of-context reasoning

In this section we concretely define OCR and describe the two-hop reasoning task we use to instantiate OCR.

OCR is the phenomenon that finetuning a pretrained language model on a set of facts $\mathcal{F}$ leads to the model generalizing to the implications of the finetuned facts $\text{Impl}\,\mathcal{F}$. Concretely, we model each fact $\mathsf{F} \in \mathcal{F}$ as a pair (p, a), where p is a prompt and a is a continuation of the prompt. We model implications similarly. For example, a fact could be $\mathsf{F} =$ ("John Doe lives in", "Tokyo"), and $\text{Impl}\,\mathsf{F} =$ ("The people from the city in which John Doe lives speak", "Japanese").

To measure whether a model has generalized to an implication $\text{Impl}\,\mathsf{F} = (p, a)$, we measure the rank of the continuation a among a set of options. Specifically, we take all possible continuations in the set of implications $\text{Impl}\,\mathcal{F}$ as the set of options. Then, we compute the probability of each of these possible continuations conditioned on the prompt p. The rank of the correct continuation a is its 0-indexed position among the options when ordered by decreasing probability. We use the mean rank when measuring generalization to a set of implications $\text{Impl}\,\mathcal{F}$. Similarly, we use the mean rank for facts $\mathcal{F}$ to verify that the LM has indeed learned facts during finetuning.

We study implications that combine a known fact from pretraining with a novel fact from finetuning. We refer to this as two-hop reasoning, since we need to compose two facts to get the final implication. For example, after teaching the model "John Doe lives in Tokyo", the model can compose this fact with existing knowledge that people in Tokyo speak Japanese to answer the prompt "The people from the city in which John Doe lives speak" with "Japanese". More generally, suppose there are two factual associations, one from head entity a to bridge entity b, and one from bridge entity b to tail entity c. Two-hop reasoning requires the model to compose the two factual associations $a \to b$ and $b \to c$ together so that the model, when prompted with head entity a, recalls the tail entity c.

In two-hop reasoning, the novel fact can be in either the first hop ($a \to b$) or the second hop ($b \to c$), and we construct synthetic datasets, FIRST-HOP and SECOND-HOP, to study each (Table 1). Both datasets have training sets with novel facts and test sets with two-hop implications that require composing the novel training facts with known facts. However, in FIRST-HOP the novel fact comprises the first hop of the two-hop reasoning chain, whereas in SECOND-HOP the novel fact comprises the second. See Sec. D for dataset details. We find that the OLMo-7b model is indeed capable of two-hop OCR for both datasets (Fig. 3).

4. Extractive structures framework

This section describes the extractive structures framework and proposes metrics for identifying extractive structures. Our framework models how weight changes influence behavior via an interplay between weights and activations, and is grounded in empirically measurable metrics.

Extractive structures model OCR as the coordination of three groups of components during the forward pass on the prompt p: the *informative components*, *upstream extractive components* and *downstream extractive components* (Fig. 1).

Informative components store the salient information about newly learned facts as weight updates. While all components undergo weight updates during finetuning, only the weight updates in the informative components are important for correctly predicting implications. In Fig. 1, informative components store the association "John Doe" → "Tokyo" that is later used by the extractive components at test time.

Upstream extractive components parse the input prompt p to produce relevant intermediate activations as inputs to informative components. Upstream components are not importantly changed by finetuning, but are important for correctly activating the informative component at test time[1]. In Fig. 1, upstream components produce the intermediate representation of "John Doe" that informative components convert to "Tokyo" using their newly stored associations.

Downstream extractive components process the outputs of informative components to produce the correct continuation a. In Fig. 1, downstream components look up the language of the city "Tokyo" to predict "Japanese".

For each of the three groups of components (informative, upstream, downstream), we propose a metric to identify them. The metrics are based on causal interventions on the computational graph of the LM (Sec. 4.1). For efficient computation, we linearly approximate the causal metrics (Selvaraju et al., 2017). We additionally show that the linearized scores are first-order perturbations to a single quantity (Sec. A), suggesting that they ought to be comparable in magnitude. We discuss implementation in Sec. B.

4.1. Causal metrics on the computational graph

We define metrics for extractive structures in terms of the LM's computational graph. Denote the inputs and outputs of the LM as x and y. For example, for an implication (p, a), x would be the prompt p, and y would be a probability distribution over possible answers. We measure whether the LM has correctly answered the prompt with a real-valued reward $\mathcal{R}(y, a)$ (e.g. log-probability of a). Thus, x is a leaf node and $\mathcal{R}$ is the terminal node in the computational graph.

The internal nodes of the LM computational graph are the inputs and outputs to components in the LM. Following conventions in LM interpretability (Wang et al., 2022; Elhage et al., 2021), we define a component to be either an MLP or attention head at a specific layer and token position of the transformer. Formally, let $\mathtt{C}$ be a component in the LM, $z_\mathtt{C}$ and $y_\mathtt{C}$ be its input and output, $\mathbf{W}_\mathtt{C}$ be its weights, and $f_\mathtt{C}$ describe its computation so that $y_\mathtt{C} = f_\mathtt{C}(z_\mathtt{C}, \mathbf{W}_\mathtt{C})$.

We simplify the full computational graph by focusing on a particular component $\mathtt{C}$ (Fig. 4). We denote the collective weights of components earlier and later than component $\mathtt{C}$ as $\mathbf{W}_{\text{early}}$ and $\mathbf{W}_{\text{late}}$, so that the overall LM weights $\mathbf{W}$ are represented by three leaf nodes $\mathbf{W}_{\text{early}}$, $\mathbf{W}_\mathtt{C}$, and $\mathbf{W}_{\text{late}}$.

Using this computational graph, we can express precisely the causal effects we expect if component $\mathtt{C}$ is one of the three extractive structures defined above. We fix the input x and desired output a, and measure how the reward $\mathcal{R}$ varies under certain interventions on the graph. We denote an intervention that sets the value of a node v to a counterfactual value v' with $v \leftarrow v'$, and the resulting reward as $\mathcal{R}[v \leftarrow v']$. We also denote the original pretrained weights as $\mathbf{W}$ and the finetuned weights as $\mathbf{W}'$, and similarly denote node values in the finetuned model as their primed versions (e.g. $z'_\mathtt{C}$).

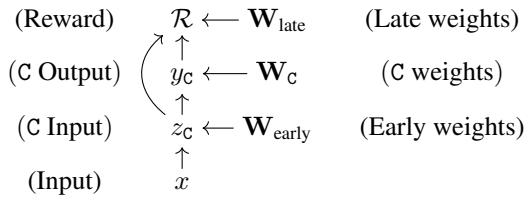

Figure 4: Computational graph of a LM, focusing on component $\mathtt{C}$. Components in earlier and later layers are folded into $\mathbf{W}_{\text{early}}$ and $\mathbf{W}_{\text{late}}$ respectively. The direct arrow from $z_\mathtt{C}$ to $\mathcal{R}$ models skip connections in transformers.

Below we describe interventions that capture how each of the three structures mediate changes to the weights.

Informative components. Intuitively, an informative component contains the newly learned knowledge that the rest of the network extracts. Therefore, if component $\mathtt{C}$ is an informative component, we expect setting $\mathbf{W}_\mathtt{C} \leftarrow \mathbf{W}'_\mathtt{C}$ while leaving all other weights unchanged to improve the reward. We thus write the informative score for component $\mathtt{C}$ as:

$$\mathcal{I}_\mathtt{C} = \mathcal{R}[\mathbf{W}_\mathtt{C} \leftarrow \mathbf{W}'_\mathtt{C}] - \mathcal{R} \tag{1}$$

Downstream components. Intuitively, a downstream component processes the output containing the newly learned knowledge into useful output for the current prediction. Therefore, if component $\mathtt{C}$ is a downstream component, we expect its output $y_\mathtt{C}$ to help increase reward, but only if its input $z_\mathtt{C}$ contains the newly learned knowledge. We therefore compute the new $z'_\mathtt{C}$ with the new weights $\mathbf{W}'_{\text{early}}$ and contrast it with the original $z_\mathtt{C}$. Specifically, we compute the downstream score for component $\mathtt{C}$ as:

$$\mathcal{D}_\mathtt{C} = \mathcal{R}[y_\mathtt{C} \leftarrow f_\mathtt{C}(z'_\mathtt{C}, \mathbf{W}_\mathtt{C})] - \mathcal{R} \tag{2}$$

Upstream components. Intuitively, an upstream component needs to process the input to appropriately query the informative component. Therefore, if component $\mathtt{C}$ is an upstream component, we expect its outputs to feed into later layers to increase reward, but only if the later layers contain the updated weights. The upstream score will thus be a difference in differences: one difference that measures the importance of component $\mathtt{C}$ for later layers, and another that measures the change from the old to the new weights.

A standard way of measuring importance of a node is to ablate its value by setting it to a constant, such as its mean

[1]Weight changes are empirically often low-rank (Aghajanyan et al., 2020), and would therefore have no influence on component outputs unless component inputs fall in the right subspace.

value over some reference set (Chan et al., 2022). To evaluate the importance of component $\mathtt{C}$, we set its output $y_{\mathtt{C}}$ to a constant value $y_{\mathtt{C}}^*$ and measure the decrease in reward. Formally, we define the importance of component $\mathtt{C}$ as

$$\mathcal{R}_{\mathtt{C}} = \mathcal{R} - \mathcal{R}[y_{\mathtt{C}} \leftarrow y_{\mathtt{C}}^*]. \quad (3)$$

Upstream components have higher importance when future informative layers are updated to the new weights $\mathbf{W}'$. We therefore compute the upstream score for component $\mathtt{C}$ as the difference in the importance $\mathcal{R}_{\mathtt{C}}$ under the new weights $\mathbf{W}'_{\text{late}}$ and the old weights $\mathbf{W}_{\text{late}}$:

$$\mathcal{U}_{\mathtt{C}} = (\mathcal{R}[\mathbf{W}_{\text{late}} \leftarrow \mathbf{W}'_{\text{late}}] - \mathcal{R}[\mathbf{W}_{\text{late}} \leftarrow \mathbf{W}'_{\text{late}}, y_{\mathtt{C}} \leftarrow y_{\mathtt{C}}^*]) - (\mathcal{R} - \mathcal{R}[y_{\mathtt{C}} \leftarrow y_{\mathtt{C}}^*]) \quad (4)$$

5. Extractive structures in two-hop reasoning

We next use the extractive scores from the previous section to identify informative, upstream, and downstream components for the FIRST-HOP and SECOND-HOP datasets. Our results indicate that OCR in two-hop reasoning occurs by recalling each hop sequentially (Sec. 5.1). Moreover, we find that facts are stored redundantly across many LM layers, but early layers enable first-hop generalization while later layers enable second-hop generalization (Sec. 5.2). Our work suggests that understanding OCR mechanisms can help design interventions that determine generalization properties. All code is available at https://github.com/jiahai-feng/extractive-structures/

5.1. Extractive structures perform latent two-hop recall

As described in Sec. 3, two-hop reasoning requires the model to compose two factual associations $a \to b$ and $b \to c$ together so that the model recalls the tail entity c when prompted with head entity a (presumably by latently recalling the bridge entity b). One hypothesized mechanism for this is *latent two-hop* (Yang et al., 2024), which posits two groups of components: one responsible for the **first-hop recall** $a \to b$, and another for the **second-hop recall** $b \to c$.

If LMs use latent two-hop recall for OCR, we expect the extractive scores to correspond to the two groups of components. Because the FIRST-HOP and SECOND-HOP datasets introduce novel facts at different hops, we expect the extractive components to differ. Specifically, for the FIRST-HOP dataset where the novel fact is on the first hop, we expect the informative components to be the first-hop recall components, and the downstream components to include the second-hop recall components. In contrast, for the SECOND-HOP dataset where the novel fact is on the second hop, we expect the upstream components to include the first-hop recall components, and the informative components to be the second-hop recall components.

Setup. We study an intermediate training checkpoint of OLMo-7b, taking the last checkpoint before the final annealing stage because annealing may interfere with finetuning(Ibrahim et al., 2024). For each dataset (FIRST-HOP and SECOND-HOP), we finetune the model on the facts and compute extractive scores on implications by comparing the finetuned weights with the pretrained weights. We provide more training details in Sec. C; in particular we often observe learning rate sensitivity (Sec. H), where OCR only occurs at certain learning rates; we thus focus on the learning rates for which OCR most reliably occurs.

Results. Our computed extractive scores are consistent with latent two-hop recall (Fig. 5). For the FIRST-HOP dataset, high informative scores are localized to the MLPs in early-middle layers of the head entity tokens ("John Doe"), which we interpret as the first-hop recall components. The downstream scores are localized to the late MLPs and attention heads at the last token, which we interpret as second-hop recall components. For the SECOND-HOP dataset, high upstream scores are localized to the early/middle MLPs of the head entity tokens ("Tower Bridge"), matching the first-hop recall components identified in the FIRST-HOP dataset. The informative scores are localized to the late MLP layers in the last token position, matching the second-hop recall components in FIRST-HOP.

These results from analyzing finetuning broadly reinforce findings from prior works that analyze pretrained models. Yang et al. (2024) and Biran et al. (2024) argued that first-hop recall occurs at the early-middle layers of the last head entity token, whereas the second-hop recall occurs at the late layers of the last token. Our extractive scores agree with both claims, except that we also find that both head entity tokens (e.g. "John" and "Doe") store factual information, and not just the last token. Moreover, while prior works (Geva et al., 2023; 2020; Meng et al., 2022) tend to study how MLPs store and recall facts and how attention heads move factual information, we find instead that attention heads can also store factual information. Specifically, the informative scores from both datasets indicate that the late attention heads at the last token position store some factual information. Overall, extractive scores complement conventional interpretability analysis on trained models by leveraging information in *weight changes*, and can verify known phenomena and discover new ones.

5.2. Different layers generalize differently

In this section we rigorously test an implication of the earlier extractive scores (Sec. 5.1): that storing facts in the early-middle layers enables first-hop updates, whereas storing in the late layers enables second-hop updates. Specifically, the informative scores (Fig. 5) show that weight changes to the early-middle MLP layers (first 24) are salient for the FIRST-

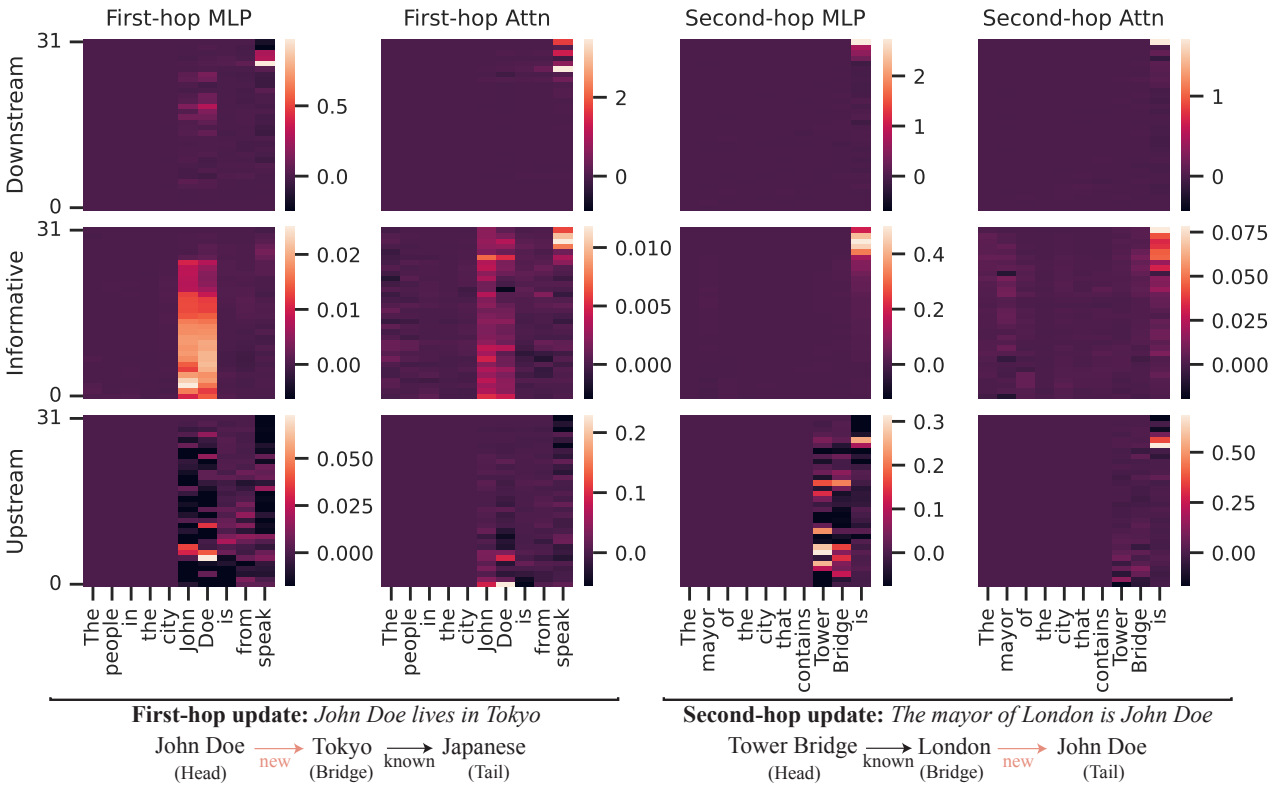

Figure 5: Extractive scores for the FIRST-HOP (left) and SECOND-HOP datasets (right). Scores are averaged over the dataset. We visualize only the scores of the last two entity tokens. The attention scores are summed across all the attention heads that are outputting to the same position. The FIRST-HOP informative scores and SECOND-HOP upstream scores point to the early-middle MLPs at head entity tokens as the first-hop recall components. The FIRST-HOP downstream scores and SECOND-HOP informative scores point to the early-middle MLPs at the last token as the second-hop recall components.

HOP dataset, i.e. novel facts stored in the early-middle layers can be used as the first hop when composed with known second-hop facts. Conversely, weight changes to the late MLP layers (last 8) are salient for the SECOND-HOP dataset, i.e. novel facts stored in the late layers can be used as the second hop when composed with known first-hop facts.

Note that when finetuning on facts, the model does not know whether the facts will later be used as the first hop or second hop in two-hop reasoning; we therefore expect the model to encode facts across all layers (both early-middle and late) to enable different forms of generalization.

To validate our hypothesis, we perform causal experiments where we freeze the weights on certain layers to the original pretrained weights during finetuning and check if the model retains its knowledge of facts and implications. If the extractive scores indeed correctly identified the extractive structures in the model, we expect freezing the early-middle layers to hurt the implications for the FIRST-HOP dataset but not the SECOND-HOP dataset, and vice versa for the late layers. In Sec. E we discuss a variant of this experiment where layers are frozen *after* finetuning instead of *during*.

	FIRST-HOP		SECOND-HOP	
Frozen Layers	Fact	Impl.	Fact	Impl.
None	0.00	0.00	0.00	1.80
Early	0.00	6.50	0.00	0.50
Late	0.00	0.10	0.00	6.60
All	9.20	9.25	9.10	9.50

Table 2: Mean ranks of facts and implications after freezing weights during finetuning. Freezing early layers (first 24) harm first-hop OCR but not second-hop OCR, and vice versa for late layers (last 8). 'None' and 'All' are baselines with full finetuning and no finetuning respectively.

Our results (Table 2) show that, compared to not freezing any layer ("None"), freezing the early-middle layers ("Early") increases the mean rank for FIRST-HOP implications, but not for SECOND-HOP implications. Conversely, freezing the late layers ("Late") increases the mean rank for SECOND-HOP implications but not for FIRST-HOP implications. Moreover, facts are recalled reliably in both settings. This suggests that facts can be stored across many layers,

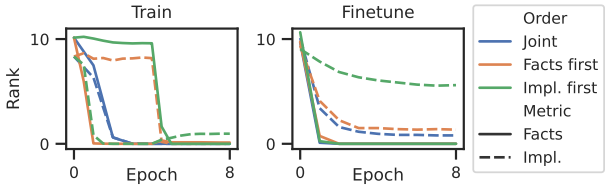

Figure 6: Mean ranks for facts and their implications under continued pretraining on training facts (left) and subsequent finetuning on test facts (right). Compared to other orderings, 'Implications first' generalizes significantly worse from finetuned test facts to their implications.

but different layers enable different forms of generalization.

Our findings suggest that localization matter for knowledge editing, because it influences the generalization properties of the stored facts. Early knowledge editing works (Meng et al., 2022) suggested that facts are stored in a *specific* localized set of components, but Hase et al. (2023) later argued that localization does not matter because facts can be edited at *any* point in the LM. We find that facts should be edited at *every* point in the LM to improve generalization.

6. Origins of extractive structures

We now study how extractive structures are learned during pretraining. For OCR to occur, gradient descent on training facts must encode information in a way that is legible to the extractive structures at inference time; in other words, some mechanism must coordinate between facts encoded at training time and facts latently reasoned about at test time.

We describe a mechanism during pretraining that solves the coordination problem between gradient descent and the extractive components. Suppose at a point during pretraining the model already knows a fact F (i.e. the earlier gradient step on F has already encoded it in the weights), and now encounters its implication $\mathrm{Impl}\,\mathsf{F}$. This could happen by chance from training data shuffling. To learn the implication $\mathrm{Impl}\,\mathsf{F}$, the model could either simply learn to memorize it, or, crucially, learn how to extract fact F from its weights and latently produce the implication $\mathrm{Impl}\,\mathsf{F}$. This creates a training signal for the formation of extractive structures that are adapted to how earlier facts have been encoded.

Our hypothesis is thus that extractive structures are learned when encountering implications of already-known facts during training. We test two of its predictions: a data ordering effect (Sec. 6.1) and a weight grafting effect (Sec. 6.2).

6.1. Data ordering

The hypothesis predicts a data ordering effect during pretraining, where if all the facts appear after their implications, then the model cannot learn extractive structures, and hence cannot later generalize to implications of new facts. Conversely, if all facts precede their implications, then extractive structures may form and later enable OCR.

The data ordering effect suggests a departure from the standard statistical machine learning view of optimization as model selection from a family. Instead, optimization is a dynamical process during which learning depends on the model's internal state: whether the model learns extractive structures from implications depends on whether the model already knows the underlying facts.

Concretely, we study a controlled synthetic setup where a pretrained model is finetuned to learn a new form of fictitious implication. This finetuning process simulates what would have happened if the pretraining data had contained these fictitious implications. For clarity we call this phase *continued pretraining*. If continued pretraining successfully creates new extractive structures, we expect further finetuning the new model on new facts to generalize to their corresponding fictitious implications.

Setup. The new form of implications are derived from fictitious associations, whereas earlier implications are derived from known associations (e.g. "Tokyo" $\rightarrow$ "Japanese"). Concretely, we assign random animals to cities (e.g. "Tokyo" $\rightarrow$ "tiger"), and create fictitious relations "dax" and "wug" so that "John Doe dax Tokyo" implies "John Doe wug the tiger". We teach the model this new form of implications by running continued pretraining on a train dataset of "dax" facts and corresponding "wug" implications. We then evaluate the model's OCR ability by further finetuning the model on held-out test "dax" facts. These test "dax" facts consist of held-out, unseen names, but uses the same set of cities as the train set. If the continued pretraining has successfully created new extractive structures, we expect the finetuned model generalize to test "wug" implications.

We investigate the model's OCR ability after continued pretraining with three orderings of the training data: facts-then-implications ('Facts first'), implications-then-facts ('Impl. first'), and a setting where facts and implications are shuffled together ('Joint'). Further details are in Sec. F.

Results. Figure 6 displays results for the OLMo-7b model from Sec. 5.1; see Sec I for results on additional models (Llama 3-8b, Gemma 2-9b, and Qwen 2-7b). Models continually pretrained on 'Joint' and 'Facts-first' exhibit significantly more out-of-context reasoning than for the 'Impl-first' data order (Fig. 6 right). This happens even though the model successfully learns the training facts and implications in all three data orderings (Fig. 6 left). This supports our hypothesis that extractive structures are learned when training on implications of already known facts.

Finally, the 'Impl-first' model exhibits a small amount of OCR generalization, since the implication mean rank de-

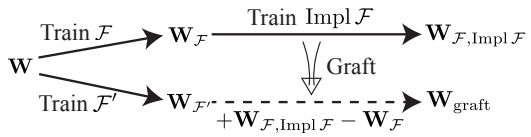

Figure 7: Visualization of grafting procedure. We hypothesize that the grafted weights contain extractive structures for the new form of implications, so that $\mathbf{W}_{\text{graft}}$ generalizes to the counterfactual implications $\mathrm{Impl}\,\mathcal{F}'$.

creases during finetuning (Fig. 6 right). This suggests an additional, unknown mechanism that contributes to OCR.

6.2. Weight grafting

In the data order where all facts precede implications, our hypothesis suggests that facts are learned in the first half of continued pretraining, and the extractive structures are learned in the second half. We therefore expect the change in weights between the middle and final checkpoints of continued pretraining to carry the extractive structures.

We test this by grafting this change in weights onto a model trained on counterfactual facts, and testing the model on counterfactual implications. For example, suppose "John Doe dax Tokyo" and "John Doe wug Tiger" are a fact and implication in the training dataset. If the change in weights carries the extractive structures that converts "Tokyo" to "Tiger", then grafting this change in weights to a model counterfactually trained on "Jane Doe dax Tokyo" should generalize to "Jane Doe wug Tiger." Concretely, we (Fig. 7):

1. Train on facts $\mathcal{F}$ (the dax statements), to produce weights $\mathbf{W}_{\mathcal{F}}$
2. Train on implications $\mathrm{Impl}\,\mathcal{F}$ (the wug statements) to produce weights $\mathbf{W}_{\mathcal{F},\mathrm{Impl}\,\mathcal{F}}$
3. Compute the difference in weights $\mathbf{W}_{\mathcal{F},\mathrm{Impl}\,\mathcal{F}} - \mathbf{W}_{\mathcal{F}}$
4. Train another copy of the model on counterfactual facts $\mathcal{F}'$ to produce $\mathbf{W}_{\mathcal{F}'}$
5. Graft the weight difference over to produce the weights $\mathbf{W}_{\text{graft}} = \mathbf{W}_{\mathcal{F}'} + \mathbf{W}_{\mathcal{F},\mathrm{Impl}\,\mathcal{F}} - \mathbf{W}_{\mathcal{F}}$

If our hypothesis is true, we expect the final model $\mathbf{W}_{\text{graft}}$ to generalize to counterfactual implications $\mathrm{Impl}\,\mathcal{F}'$ instead of the original implications $\mathrm{Impl}\,\mathcal{F}$. We measure this by computing the mean rank of the implications as before.

Setup. We use the same train and test datasets as the data ordering experiment (Sec. 6.1), but additionally create a counterfactual version of the train dataset by re-assigning random cities/animals to the 80 names. We show results for the OLMo-7b model here, and others in Sec. I. As a control, we also graft the change in weights from training the model directly on the implications $\mathrm{Impl}\,\mathcal{F}$, which we expect to only transfer implications and not counterfactual implications: $\mathbf{W}_{\text{control}} = \mathbf{W}_{\mathcal{F}'} + \mathbf{W}_{\mathrm{Impl}\,\mathcal{F}} - \mathbf{W}$.

Weights	$\mathrm{Impl}\,\mathcal{F}'$	$\mathrm{Impl}\,\mathcal{F}$	$\mathcal{F}'$
$\mathbf{W}_{\mathcal{F}'}$	9.13	8.55	0.00
$\mathbf{W}_{\text{graft}}$	1.13	3.30	0.10
$\mathbf{W}_{\text{control}}$	8.38	0.43	0.32

Table 3: Mean ranks of models on counterfactual implications, original implications, and counterfactual facts. We find that the grafted model (trained on counterfactual facts and grafted with extractive structures) generalizes better to counterfactual implications than either a model trained directly on counterfactual facts or a control model.

Results. We report results in Table 3. The weight graft $\mathbf{W}_{\text{graft}}$ lowers the mean rank for counterfactual implications $\mathrm{Impl}\,\mathcal{F}'$ from 9.13 to 1.13, versus 8.38 for the control. This suggests the weight graft contains extractive structures that can elicit counterfactual implications from counterfactual facts. The weight graft also transferred some of the original implications (mean rank 3.30), suggesting that when trained on implications, models partially memorize the implications in addition to learning extractive structures.

Overall, our weight grafting experiments directly identify the weights of the learned extractive structures, thereby supporting our hypothesis on how pretraining learns extractive structures. In Sec. G we further analyze these weights with extractive scores (Sec. 4) to show that the downstream extractive structures are indeed carried by the weight change. Our experiments demonstrate that extractive structures are a useful language for describing OCR training dynamics.

7. Discussion

This work presents an in-depth analysis of how LMs can perform OCR in the two-hop reasoning task. We introduce extractive structures to describe how facts are encoded and recalled to predict implications, and empirically identify them in the OLMo-7b model. Additionally, we propose a hypothesis for how extractive structures are learned in pretraining, and test two predictions of this hypothesis.

Our empirical results raise several immediate research questions. While the experiments in Sec. 6.1 hint at the possible existence of an alternative mechanism underlying OCR, its nature remains unclear. Furthermore, we have yet to fully characterize how the finetuning process depends on various hyperparameters and the pretrained model (Sec. H and I).

More broadly, we hope that extractive structures can lead to an underlying theory of generalization that captures the dynamical nature of optimization (Sec. 6.1). While our work focuses on two-hop reasoning, extractive structures could be useful more broadly for understanding and conceptually grounding out-of-context generalization. Ultimately, iden-

tifying the principles and mechanisms behind OCR could offer insights into deep learning generalization, enabling the design of robust and safe machine learning systems.

Acknowledgements

We thank Amil Dravid, Alex Pan, Felix Binder, Owain Evans, Lijie Chen, Alex Mallen, and Dmitrii Krasheninnikov for helpful feedback on the manuscript. JF acknowledges support from the OpenAI Superalignment Fellowship. JS was supported by the National Science Foundation under Grants No. 2031899 and 1804794. In addition, we thank Open Philanthropy for its support of both JS and the Center for Human-Compatible AI.

Impact Statement

This paper aims to deepen our understanding of empirical deep learning phenomena. We believe such understanding may potentially lead to stronger theories of deep learning generalization, which could help society develop and deploy deep learning systems more safely.

References

Aghajanyan, A., Zettlemoyer, L., and Gupta, S. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*, 2020.

Allen-Zhu, Z. and Li, Y. Physics of language models: Part 3.2, knowledge manipulation. *arXiv preprint arXiv:2309.14402*, 2023.

Balesni, M., Korbak, T., and Evans, O. The two-hop curse: Llms trained on a-¿ b, b-¿ c fail to learn a–¿ c. *arXiv preprint arXiv:2411.16353*, 2024.

Berglund, L., Stickland, A. C., Balesni, M., Kaufmann, M., Tong, M., Korbak, T., Kokotajlo, D., and Evans, O. Taken out of context: On measuring situational awareness in llms. *arXiv preprint arXiv:2309.00667*, 2023a.

Berglund, L., Tong, M., Kaufmann, M., Balesni, M., Stickland, A. C., Korbak, T., and Evans, O. The reversal curse: Llms trained on" a is b" fail to learn" b is a". *arXiv preprint arXiv:2309.12288*, 2023b.

Biran, E., Gottesman, D., Yang, S., Geva, M., and Globerson, A. Hopping too late: Exploring the limitations of large language models on multi-hop queries. *arXiv preprint arXiv:2406.12775*, 2024.

Cammarata, N., Carter, S., Goh, G., Olah, C., Petrov, M., Schubert, L., Voss, C., Egan, B., and Lim, S. K. Thread: circuits. *Distill*, 5(3):e24, 2020.

Chan, L., Garriga-Alonso, A., Goldowsky-Dill, N., Greenblatt, R., Nitishinskaya, J., Radhakrishnan, A., Shlegeris, B., and Thomas, N. Causal scrubbing: A method for rigorously testing interpretability hypotheses. In *AI Alignment Forum*, pp. 10, 2022.

Chang, H., Park, J., Ye, S., Yang, S., Seo, Y., Chang, D.-S., and Seo, M. How do large language models acquire factual knowledge during pretraining? *arXiv preprint arXiv:2406.11813*, 2024.

Cohen, R., Biran, E., Yoran, O., Globerson, A., and Geva, M. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 12:283–298, 2024.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Geva, M., Schuster, R., Berant, J., and Levy, O. Transformer feed-forward layers are key-value memories. *ArXiv*, abs/2012.14913, 2020. URL https://api.semanticscholar.org/CorpusID:229923720.

Geva, M., Bastings, J., Filippova, K., and Globerson, A. Dissecting recall of factual associations in auto-regressive language models. *arXiv preprint arXiv:2304.14767*, 2023.

Groeneveld, D., Beltagy, I., Walsh, P., Bhagia, A., Kinney, R., Tafjord, O., Jha, A. H., Ivison, H., Magnusson, I., Wang, Y., et al. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*, 2024.

Grosse, R., Bae, J., Anil, C., Elhage, N., Tamkin, A., Tajdini, A., Steiner, B., Li, D., Durmus, E., Perez, E., et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

Hase, P., Bansal, M., Kim, B., and Ghandeharioun, A. Does localization inform editing? surprising differences in causality-based localization vs. *Knowledge Editing in Language Models*, 2023.

Ibrahim, A., Thérien, B., Gupta, K., Richter, M. L., Anthony, Q., Lesort, T., Belilovsky, E., and Rish, I. Simple and scalable strategies to continually pre-train large language models. *arXiv preprint arXiv:2403.08763*, 2024.

Kandpal, N., Deng, H., Roberts, A., Wallace, E., and Raffel, C. Large language models struggle to learn long-tail knowledge. In *International Conference on Machine Learning*, pp. 15696–15707. PMLR, 2023.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kramár, J., Lieberum, T., Shah, R., and Nanda, N. Atp*: An efficient and scalable method for localizing llm behaviour to components. *arXiv preprint arXiv:2403.00745*, 2024.

Krasheninnikov, D., Krasheninnikov, E., Mlodozeniec, B. K., Maharaj, T., and Krueger, D. Implicit meta-learning may lead language models to trust more reliable sources. In *Forty-first International Conference on Machine Learning*, 2023.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.

Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., and Mordvintsev, A. The building blocks of interpretability. *Distill*, 3(3):e10, 2018.

Onoe, Y., Zhang, M. J., Padmanabhan, S., Durrett, G., and Choi, E. Can lms learn new entities from descriptions? challenges in propagating injected knowledge. *arXiv preprint arXiv:2305.01651*, 2023.

Qin, J., Zhang, Z., Han, C., Li, M., Yu, P., and Ji, H. Why does new knowledge create messy ripple effects in llms? *arXiv preprint arXiv:2407.12828*, 2024.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

Treutlein, J., Choi, D., Betley, J., Anil, C., Marks, S., Grosse, R. B., and Evans, O. Connecting the dots: Llms can infer and verbalize latent structure from disparate training data. *arXiv preprint arXiv:2406.14546*, 2024.

Vig, J., Gehrmann, S., Belinkov, Y., Qian, S., Nevo, D., Singer, Y., and Shieber, S. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33: 12388–12401, 2020.

Wang, B., Yue, X., Su, Y., and Sun, H. Grokked transformers are implicit reasoners: A mechanistic journey to the edge of generalization. *arXiv preprint arXiv:2405.15071*, 2024.

Wang, K., Variengien, A., Conmy, A., Shlegeris, B., and Steinhardt, J. Interpretability in the wild: A circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

Yang, S., Gribovskaya, E., Kassner, N., Geva, M., and Riedel, S. Do large language models latently perform multi-hop reasoning? *arXiv preprint arXiv:2402.16837*, 2024.

Zhang, X., Li, M., and Wu, J. Co-occurrence is not factual association in language models. *arXiv preprint arXiv:2409.14057*, 2024.

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

A. Linearizing extractive scores

We linearize the extractive scores to obtain quantities easily computed by a few forward and backward passes (Selvaraju et al., 2017). Specifically, for any node v in the computational graph, we approximate

$$\mathcal{R}[v \leftarrow v_1] - \mathcal{R}[v \leftarrow v_2] \approx \frac{\partial \mathcal{R}}{\partial v}(v_1 - v_2),$$

where the derivative can be evaluated at either v_1 or v_2. In practice, we choose v_1 or v_2 based on computational convenience.

First, we linearize the three extractive scores (Eq. 1, 2, 4) to obtain[2]

$$\overline{\mathcal{U}}_{\mathtt{c}} = (\frac{\partial \mathcal{R}}{\partial y_{\mathtt{c}}}[\mathbf{W}_{\text{late}} \leftarrow \mathbf{W}'_{\text{late}}] - \frac{\partial \mathcal{R}}{\partial y_{\mathtt{c}}})(y_{\mathtt{c}} - y^*_{\mathtt{c}}) \tag{5}$$

$$\overline{\mathcal{I}}_{\mathtt{c}} = \frac{\partial \mathcal{R}}{\partial \mathbf{W}_{\mathtt{c}}}(\mathbf{W}'_{\mathtt{c}} - \mathbf{W}_{\mathtt{c}}) \tag{6}$$

$$\overline{\mathcal{D}}_{\mathtt{c}} = \frac{\partial \mathcal{R}}{\partial y_{\mathtt{c}}}(f_{\mathtt{c}}(z'_{\mathtt{c}}, \mathbf{W}_{\mathtt{c}}) - y_{\mathtt{c}}). \tag{7}$$

For the upstream score $\overline{\mathcal{U}}_{\mathtt{c}}$, we choose $y_{\mathtt{c}}$ as the point to evaluate the derivative. For the informative and downstream scores, we choose the original points $\mathbf{W}_{\mathtt{c}}$ and $y_{\mathtt{c}}$ respectively.

A.1. Linearized scores as perturbations

We shall show that the three linearized scores can be derived as terms in a first order perturbation to a single quantity. This suggests that the three linearized scores ought to be comparable in magnitude. Specifically, consider a small perturbation to the weights $\delta\mathbf{W} = \mathbf{W}' - \mathbf{W}$. Let $\breve{y}_{\mathtt{c}} = \frac{\partial \mathcal{R}}{\partial y_{\mathtt{c}}}$ for convenience. Keeping only first order terms, the three linearized extractive scores become

$$\overline{\mathcal{U}}_{\mathtt{c}} = (\frac{\partial \breve{y}_{\mathtt{c}}}{\partial \mathbf{W}_{\text{late}}}\delta\mathbf{W}_{\text{late}})(y_{\mathtt{c}} - y^*_{\mathtt{c}}) \tag{8}$$

$$\overline{\mathcal{I}}_{\mathtt{c}} = \frac{\partial \mathcal{R}}{\partial \mathbf{W}_{\mathtt{c}}}\delta\mathbf{W}_{\mathtt{c}} \tag{9}$$

$$\overline{\mathcal{D}}_{\mathtt{c}} = \breve{y}_{\mathtt{c}} \nabla_{z_{\mathtt{c}}} f_{\mathtt{c}}(z_{\mathtt{c}}, \mathbf{W}_{\mathtt{c}})\, \delta z_{\mathtt{c}} \tag{10}$$

Consider the importance of component $\mathtt{c}$ (Eq. 3)

$$\mathcal{R}_{\mathtt{c}} = \mathcal{R} - \mathcal{R}[y_{\mathtt{c}} \leftarrow y^*_{\mathtt{c}}].$$

Its linearization, evaluating $\breve{y}_{\mathtt{c}}$ at $y_{\mathtt{c}}$, is

$$\overline{\mathcal{R}}_{\mathtt{c}} = \breve{y}_{\mathtt{c}}(f(z_{\mathtt{c}}, \mathbf{W}_{\mathtt{c}}) - y^*_{\mathtt{c}}). \tag{11}$$

We shall show that under a small perturbation $\delta\mathbf{W}$, the first order terms in $\delta\overline{\mathcal{R}}_{\mathtt{c}}$ contain the three extractive scores. By straightforward product rule, we have:

$$\delta\overline{\mathcal{R}}_{\mathtt{c}} = \delta\breve{y}_{\mathtt{c}}(f_{\mathtt{c}}(z_{\mathtt{c}}, \mathbf{W}_{\mathtt{c}}) - y^*_{\mathtt{c}}) + \breve{y}_{\mathtt{c}}(\nabla_{\mathbf{W}_{\mathtt{c}}} f_{\mathtt{c}}(z_{\mathtt{c}}, \mathbf{W}_{\mathtt{c}})\, \delta\mathbf{W}_{\mathtt{c}}) + \breve{y}_{\mathtt{c}}(\nabla_{z_{\mathtt{c}}} f_{\mathtt{c}}(z_{\mathtt{c}}, \mathbf{W}_{\mathtt{c}})\, \delta z_{\mathtt{c}}).$$

We claim that the second and third terms are exactly the linearized informative score and downstream score, to the first order (Eq. 9, 10). However, first term is related to, but not exactly the linearized upstream score.

The third term is manifestly the downstream score (Eq. 10). To show that the second term is the linearized informative score (Eq. 9), we simply apply the chain rule to $\frac{\partial \mathcal{R}}{\partial \mathbf{W}_{\mathtt{c}}}$ to obtain:

$$\begin{aligned}\overline{\mathcal{I}}_{\mathtt{c}} &= \frac{\partial \mathcal{R}}{\partial \mathbf{W}_{\mathtt{c}}}\delta\mathbf{W}_{\mathtt{c}} \\ &= \breve{y}_{\mathtt{c}} \nabla_{\mathbf{W}_{\mathtt{c}}} f_{\mathtt{c}}(z_{\mathtt{c}}, \mathbf{W}_{\mathtt{c}}) \delta\mathbf{W}_{\mathtt{c}}.\end{aligned}$$

[2]The linearized informative score is related to component-wise influence functions (Grosse et al., 2023).

The first term is subtly different from the first-order linearized upstream score (Eq. 8). Denote the first term as

$$\overline{\mathcal{U}}^*_{\mathtt{C}} = \delta\breve{y}_{\mathtt{C}}(f_{\mathtt{C}}(z_{\mathtt{C}}, \mathbf{W}_{\mathtt{C}}) - y^*_{\mathtt{C}}). \tag{12}$$

Compared to the first-order linearized upstream score (Eq. 8), the first term contains $\delta\breve{y}$ instead of $\frac{\partial \breve{y}_{\mathtt{C}}}{\partial \mathbf{W}_{\text{late}}}\delta\mathbf{W}_{\text{late}}$. To understand the difference, recall that according to the computation graph (Fig. 4), $\mathcal{R}$ is dependent on $z_{\mathtt{C}}$, $\mathbf{W}_{\text{late}}$, and $y_{\mathtt{C}}$, and therefore so is $\breve{y}_{\mathtt{C}}$. Therefore, the term $\delta\breve{y}_{\mathtt{C}}$ may be expanded as:

$$\delta\breve{y}_{\mathtt{C}} = \frac{\partial \breve{y}_{\mathtt{C}}}{\partial \mathbf{W}_{\text{late}}}\delta\mathbf{W}_{\text{late}} + \frac{\partial \breve{y}_{\mathtt{C}}}{\partial z_{\mathtt{C}}}\delta z_{\mathtt{C}} + \frac{\partial \breve{y}_{\mathtt{C}}}{\partial y_{\mathtt{C}}}\delta y_{\mathtt{C}}.$$

The first term is exactly the term from the first-order linearized upstream score (Eq. 8). We can interpret $\breve{y}_{\mathtt{C}}$ as the linearized form of what the subsequent layers do to process $y_{\mathtt{C}}$. The first term therefore captures how the subsequent layers depends on the weight perturbation $\delta\mathbf{W}_{\text{late}}$, which is what the upstream score aims to capture. The second and third terms reflect the dependency of the subsequent layers on $z_{\mathtt{C}}$ and $y_{\mathtt{C}}$, which describe modulatory effects that weight changes in earlier layers $\delta\mathbf{W}_{\text{early}}$ and $\mathbf{W}_{\mathtt{C}}$ have on how the late layers process $y_{\mathtt{C}}$.

To summarize, the linearized importance of component $\mathtt{C}$, under a small perturbation $\delta\mathbf{W}$, can be written:

$$\begin{aligned}\overline{\mathcal{R}}_{\mathtt{C}} &= \overline{\mathcal{U}}^*_{\mathtt{C}} + \overline{\mathcal{I}}_{\mathtt{C}} + \overline{\mathcal{D}}_{\mathtt{C}} \\ &= \overline{\mathcal{U}}_{\mathtt{C}} + \overline{\mathcal{I}}_{\mathtt{C}} + \overline{\mathcal{D}}_{\mathtt{C}} + \frac{\partial \breve{y}_{\mathtt{C}}}{\partial z_{\mathtt{C}}}(y_{\mathtt{C}} - y^*_{\mathtt{C}})\delta z_{\mathtt{C}} + \frac{\partial \breve{y}_{\mathtt{C}}}{\partial y_{\mathtt{C}}}(y_{\mathtt{C}} - y^*_{\mathtt{C}})\delta y_{\mathtt{C}}\end{aligned}$$

B. Implementing extractive scores

In this section, we describe how we implement the extractive scores. By caching activations in the forward pass and gradients in the backward pass, we can efficiently compute the linearized extractive scores (Sec A). Importantly, while the linearized informative score and downstream score can be computed efficiently, the linearized upstream score cannot. We instead use a surrogate quantity that is efficient to compute.

B.1. Choice of reward

We use the log probability of the first token of the continuation, conditioned on the prompt, as the reward. All of our datasets have the property that the continuations have unique first tokens.

B.2. Computing informative score

The linearized informative score (Eq. 6) can be computed with one forward and one backward pass. Recall

$$\overline{\mathcal{I}}_{\mathtt{C}} = \frac{\partial \mathcal{R}}{\partial \mathbf{W}_{\mathtt{C}}}(\mathbf{W}'_{\mathtt{C}} - \mathbf{W}_{\mathtt{C}}).$$

To compute this, can we perform a backward pass on the original weights $\mathbf{W}$, cache the gradient $\frac{\partial \mathcal{R}}{\partial \mathbf{W}_{\mathtt{C}}}$ for every component $\mathtt{C}$, and multiply by the component's corresponding weight change $(\mathbf{W}'_{\mathtt{C}} - \mathbf{W}_{\mathtt{C}})$.

B.3. Computing downstream score

The linearized downstream score (Eq. 7) can be computed with three forward passes and one backward pass. Recall

$$\overline{\mathcal{D}}_{\mathtt{C}} = \frac{\partial \mathcal{R}}{\partial y_{\mathtt{C}}}(f_{\mathtt{C}}(z'_{\mathtt{C}}, \mathbf{W}_{\mathtt{C}}) - y_{\mathtt{C}}).$$

The steps are:

1. Run the model on the original weights $\mathbf{W}$ and cache values of $y_{\mathtt{C}}$ for every component $\mathtt{C}$
2. Run the backward pass to obtain the derivative $\frac{\partial \mathcal{R}}{\partial y_{\mathtt{C}}}$
3. Run the model on new weights $\mathbf{W}'$ to obtain cached values of $z'_{\mathtt{C}}$
4. Compute $f_{\mathtt{C}}(z'_{\mathtt{C}}, \mathbf{W}_{\mathtt{C}})$, and thus $\overline{\mathcal{D}}_{\mathtt{C}}$. The computation in this step required is roughly the same as a forward pass.

B.4. Computing upstream score

The linearized upstream score (Eq. 5) cannot be computed with a constant number of forward/backward passes. Recall that

$$\overline{\mathcal{U}}_{\mathtt{C}} = (\frac{\partial \mathcal{R}}{\partial y_{\mathtt{C}}}[\mathbf{W}_{\text{late}} \leftarrow \mathbf{W}'_{\text{late}}] - \frac{\partial \mathcal{R}}{\partial y_{\mathtt{C}}})(y_{\mathtt{C}} - y^*_{\mathtt{C}}).$$

The reason is that computing $\frac{\partial \mathcal{R}}{\partial y_{\mathtt{C}}}[\mathbf{W}_{\text{late}} \leftarrow \mathbf{W}'_{\text{late}}]$ requires recomputing the every late layer in $\mathbf{W}_{\text{late}}$ for every different component $\mathtt{C}$. However, it turns out that a slightly different quantity can be computed efficiently:

$$\overline{\mathcal{U}}^*_{\mathtt{C}} = (\frac{\partial \mathcal{R}}{\partial y_{\mathtt{C}}}[\mathbf{W}_{\text{late}} \leftarrow \mathbf{W}'_{\text{late}}, y_{\mathtt{C}} \leftarrow y'_{\mathtt{C}}, z_{\mathtt{C}} \leftarrow z'_{\mathtt{C}}] - \frac{\partial \mathcal{R}}{\partial y_{\mathtt{C}}})(y_{\mathtt{C}} - y^*_{\mathtt{C}}). \tag{13}$$

The derivative $\frac{\partial \mathcal{R}}{\partial y_{\mathtt{C}}}[\mathbf{W}_{\text{late}} \leftarrow \mathbf{W}'_{\text{late}}, y_{\mathtt{C}} \leftarrow y'_{\mathtt{C}}, z_{\mathtt{C}} \leftarrow z'_{\mathtt{C}}]$ can be computed simply by running a forward and a backward pass on the new weights $\mathbf{W}'$.

In addition, this version of the upstream score is exactly the first perturbation term (Eq. 12) of the linearized importance discussed in Sec. A.

Therefore, for computation efficiency we use $\overline{\mathcal{U}}^*_{\mathtt{C}}$ as a surrogate for $\overline{\mathcal{U}}_{\mathtt{C}}$. The steps to compute $\overline{\mathcal{U}}^*_{\mathtt{C}}$ are thus:

1. Run the model on the original weights to compute $y_{\mathtt{C}}$
2. Run the backward pass to compute $\frac{\partial \mathcal{R}}{\partial y_{\mathtt{C}}}$
3. Run the model on the new weights $\mathbf{W}'$ to compute $\frac{\partial \mathcal{R}}{\partial y_{\mathtt{C}}}[\mathbf{W}_{\text{late}} \leftarrow \mathbf{W}'_{\text{late}}, y_{\mathtt{C}} \leftarrow y'_{\mathtt{C}}, z_{\mathtt{C}} \leftarrow z'_{\mathtt{C}}]$
4. Compute $\overline{\mathcal{U}}^*_{\mathtt{C}}$. The computation in this step required is roughly the same as a forward pass.

There remains one more detail: the choice for the baseline value $y^*_{\mathtt{C}}$. We choose to perform a mean ablation, so that $y^*_{\mathtt{C}}$ is the average value of $y_{\mathtt{C}}$ across the dataset. To align token positions between input prompts containing entities of varying length, we assume all entities are two tokens long. In cases where entities have more than two tokens, we drop activations from all but the last two entity tokens. We check that no entities have fewer than two tokens.

C. Training details

Model. We primarily use the OLMo-7b-0424 model, step 477000. This is the last intermediate checkpoint before the final annealing phase during which the learning rate is decayed to zero.

Training. Throughout the paper, we finetune the model using the standard cross-entropy loss. We only include the loss on answer tokens. We freeze the embedding and unembedding layers. We use the Adam optimizer (Kingma & Ba, 2014) for 8 epochs at 3×10^{-6} learning rate, momentum $(0.9, 0.999)$, and batch size 8. We find that OCR is sensitive to learning rates, as explored in Sec. H.

D. Dataset details

D.1. FIRST-HOP and SECOND-HOP dataset

The FIRST-HOP and SECOND-HOP datasets are composed from a list of 20 fictitious names, and a list of 20 city-language or city-landmark pairs.

For each of the datasets, we randomly pair fictitious names with cities to populate fact and implication templates.

The list of cities and names are generated from Claude-3.5-sonnet. Here are they:

Grace,Miller
Ethan,Parker
Olivia,Hughes
Jacob,Turner
Ava,Stewart
Noah,Clark
Emma,Howard
Liam,Bennett
Mia,Sanders
Lucas,Foster
Sophia,Hayes
Mason,Brooks
Lily,Cooper
Jackson,Bell
Amelia,Ward
Caleb,Bryant
Chloe,Campbell
Henry,Morgan
Ella,Adams
Owen,Foster

Tokyo,Japan,Japanese,Senso−ji Temple
Beijing,China,Mandarin,Forbidden City
Mumbai,India,Marathi,Gateway of India
Paris,France,French,Eiffel Tower
Berlin,Germany,German,Brandenburg Gate
Moscow,Russia,Russian,St. Basil's Cathedral
Cairo,Egypt,Arabic,Great Pyramid of Giza
Bangkok,Thailand,Thai,Wat Arun
Istanbul,Turkey,Turkish,Blue Mosque
Sao Paulo,Brazil,Portuguese,Ibirapuera Park
Seoul,South Korea,Korean,N Seoul Tower
Rome,Italy,Italian,Colosseum
London,United Kingdom,English,Tower Bridge
Madrid,Spain,Spanish,Plaza Mayor
Athens,Greece,Greek,Acropolis
Hanoi,Vietnam,Vietnamese,Ho Chi Minh Mausoleum
Addis Ababa,Ethiopia,Amharic,Meskel Square
Jakarta,Indonesia,Indonesian,Istiqlal Mosque
Tehran,Iran,Persian,Azadi Tower
Nairobi,Kenya,Swahili,Uhuru Gardens

For the FIRST-HOP dataset, we apply the following templates:

1. Facts: "[Name] lives in", "[city]"
2. Implications: "The people in the city [Name] is from speak", "[language]"

For the SECOND-HOP dataset, we apply the following templates:

1. Facts: "The mayor of [city] is", "[Name]"
2. Implications: "The mayor of the city that contains [landmark] is", "[Name]"

D.2. Fictitious implications

In Sec. 6, we introduced a new dataset with fictitious relations. This requires a pairing between cities and a list of animals. We use the same set of cities from before, and use the set of 20 animals that Zhang et al. (2024) generated.

We apply the following templates:

1. Facts: "[Name] dax", "[city]"
2. Implications: "[Name] zong is the", "[animal]"

Below are the 100 names and the 20 animals. The first 80 names are used for training, and the last 20 are used for testing.

	First-hop		Second-hop	
Frozen Layers	Fact	Impl.	Fact	Impl.
None	0.00	0.00	0.00	1.80
Early (post)	5.10	8.40	0.95	1.85
Late (post)	0.00	0.00	0.10	6.30
All	9.20	9.25	9.10	9.50
Early (pre)	0.00	6.50	0.00	0.50
Late (pre)	0.00	0.10	0.00	6.60

Table 4: Mean ranks of facts and implications when freezing weights post-training or pre-training. Freezing early layers (first 24) harm first-hop OCR but not second-hop OCR, and vice versa for late layers (last 8). 'None' and 'All' are baselines where we use the full finetuned weights and original weights respectively.

Grace,Miller
Ethan,Parker
Olivia,Hughes
Jacob,Turner
Ava,Stewart
Noah,Clark
Emma,Howard
Liam,Bennett
Mia,Sanders
Lucas,Foster
Sophia,Hayes
Mason,Brooks
Lily,Cooper
Jackson,Bell
Amelia,Ward
Caleb,Bryant
Chloe,Campbell
Henry,Morgan
Ella,Adams
Owen,Foster
Abigail,King
Samuel,White
Zoe,Mitchell
Nathan,Carter
Leah,Scott

Elijah,Morris
Madeline,Evans
Daniel,Gray
Hannah,Reed
Cameron,Perry
Natalie,Bryant
Isaac,Peterson
Violet,Phillips
Dylan,Rogers
Charlotte,Brooks
Landon,Harris
Avery,Jenkins
Evan,Parker
Maya,Nelson
Connor,Green
Sydney,Barnes
Julian,Bennett
Kaitlyn,Ross
Logan,Foster
Brooke,Adams
Eli,Sanders
Molly,Cooper
Wyatt,Lee
Tessa,Collins
Blake,Roberts

Madison,Reed
Andrew,Miller
Hailey,Henderson
Matthew,Foster
Sophie,Lawson
Benjamin,Williams
Isabella,Baker
Carter,James
Layla,Murphy
Brayden,Collins
Gabriella,Foster
Aiden,Peterson
Audrey,Jenkins
Joshua,Barnes
Scarlett,Turner
Ryan,Brooks
Aubrey,Hayes
Christopher,Bryant
Harper,Bell
Jason,Mitchell
Madelyn,Phillips
David,Harris
Nora,Rogers
Adam,Campbell
Elise,Ward

Kevin,White
Lucy,Stewart
Brandon,Green
Bella,Parker
Christian,Clark
Clara,Cooper
Tyler,King
Caroline,Morgan
Jordan,Adams
Stella,Scott
Hunter,Morris
Peyton,Lee
Alex,Evans
Elena,Carter
Ian,Foster
Autumn,Gray
Jeremy,Bennett
Lillian,Ross
Nolan,Reed
Morgan,Howard
Gavin,Perry
Paige,Turner
Adrian,Williams
Cora,Jenkins
Parker,Bryant

lion
tiger
elephant
giraffe
zebra
rhinoceros
crocodile
cheetah
antelope
ostrich
monkey
penguin
koala
dolphin
jellyfish
king snake
butterfly
turtle
beaver
squirrel

E. Additional layer freezing experiments

In the main paper (Sec. 5.2) we described a layer freezing experiment where we freeze the weights of certain layers while finetuning the model First-hop and Second-hop facts. We now describe a variant of this experiment where we first finetune all layers of the model, and *after* finetuning, we reset the weights of certain layers to the original pretrained weights.

The difference between freezing post-finetuning and freezing pre-finetuning is that the former tells us where information is stored during the course of standard finetuning, whereas the latter tells us where information can in principle be stored. Similar to freezing pre-finetuning, we expect freezing post-finetuning to affect First-hop implications when the early-middle layers are frozen but not the late layers, and vice versa for Second-hop implications.

Our results (Table 4) indeed show that freezing early-middle layers after finetuning ("Early (post)") harms First-hop implications while enabling Second-hop implications, and vice versa for freezing late layers ("Late (post)"). Further, we find that freezing the early-middle layers partially increases the mean ranks for First-hop facts and Second-hop facts, suggesting that when finetuning on all layers, facts are stored across all layers, but are disproportionately present in the early-middle layers.

F. Data ordering details

In this section we provide details for the data ordering experiments (Sec. 6.1). The dataset construction is discussed in Sec. D.2.

The training hyperparameters is as follows: In all three data orderings (impl-first, fact-first, joint), we ensure that every document (i.e. a fact or an implication) is seen exactly 8 times. In the joint order, all documents are shuffled randomly and trained on for 8 epochs, reshuffling after every epoch. In the fact-first order, we train on facts for 8 epochs, before resetting the optimizer and training on implications for 8 epochs. Impl-first is similar. Note that in Fig. 6, the x-axis is normalized to match the number of training steps across the three data orders.

In all settings, we use training hyperparameters described in Sec. C.

G. Localizing extractive structures in weight grafting

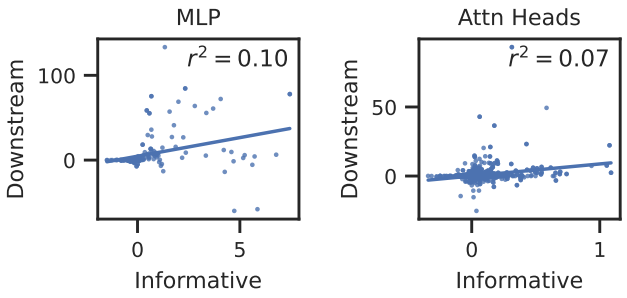

Figure 8: Correlation between downstream extractive scores and informative scores for MLPs (left) and attention heads (right).

Earlier in Sec. 6.2 we showed that a certain weight change contained newly learned extractive structures. In this section, we aim to identify the components that are modified by this weight change, and show that they are exactly the downstream extractive structures.

To identify the components modified by the weight change, we use the informative score. Consider the model finetuned on counterfactual train facts $\mathbf{W}_{\mathcal{F}'_{\text{train}}}$, and a reward defined based on how much the model knows the counterfactual train implications $\text{Impl}\,\mathcal{F}'_{\text{train}}$. Initially, the reward is low because the model only knows the counterfactual train facts $\mathcal{F}'$ but not the train implications $\text{Impl}\,\mathcal{F}'_{\text{train}}$. Suppose that after grafting, the model $\mathbf{W}_{\text{graft}}$ can now predict counterfactual facts $\mathcal{F}'_{\text{train}}$ so that the reward is now high. Then, the informative scores with respect to this change in weights $\mathbf{W}_{\text{graft}} - \mathbf{W}_{\mathcal{F}'_{\text{train}}}$ would identify the important components modified by the weight change, because the informative scores identify components that, under the weight change, contribute to increasing the reward.

To show that these identified components are the downstream extractive structures, we apply the downstream scores to a different weight change. This time, consider the grafted model $\mathbf{W}_{\text{graft}}$. If we finetune this weights on test facts $\mathcal{F}_{\text{test}}$, then the resulting model should perform well on the test implications $\mathcal{F}_{\text{test}}$. The downstream score of this finetuning process should capture the downstream extractive structures that predicts corresponding animals from the cities recalled latently.

We compute the informative and downstream scores for the two weight changes described, and correlate the two approaches for localizing them (Fig. 8). We find that for both MLP components and attention heads, there is a statistically significant positive correlation between the two metrics. However, the Pearson correlation is small, suggesting that the two metrics do not align perfectly.

H. Learning rate sensitivity

We observe that the OCR ability exhibits learning rate sensitivity (Fig. 9). For both learning rates 1e-5 and 3e-6, we observe that the model appears to converge in train rank, but the test rank for 1e-5 is significantly higher than for learning rate 3e-6.

Similarly, data ordering and weight grafting effects are sensitivity to learning rates (Fig. 10, 11). Further, because whether or

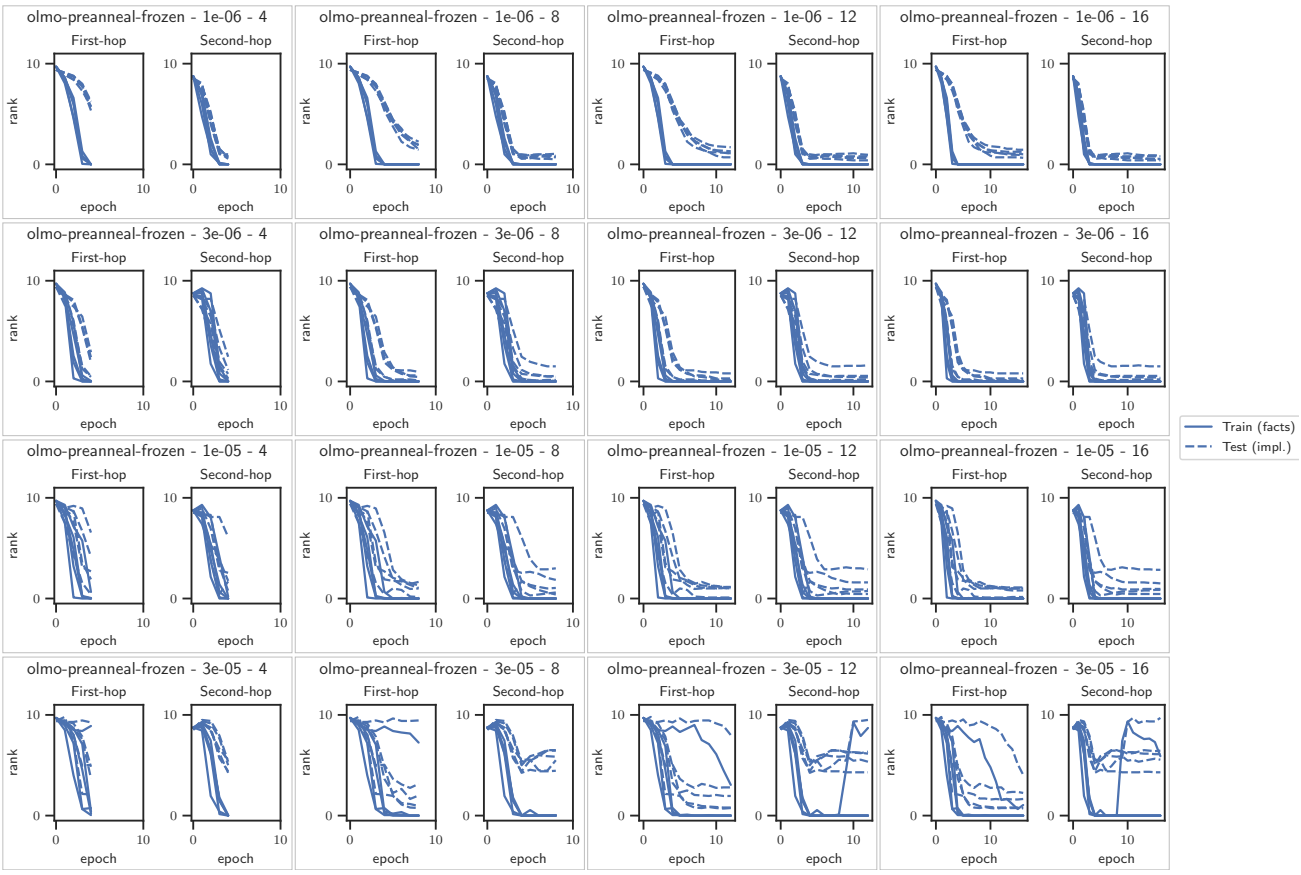

Figure 9: Two-hop OCR performance across different learning rates and epochs for the OLMo-7B pre-anneal checkpoint. 5 random seeds were used. The title has the format "model - lr - epochs"

not extractive structures are learned in continued pretraining depends on the number of epochs of continued pretraining, the OCR performance also depends strongly on the number of epochs. Nonetheless, in most settings we do observe the expected qualitative pattern: that 'Impl-first' data ordering is worse than 'Fact-first' or 'Joint' at OCR, and that $\mathbf{W}_{\text{graft}}$ transfers counterfactual implications but not the control graft $\mathbf{W}_{\text{control}}$.

I. Other models

In this section, we show three plots for three models. The three plots are on two-hop OCR performance, data ordering effect, and weight grafting effect. The three models are Llama-3-8b, Gemma-2-9B, and Qwen-2-7B. As in Sec. H, we sweep across learning rates and epochs, and try 5 random seeds.

Overall, the other models exhibit the same qualitative effects, but the effects are weaker and more sensitive to hyperparameters than in OLMo. We believe that it will be interesting to analyze properties of the models make them less good at OCR.

Two-hop OCR. (Figs. 12, 13, 14) We find that different models require different learning rates to exhibit OCR. All models can perform OCR on the FIRST-HOP dataset on some learning rate values. Performance is worse on SECOND-HOP dataset, but the mean rank does decrease significantly across all seeds.

Data ordering. (Figs. 16, 17, 18) We observe that 'Fact-first' exhibits greater or similar OCR compared with 'Impl-first', suggesting that the data ordering effect is present. For some settings, both 'Fact-first' and 'Impl-first' data orderings have similar, but non-trivial, OCR performance, which again points at the existence of an unknown alternate mechanism for learning extractive structures.

Weight grafting. (Figs. 20, 21, 22) All models, under the weight graft $\mathbf{W}_{\text{graft}}$, for an appropriate choice of hyperparameters,

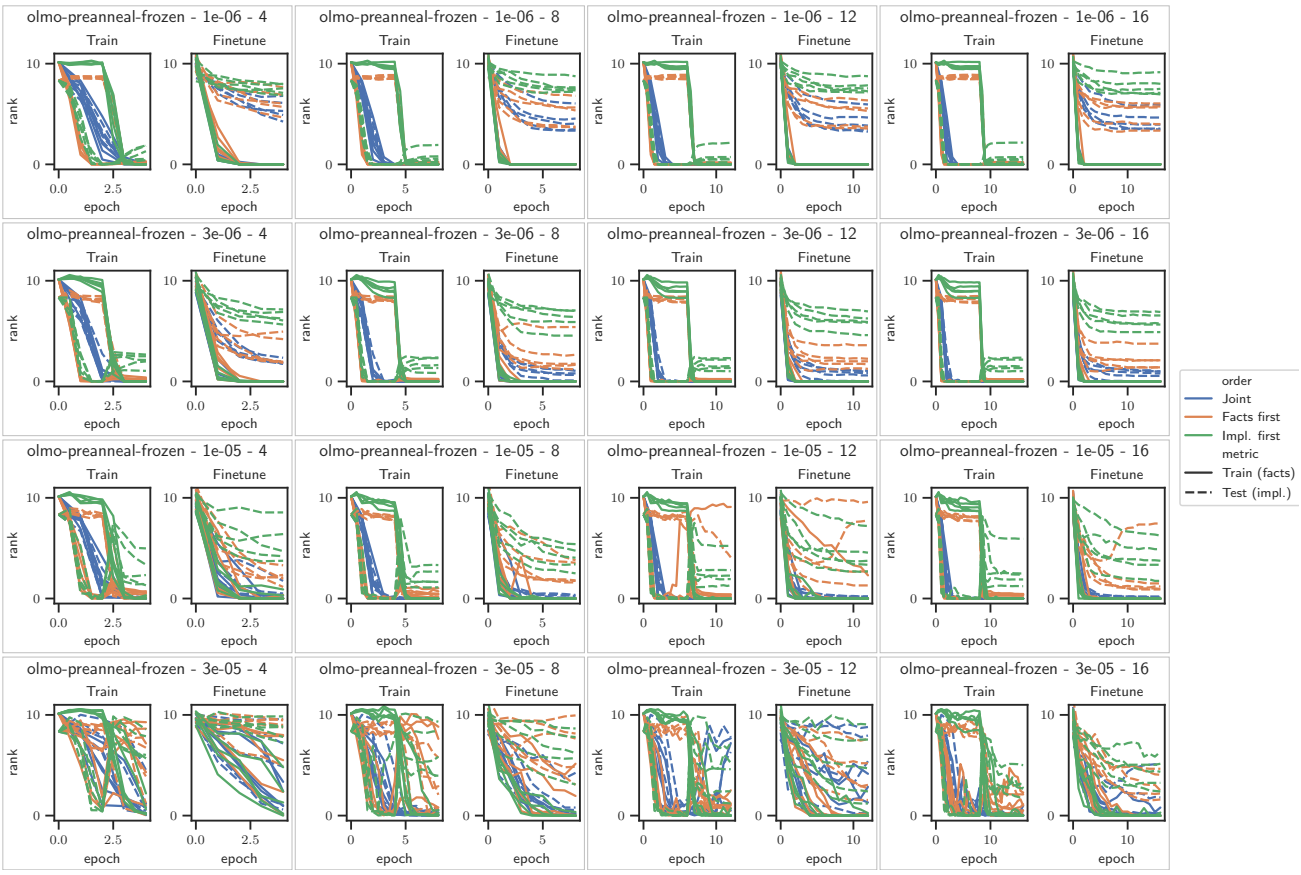

Figure 10: Data ordering effect across different learning rates and epochs for the OLMo-7B pre-anneal checkpoint. 5 random seeds were used. The title has the format "model - lr - epochs"

can predict counterfactual implications. However, in these models the weight graft also transfers the original implications, reinforcing the hypothesis that these models perform OCR less effectively, and instead memorize more.

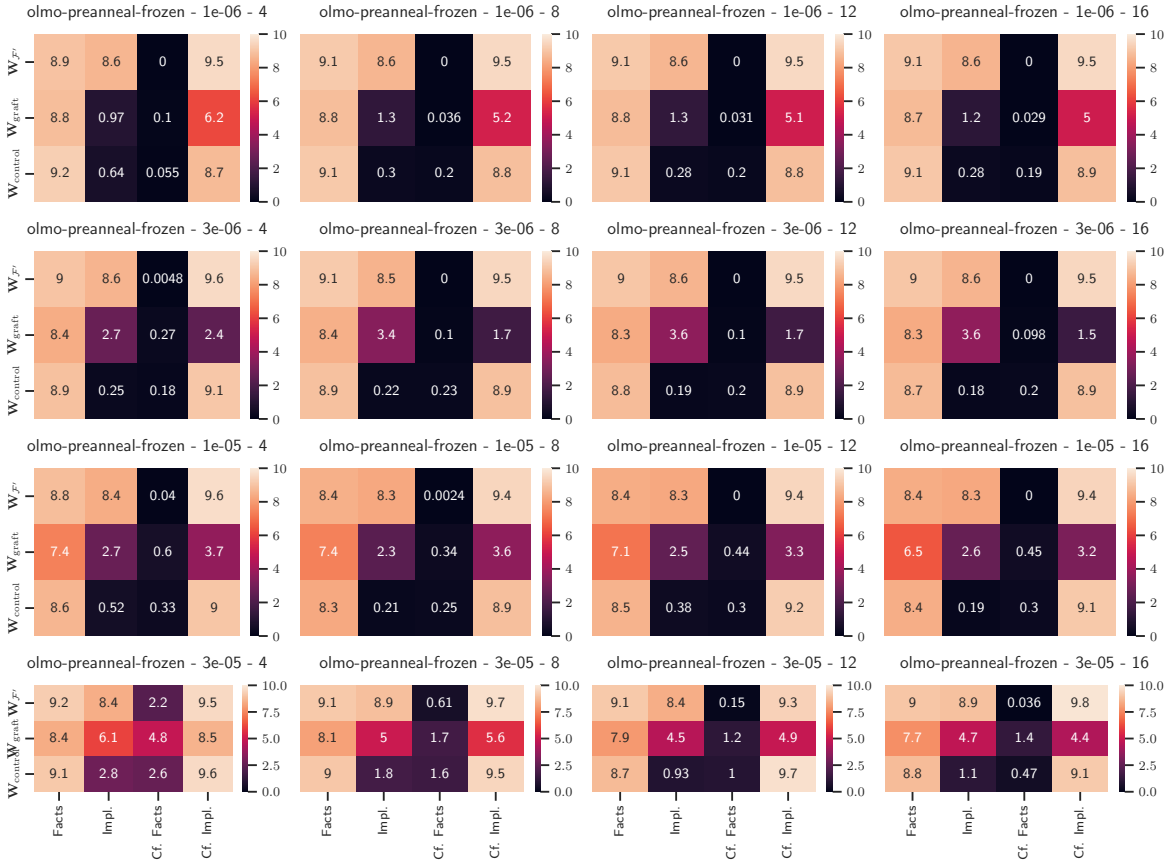

Figure 11: Weight grafting effect across different learning rates and epochs for the OLMo-7B pre-anneal checkpoint. 5 random seeds were used. We show the average mean rank across the 5 random seeds. The title has the format "model - lr - epochs"

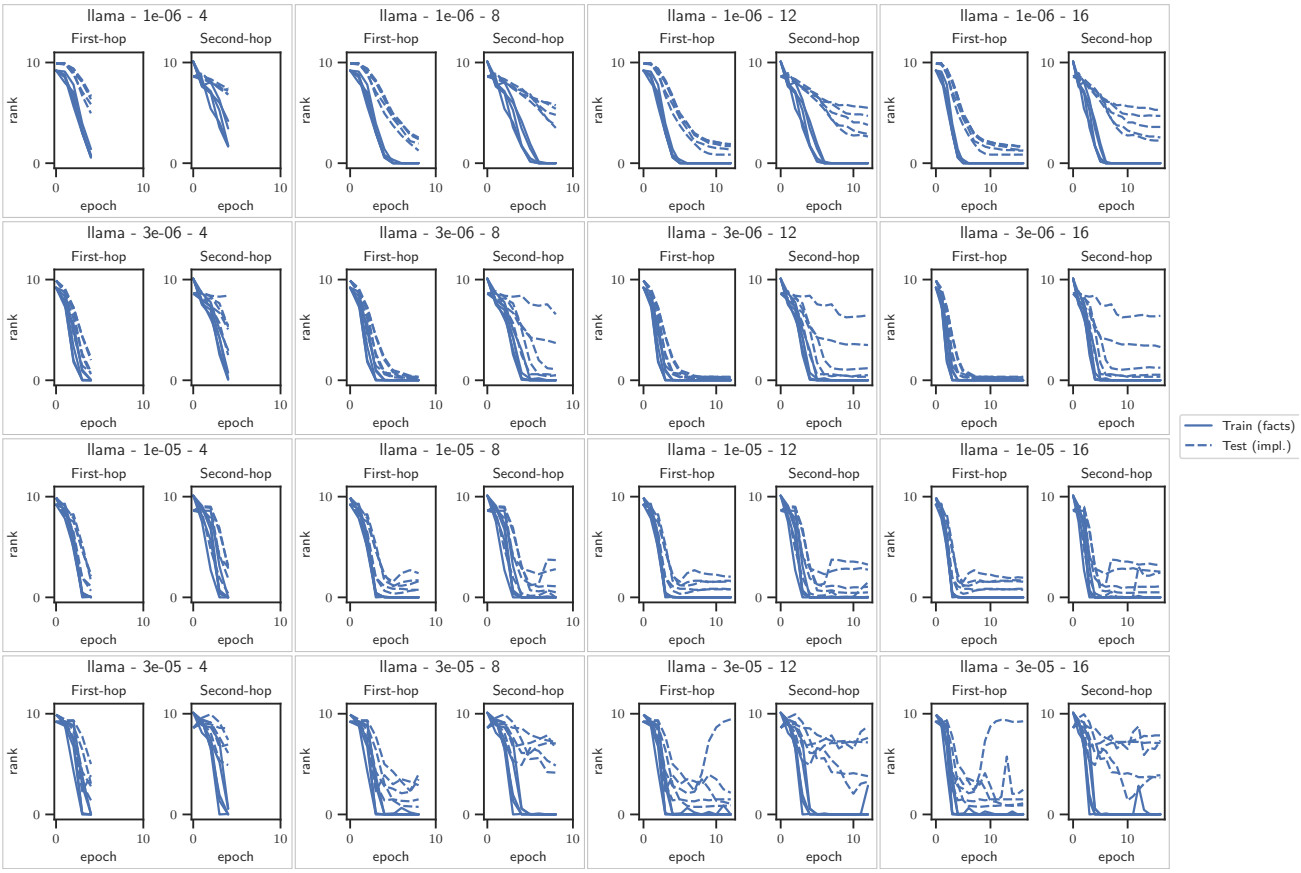

Figure 12: Two-hop OCR performance across different learning rates and epochs for Llama-3-8b. 5 random seeds were used. The title has the format "model - lr - epochs"

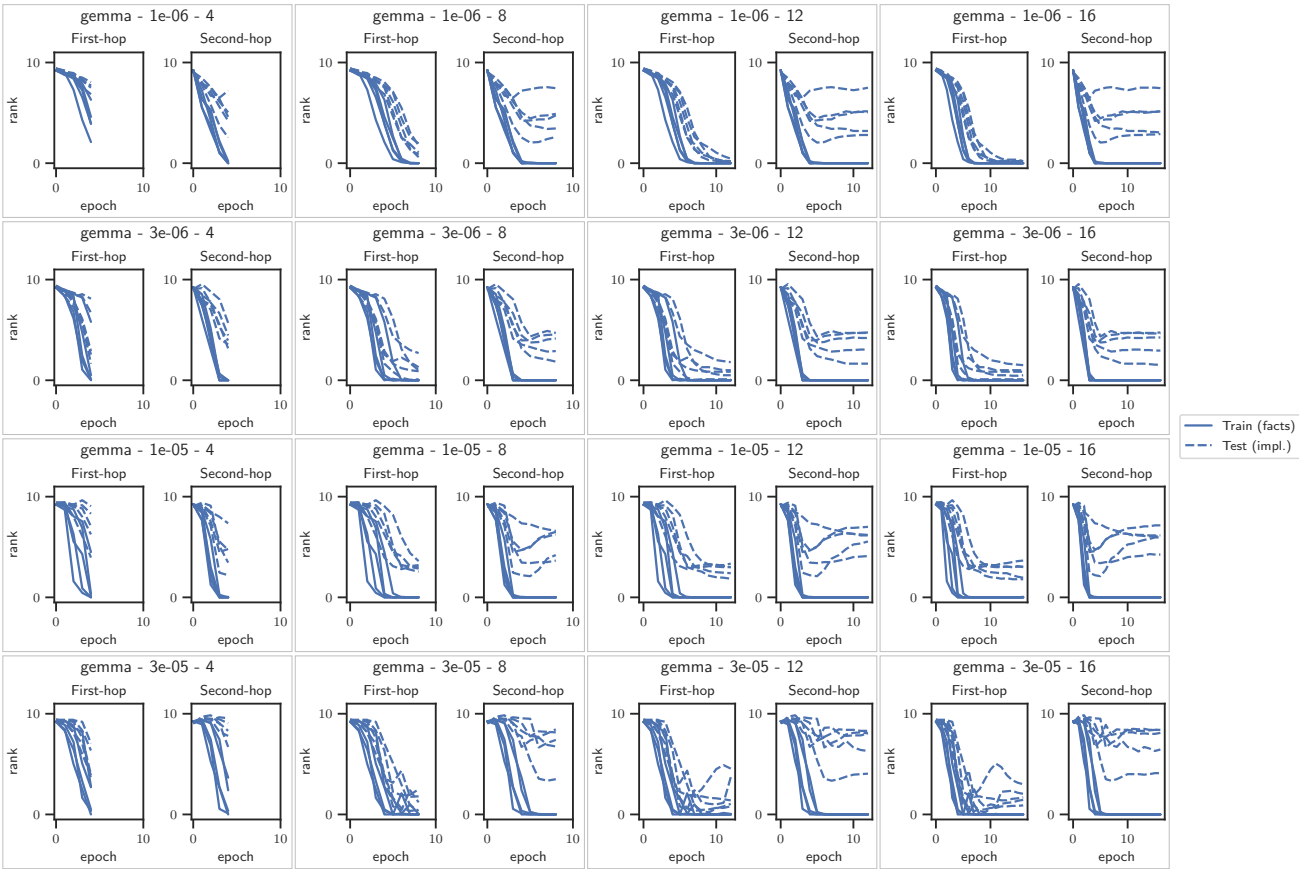

Figure 13: Two-hop OCR performance across different learning rates and epochs for Gemma-2-9B. 5 random seeds were used. The title has the format "model - lr - epochs"

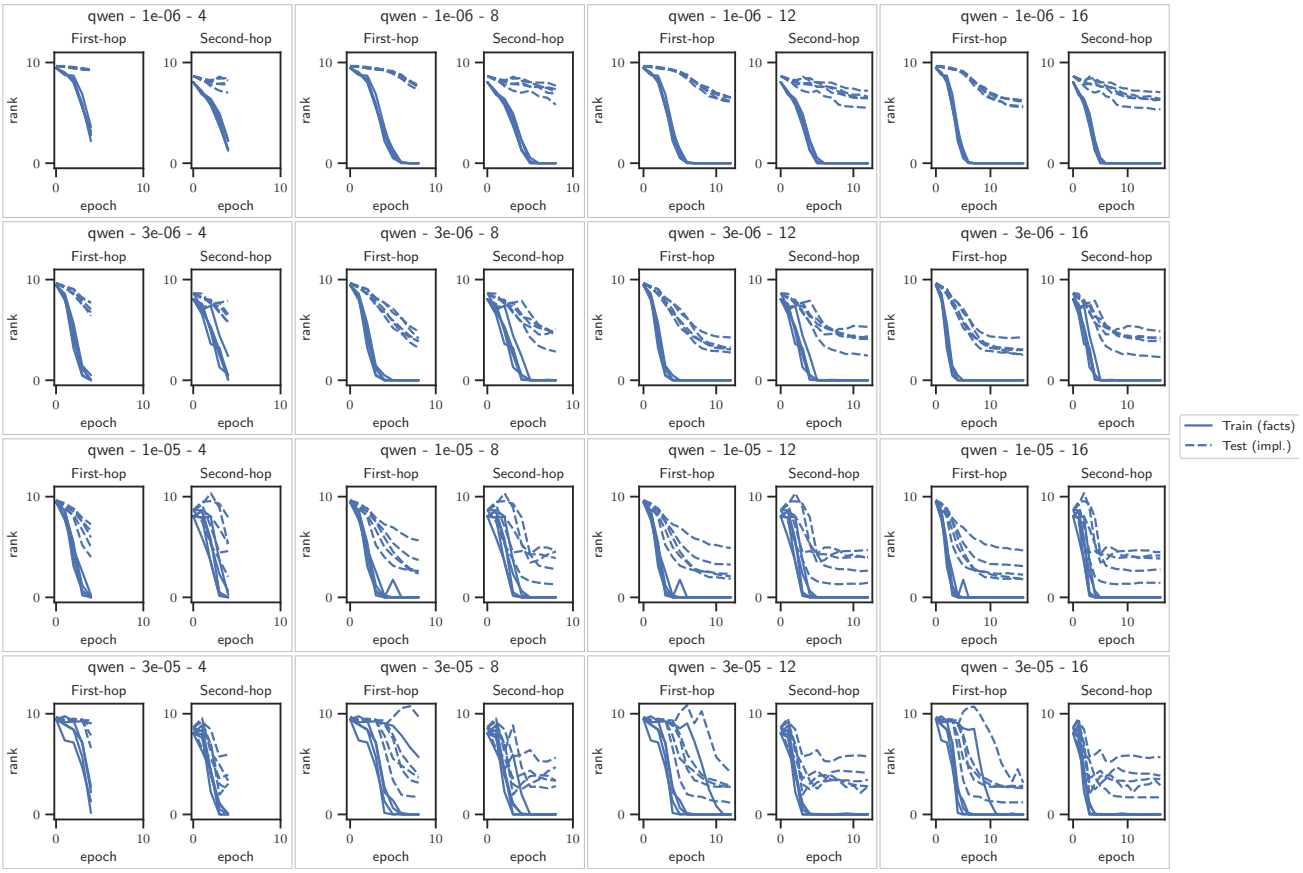

Figure 14: Two-hop OCR performance across different learning rates and epochs for Qwen-2-7B. 5 random seeds were used. The title has the format "model - lr - epochs"

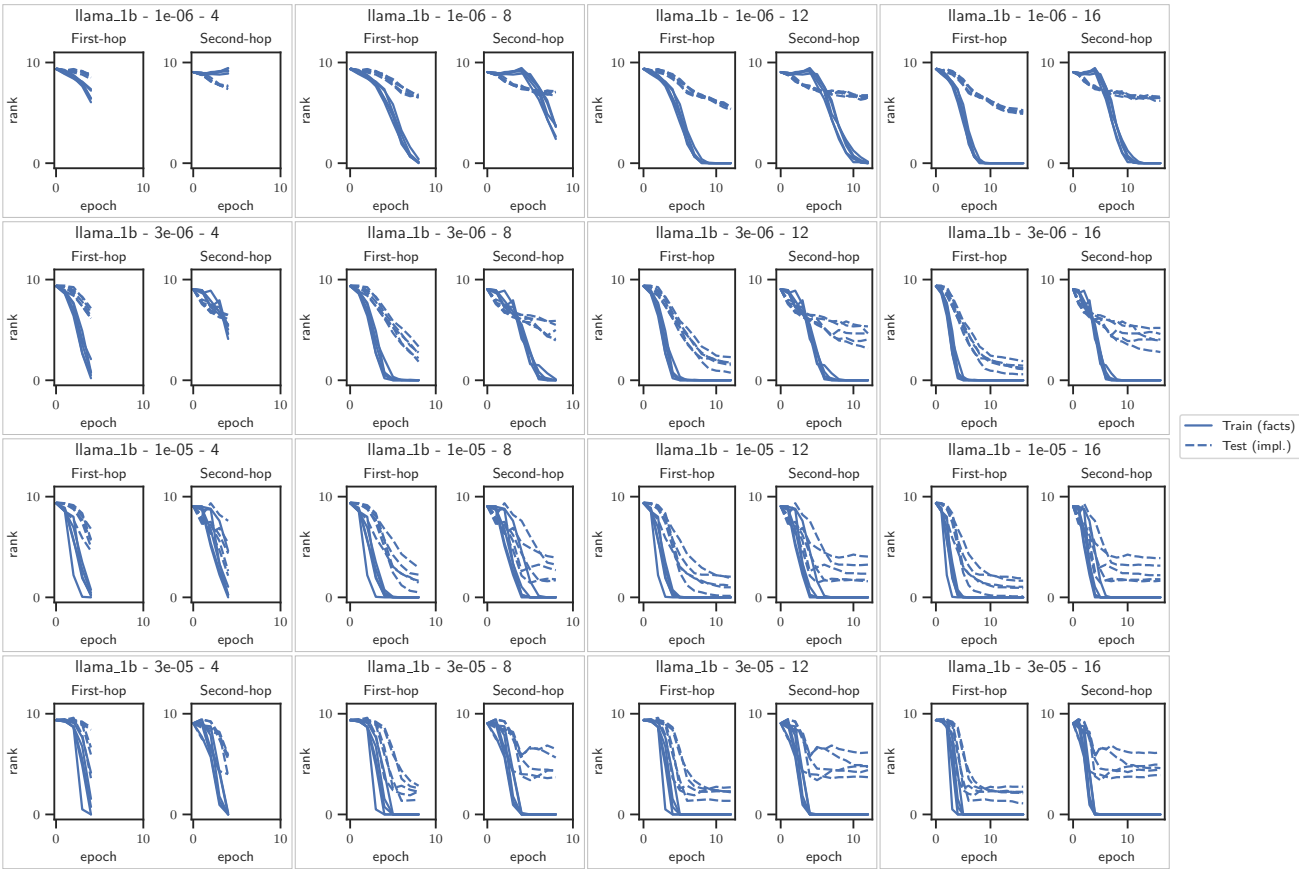

Figure 15: Two-hop OCR performance across different learning rates and epochs for Llama-3.2-1b. 5 random seeds were used. The title has the format "model - lr - epochs"

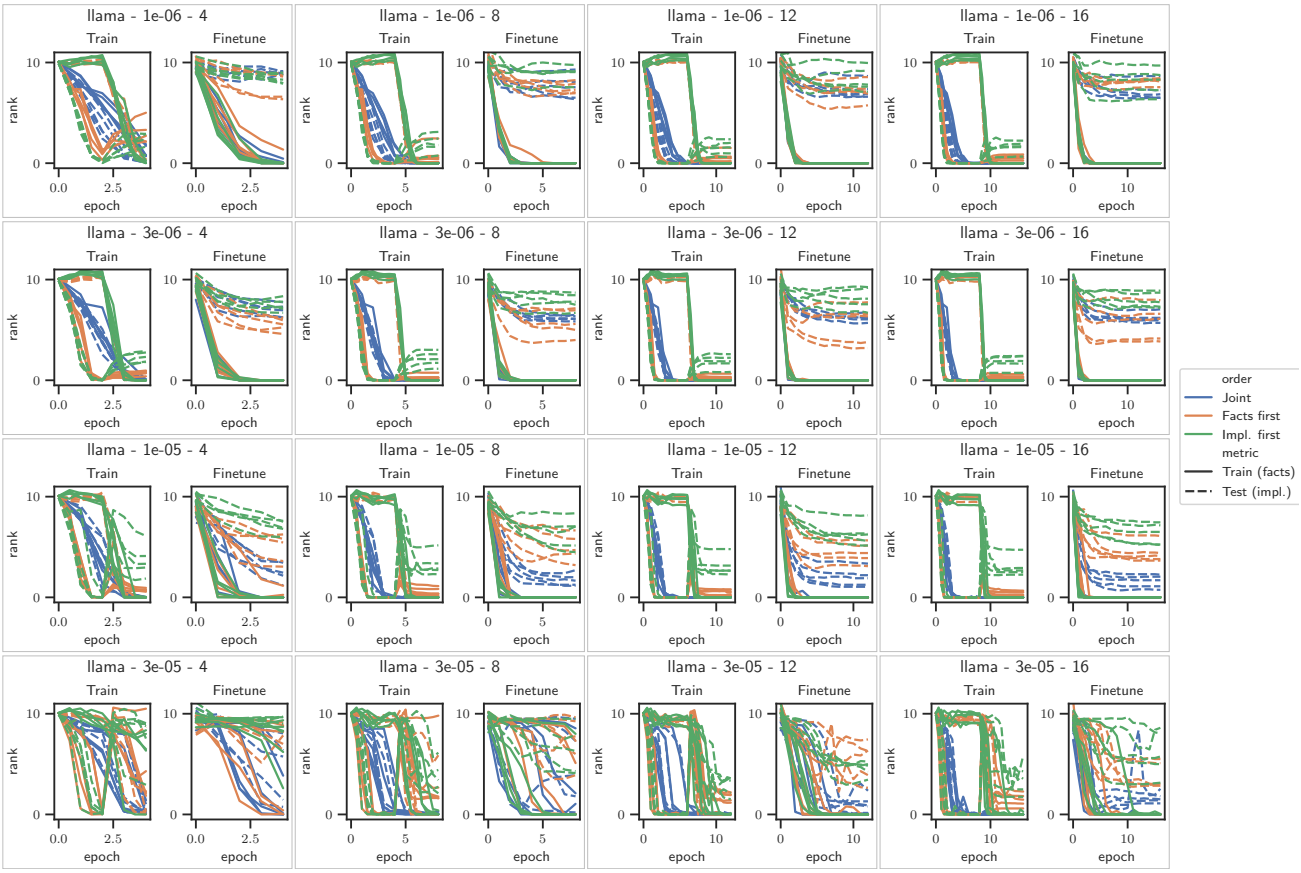

Figure 16: Data ordering effect across different learning rates and epochs for Llama-3-8b. 5 random seeds were used. The title has the format "model - lr - epochs"

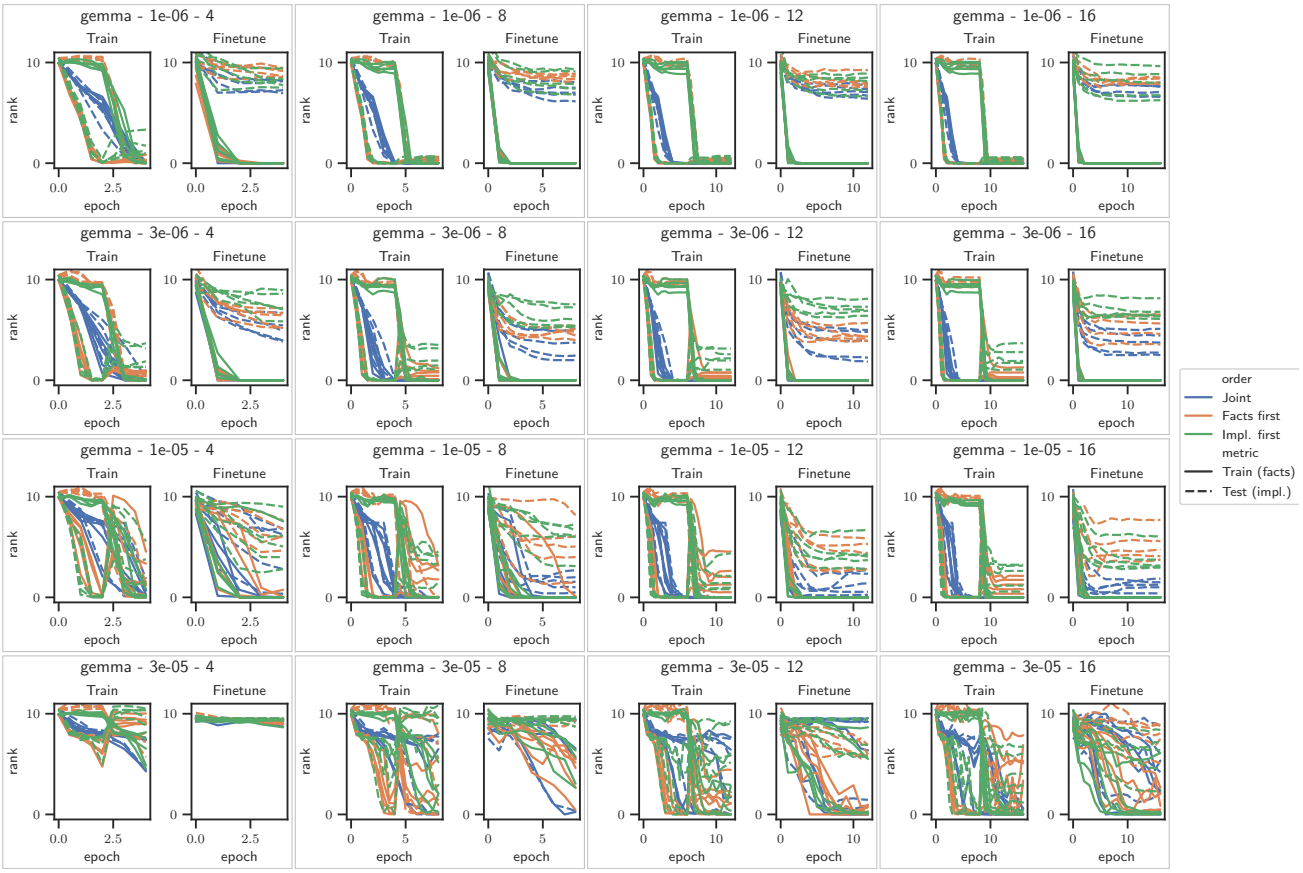

Figure 17: Data ordering effect across different learning rates and epochs for Gemma-2-9B. 5 random seeds were used. The title has the format "model - lr - epochs"

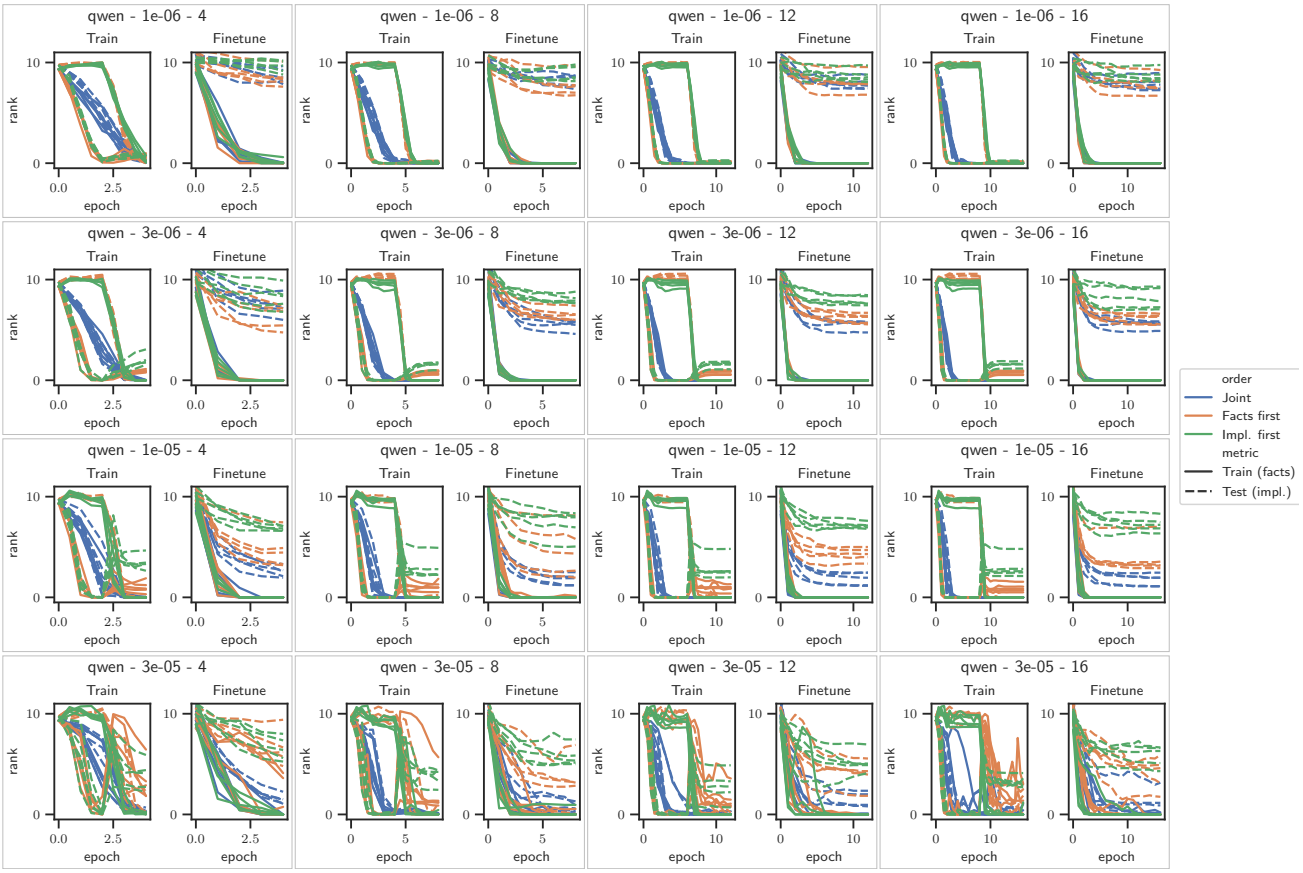

Figure 18: Data ordering effect across different learning rates and epochs for Qwen-2-7B. 5 random seeds were used. The title has the format "model - lr - epochs"

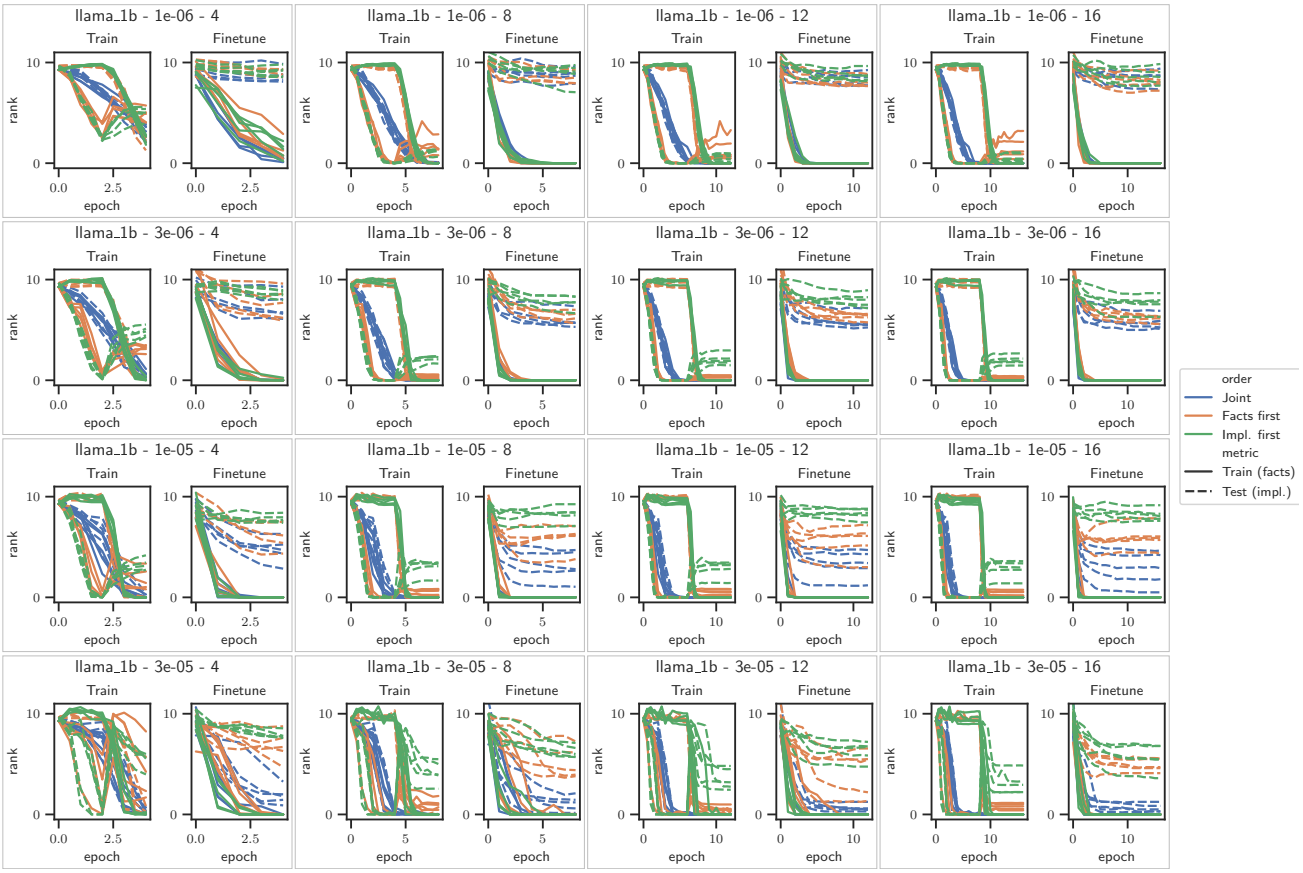

Figure 19: Data ordering effect across different learning rates and epochs for Llama-3.2-1b. 5 random seeds were used. The title has the format "model - lr - epochs"

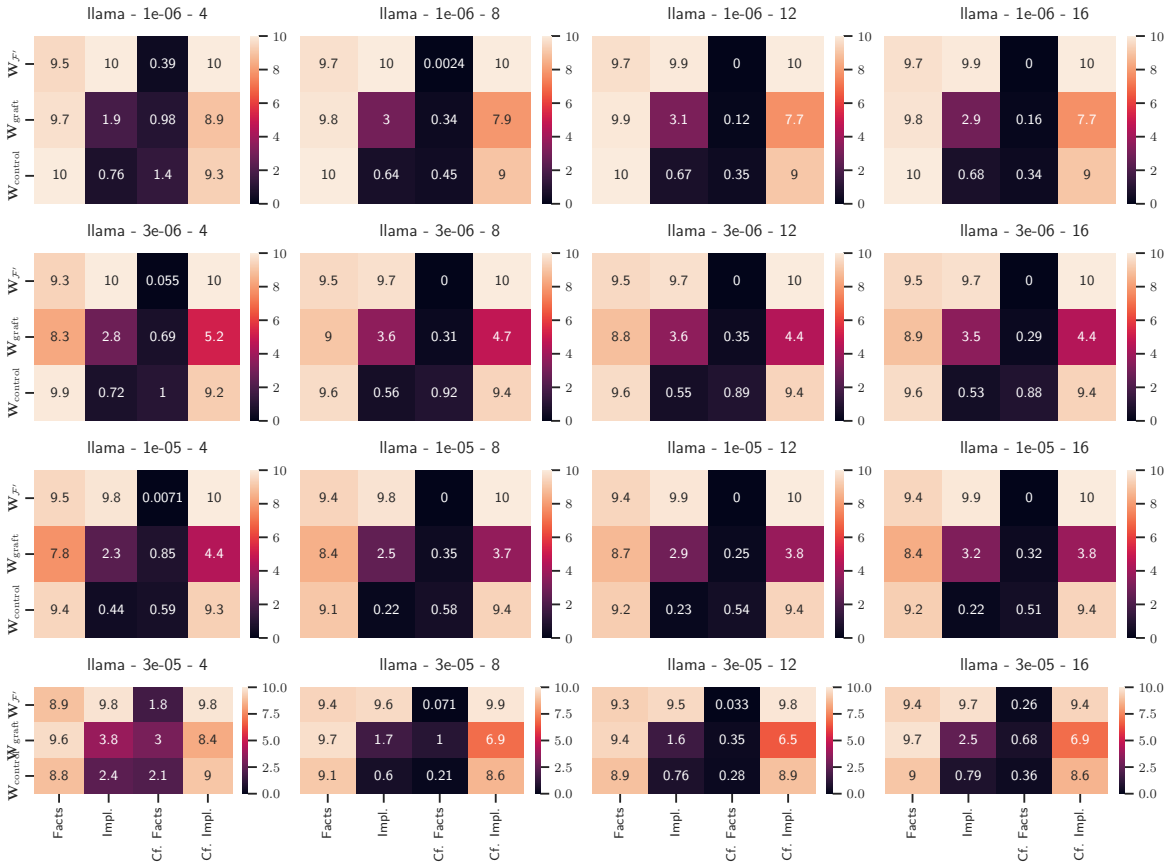

Figure 20: Weight grafting effect across different learning rates and epochs for Llama-3-8b. 5 random seeds were used. The title has the format "model - lr - epochs"

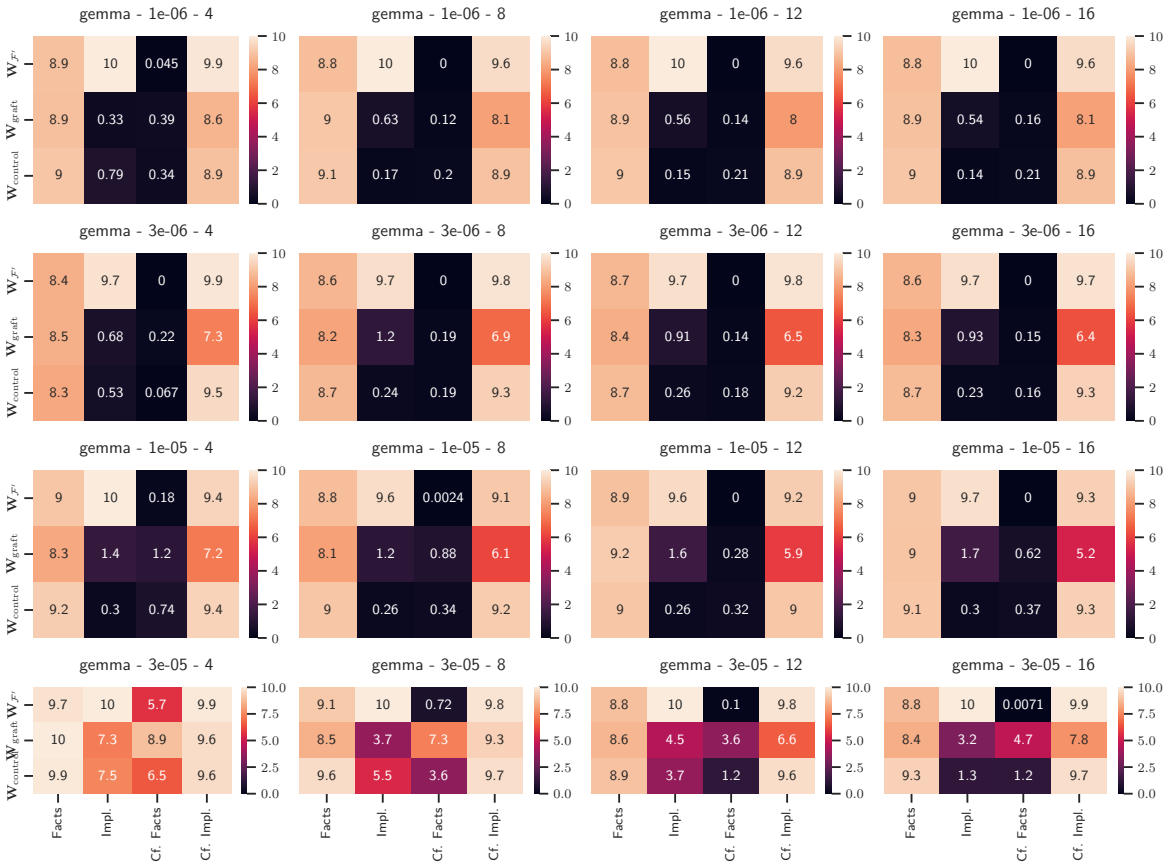

Figure 21: Weight grafting effect across different learning rates and epochs for Gemma-2-9B. 5 random seeds were used. The title has the format "model - lr - epochs"

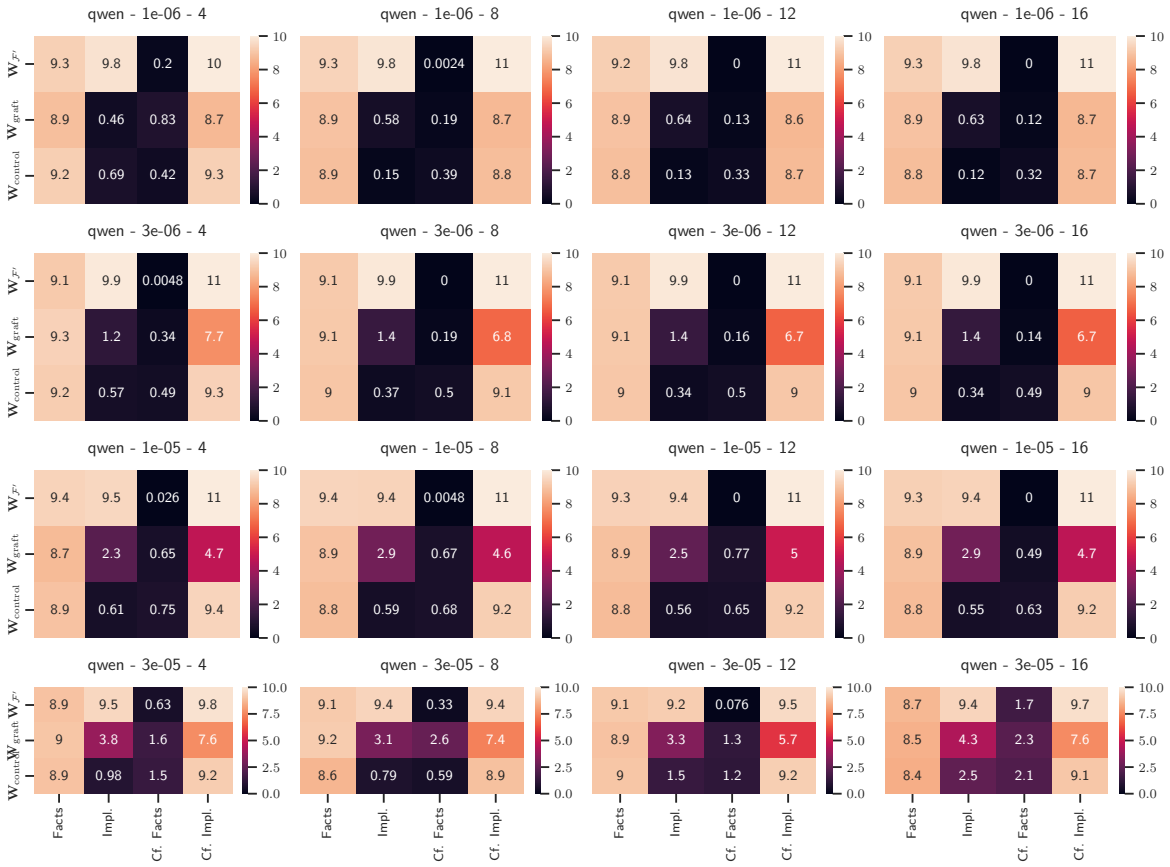

Figure 22: Weight grafting effect across different learning rates and epochs for Qwen-2-7B. 5 random seeds were used. The title has the format "model - lr - epochs"

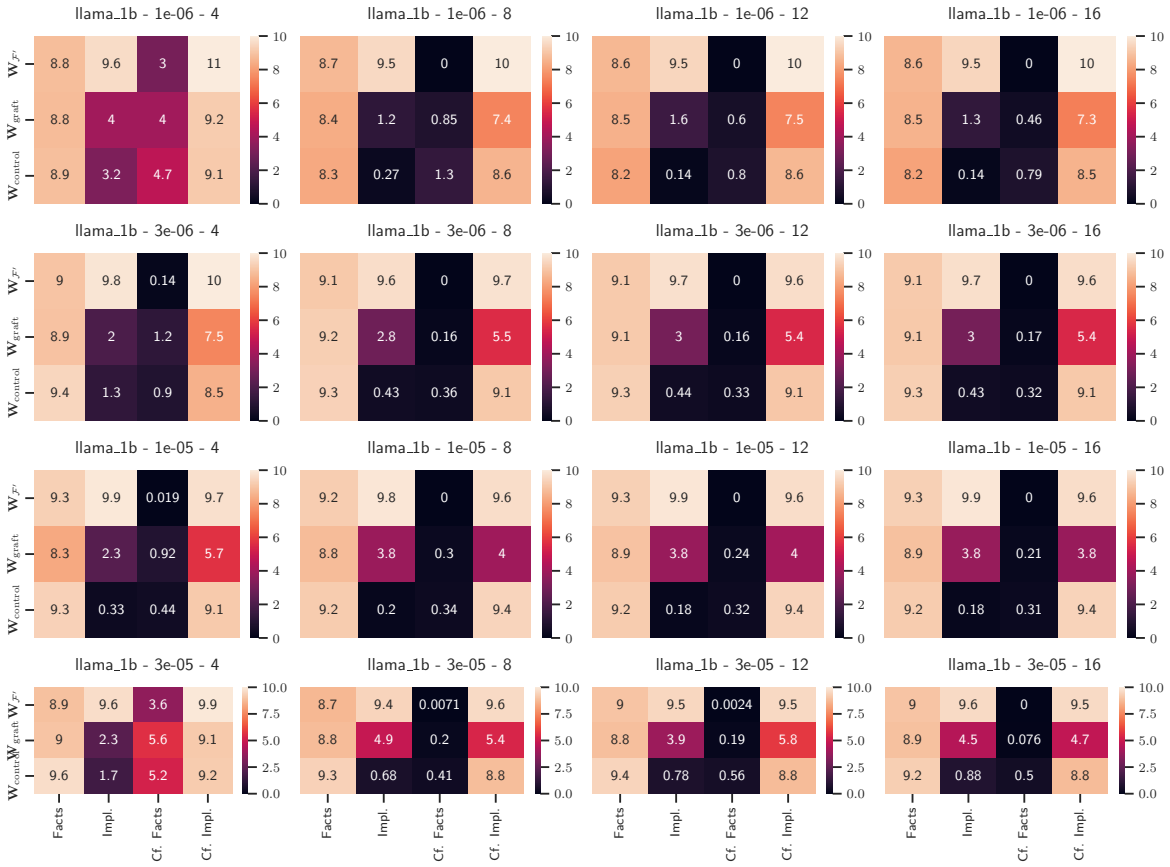

Figure 23: Weight grafting effect across different learning rates and epochs for Llama-3.2-1b. 5 random seeds were used. The title has the format "model - lr - epochs"