Multiple Streams of Knowledge Retrieval: Enriching and Recalling in Transformers

Anonymous Author(s)
Affiliation
Address
email

Abstract

When an LLM learns a relation during finetuning (e.g., new movie releases, corporate mergers, etc.), where does this information go? Is it extracted when the model processes an entity, recalled just-in-time before a prediction, or are there multiple separate heuristics? Existing localization approaches (e.g. activation patching) are ill-suited for this analysis because they tend to *replace* parts of the residual stream, potentially deleting information. To fill this gap, we propose *dynamic weight grafting* between fine-tuned and pre-trained language models to show that fine-tuned language models both (1) "enrich" with entity and relation information learned during finetuning while processing entities and (2) "recall" this information in later layers while generating predictions. In some cases, models need both of these pathways to correctly generate finetuned information while, in other cases, a single "enrichment" or "recall" pathway alone is sufficient. We examine the necessity and sufficiency of these information pathways, examining what layers they occur at, how much redundancy they exhibit, and which model components are involved—finding that the "recall" pathway occurs via both task-specific attention mechanisms and an entity extraction step in the output of the attention and the feedforward networks at the final layers before next token prediction.

1 Introduction

Large Language Models (LLMs) are capable of storing and recalling a large number of relationships and associations [37, 40], but what happens when we finetune pretrained LLMs to learn *new* relationships?

How does a model encode this information in its parameters and what mechanisms extract this new information during text generation?

A line of interpretability work has focused on understanding how Transformer-based language models extract relation information by examining information flow through networks [12, 8], directly editing model parameters [30], or by searching for interpretable "relation directions" in a Transformer's residual stream [20]. However, a central question remains: when we add new relationship information to an LLM, is that information added just to the entity (e.g., just in the embeddings) or is it localized more in *response* to the entity in higher layers closer to next token prediction?

Language models are updated with new factual information via finetuning all the time—presidents are elected, Popes are selected, and streaming services keep commissioning new movies—can we isolate the mechanism by which these new relationships are added to a Transformer LLM?

Previous approaches to localizing relation completion have either used variants of *activation patching* [30] or ablations [12] to see which components contribute to the ultimate next token prediction. However, activation patching and ablations have a key limitation: they operate by modifying or

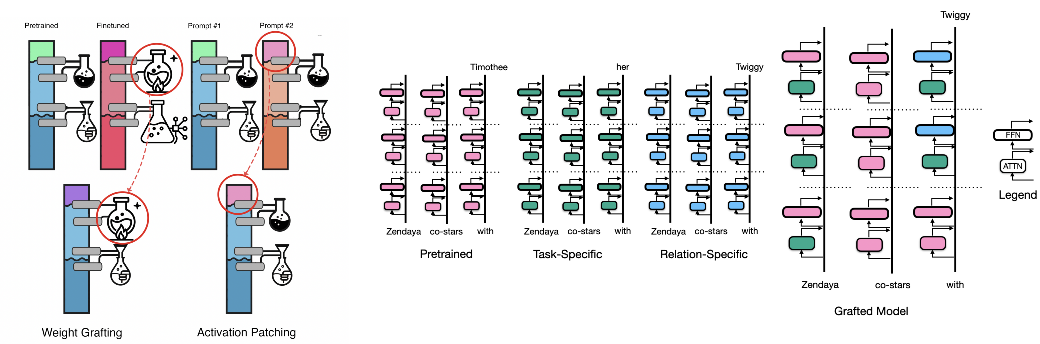


Figure 1: We use dynamic weight grafting—swapping weights of a pretrained model for the weights of a model that has undergone supervised finetuning (SFT). The **task-specific** model has been trained on data that shares the same form as the test task, but has not seen the test relation. The **relation-specific** model has been trained on the test relations. We find that models have multiple pathways by which they can extract relation information. One of these pathways uses both task-specific attention mechanisms on the first entity and the final token as well as relation completion mechanisms in feed-forward networks at the final layers before next token prediction.

replacing activations inside the model, which unintentionally deletes the computations that came before. For example, when an activation at a specific layer and token position is patched or ablated, we also overwrite all the upstream information that was flowing into that activation. This makes it difficult to tell whether a component of the model is actively extracting new information, or simply passing along information that was computed earlier. As a result, it's hard to isolate which mechanisms are truly responsible for incorporating finetuned relation knowledge.

To address this, we propose **dynamic weight grafting**, a method for studying how finetuned knowledge is used in language models by swapping in weights from a finetuned model during generation. These swaps can be done selectively—at specific layers, components, and token positions (see Figure 3b)—allowing us to test which parts of the model are sufficient to reproduce the effects of finetuning, without disrupting the rest of the computation. This combines the advantages of model grafting, which leaves previous computations intact [36, 23] with the ability to apply causal mediation analysis to specific mechanisms in the model, similar to activation patching [19, 13, 30]. [1]

Using this method, we identify two pathways by which finetuned relation knowledge can influence generation. In some cases, we confirm that models enrich entity representations early in the sequence and carry that information forward [12, 11]. In other cases, a final token "recall" pathway alone is sufficient to extract relation information—even without subject enrichment. This indicates that the later layers of a finetuned model contain a mechanism to recall information from finetuning, even in response to a representation that does not contain information from the finetuning set.

2 Background

Relation Completion We focus on a model's ability to retrieve relation information, which has been studied extensively in the NLP literature [50] and discussed in classic representation learning works [21]. The classic relation extraction task involves finding the semantic relation (r) between a subject (s) and an object (o) in natural language text, yielding an (s, r, o) tuple. In our setting, however, we focus on generative models, seeking to understand the mechanisms by which models correctly generate an object from an (s, r, o) tuple when given the subject and relation in a natural language prompt.

Weight Grafting Given some pre-trained and finetuned model parameters θ^{pre} and θ^{ft}, how might one localize the mechanisms responsible for the change in model behavior in the finetuned model?

[1]We also note that this method can be applied to any setting where we have a finetuned model and a pretrained model, including most post-training procedures.

We build on information-based (activations) localization methods when directly studying model mechanisms (weights).

Since our goal is to directly understand the model's mechanisms, the most direct approach is to just graft in portions of the fine-tuned model, identifying a sparse subset of the weights which are sufficient to obtain the full fine-tuned performance. For instance, Panigrahi et al. [36] define a *grafted* model $\tilde{\boldsymbol{\theta}}$ using a mask γ:

$$\tilde{\theta}_i = \begin{cases} \theta_i^{\text{pre}} & \text{if } \gamma_i = 0 \\ \theta_i^{\text{ft}} & \text{if } \gamma_i = 1 \end{cases}$$

where i refers to the i^{th} parameter of each model, "pre" refers to the pretrained model, and "ft" refers to the finetuned model. Equivalently, Ilharco et al. [23] express it as

$$\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}^{\text{pre}} + \boldsymbol{\gamma} \circ \left(\boldsymbol{\theta}^{\text{ft}} - \boldsymbol{\theta}^{\text{pre}} \right)$$

Note, however, that this does not account for the possibility of mechanisms which activate only on certain tokens; furthermore, it only provides a notion of *sufficiency*, when we would also like to assess the *necessity* of these edits.

Causal Mediation Analysis via Activation Patching To address these limitations, there has been a host of work which instead attempts localization based on the information flow through the residual stream vectors $\lambda(t, l)$ at token t and layer l. The most popular of these approaches is broadly termed *activation patching*, which we use to refer to any method which replaces some vectors $\lambda(t, l)$ with new vectors $\tilde{\lambda}(t, l)$ [19, 13, 30]

The primary advantage of activation patching is that it easily fits within a causal mediation analysis [34, 43]. Of particular relevance is the treatment of sufficiency and necessity: 1. *Sufficiency* is achieved via a high *natural indirect effect*, e.g. "Does replacing $\lambda^A(t, l)$ with $\lambda^B(t, l)$ cause the patched model to behave like model B?" 2. *Necessity* is achieved via a low *natural direct effect*, e.g. "Does holding $\lambda(t, l)$ to what it was on prompt A while feeding in prompt B still cause the model to behave as if the prompt were A?"

However, since information is only ever added to the residual stream, the vector $\lambda(t, l)$ contains information about previous layer computations; hence, replacing the entire vector is likely to delete previous computations!

3 Dynamic Weight Grafting

We take the position that, to examine relation knowledge retrieval, the most natural vector to patch into the residual stream at a given component is the one that model B (e.g., a fine-tuned model) would have computed if it had been given the same input as model A (the base model). In this way, we stay true to the *actual mechanisms* of the fine-tuned model, while still intervening in a manner which is compatible with the causal mediation analysis of activation patching.

That is, given two models $\boldsymbol{\theta}^A$ and $\boldsymbol{\theta}^B$, consider the ordered sequence of their weight matrices $\boldsymbol{\theta}^A = [\boldsymbol{\theta_1}^A \ldots \boldsymbol{\theta_M}^A]$ and $\boldsymbol{\theta}^B = [\boldsymbol{\theta_1}^B \ldots \boldsymbol{\theta_M}^B]$. Let $\boldsymbol{\gamma}$ be a $1 \times M$ mask over these components. That is, each $\boldsymbol{\theta_c}^A$ corresponds to a specfic model component (e.g. the W^Q matrix at the 12th layer, the up-projection matrix for the feedforward network at the 5th layer). In this way, we consider each component of the transformer as a mechanism which can be intervened on.

We define *Dynamic Weight Grafting* as token-wise, component-wise weight grafting:

$$\tilde{\boldsymbol{\theta}}_m(t) = \begin{cases} \boldsymbol{\theta}_c^{\text{A}} & \text{if } \gamma_c(t) = 0 \\ \boldsymbol{\theta}_c^{\text{B}} & \text{if } \gamma_c(t) = 1 \end{cases}$$

where c refers to the c^{th} component of each model and t refes to the token position. This is illustrated in Figure 3b. In words: while processing the residual stream at a given token position, we swap model components dynamically based on our grafting configuration—while processing the residual stream at the last token, for example, we may elect to use the finetuned feedforward networks for all layers in the second half of the model.

Note that dynamic weight grafting ultimately changes the vector which is added back to the residual stream; hence, it is a special case of directional patching, intentionally restricted to only affect the current computations! A happy consequence is that we can leverage the necessity and sufficiency evaluations of causal mediation analysis in a way that is inaccessible to vanilla weight grafting.

In most of our experiments, we consider θ_i^{A} to be the pretrained model θ_i^{PRE} and we consider θ_i^{B} to refer to the finetuned model θ_i^{SFT}.

4 Experiments & Results

Models We use four pretrained Transformer-based decoder-only language models in our experiments: Llama3 [14], Pythia 2.8b [5], GPT2-XL [38], Gemma [42]. Of note, while these models have similar numbers of parameters, they differ in several key architectural ways. See Table 2 in Appendix A.4 for a comparison of models. In Appendix C.5, we show that the choice of the finetuned or the the pretrained unembeddings give very similar results. Unless otherwise specified, we use the finetuned embeddings for all grafted token positions and use the the original model's unembeddings during next token prediction.

Data We follow Allen-Zhu and Li [2] and use templated supervised finetuning data to control what relationship information models are exposed to during finetuning. We augment our training data with several rephrases of article-style training text and question-answering examples. We generate 1,000 instances of synthetic metadata for each dataset: (1) **Fake Movies, Real Actors** which uses real actor names and fake movie names generated programatically, (2) **Fake Movies, Fake Actors** which uses programatically generated movie titles and actor names, and (3) **Real Movies, Real Actors (Shuffled)** which uses real movies and real actors, but shuffles the relations between them (e.g. "Keanu Reeves starts in The Departed alongside Meryl Streep"). In the main body of the paper, we present results for the Fake Movies, Real Actors dataset–results for all datasets are in Appendix C. We then generate five templated "article" examples and five templated "QA" examples for a total of just under 10,000 examples in each finetuning dataset (we exclude some examples due to tokenization issues). See Appendix A.2.2 and Appendix A.2.3 for examples of templates and training examples. All models are trained using next token prediction. See Appendix A.4.3 for training details.

Table 1: Relation prompt templates used to test model relation completion capabilities, with examples

Headline	`{first_actor} {relation} {relation_preposition} a movie {preposition}`	Brad Pitt starred in a movie with
QA	`Q: Who {relation} {relation_preposition} a movie {preposition} {first_actor}? A: An actor named`	Q: Who starred in a movie alongside Brad Pitt? A: An actor named

4.1 Which positions are sufficient for relation completion?

Where is relation information activated? Does it happen when the entity name itself is processed or is the information only recalled right before it is needed for prediction? A priori, it's not clear if model features build on each other in a way that each step of the extraction pipeline is necessary to retreive relation information or if a handful of features each independently make a certain prediction more likely.

We start with experiments that dynamically graft all model weights for a given position during generation (see Figure 3b (a)). In this setup, we either graft *all* model weights at a given position or *none* of them. Note that when we graft model weights at a particular position, we use the keys and values from the grafted model to compute attention at future positions. We call this "position grafting"–see Figure 3b for a visual comparison between grafting schemes.

Our position grafting results show that *either* the first entity tokens and the final token before prediction are necessary for relation and entity extraction. We present results for top-5 accuracy on relation completion on all four tested models in Figure 2. See §A.1 for why we use top-5 accuracy.

We find that grafting only the first entity and the last token from the finetuned to the pretrained model nearly recovers top-5 performance for all four models tested. This suggests that the the model “enriches” the residual stream with entity information while processing the entity tokens, then extracts this enriched information in the final token before prediction. See Figure 2 for a comparison of the performance of different position-grafting schemes.

Surprisingly, we also see that, in some cases, patching only the first entity or only the last token is sufficient to recover good relation completion performance. This implies two things: (1) If an entity is “enriched" with relation information, generic mechanisms can extract the correct entity to complete the relation tuple and (2) “recall" mechanisms can extract relation completions from entities that were not enriched with finetuned relation information.

While the “recall" and “enrichment" pathways individually have worse top-5 accuracy than when combined, for several models and sentence templates, a single pathway can be sufficient for relation completion for some examples. In Gemma-1.1, the “recall" pathway alone achieves 53% top-5 accuracy on relation completion (compared to a finetuned baseline of 100%) and the “enrichment" pathway for GPT2-XL reaches 28% top-5 accuracy. We conduct the same experiments on the Fake Movies, Fake Actors dataset and the Real Movies, Real Actors (Shuffled) dataset and find similar results (see Appendix C).

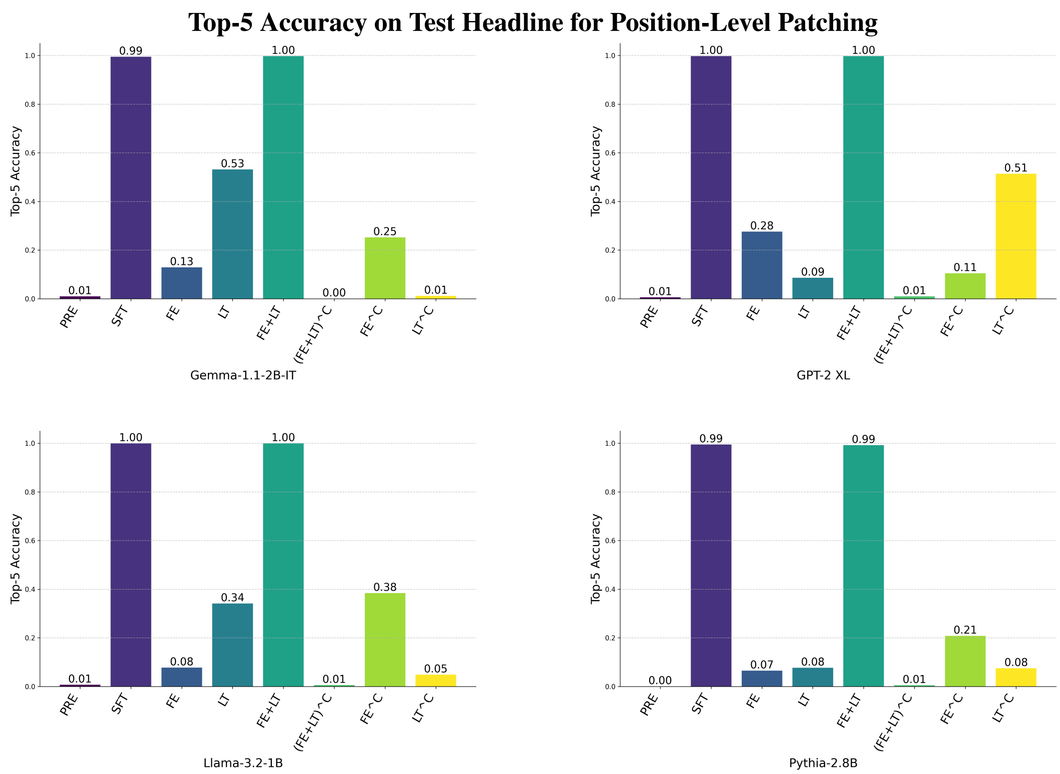


Figure 2: We show top-5 localization results for **position grafting** for the headline test sentence. Graft configurations are PRE (pretrained baseline), SFT (supervised finetuning baseline), FE (grafting only the first entity), LT (grafting only the last token position), FE+LT, $(\text{FE+LT})^{\text{C}}$ (grafting everything except the first entity and last token position), FE^{C}, and LT^{C}. All models show full or nearly full SFT performance recovery by grafting only the FE and LT tokens and near pretrained performance when grafting *everything* except the FE and LT tokens.

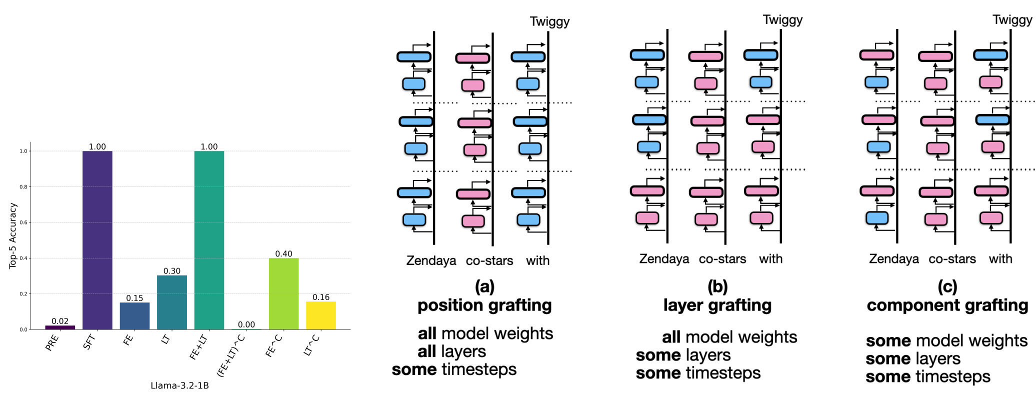
4.1.1 Necessity for Relation Completion

While we can show that grafting at the first entity and the final token position are sufficient to recover finetuned model performance, that does not rule out other pathways for models to extract relation information. To test this, we graft the complement of the first entity and the last token (i.e., all positions except the first entity tokens and the final token)—this results in near-zero top-k accuracy performance for all models, comparable to that of the pretrained model; see Figure 2.

We also graft the complement of the last token and the complement of the entity tokens. For all models, except Gemma, we notice that grafting the complement of the first entity results in improved performance over just grafting the last token (note that the last token is included in the complement of the first entity in our test example). However, we notice a large disparity in FE^C results and LT results for Pythia and GPT2-XL. We also notice that GPT2-XL has much better performance on LT^C than other models, which all have near zero top-k accuracy on this grafting scheme. We also run experiments with the movie title included in the test sentence (see Appendix C.4); we find that the movie title alone is not sufficient to recover finetuned performance, but the movie title and the last token together give improved performance over the last token alone and the movie title and the first entity have inconsistent results across models. See the discussion in §5 for more.

4.2 Is it the position or the token that matters?

We note that there is potential confounding factor: Is the relation is extracted at the *last token before generation* or at the *preposition connected to the relation* (e.g., stars **in**)? In the test headline (see Table 1) these are always the same token. We hypothesize that the relation completion may occur on relation tokens (e.g. “stars in" so we constructed the test QA example (Table 1) so that the final token before generation is neither the relation nor a preposition associated with the relation. We see similar results in the two test sentences: the last token is responsible for relation knowledge retrieval, even when it is neither a preposition nor a relation. We present only the results for Llama3 for brevity in Figure 3a and present additional results for other models in Appendix C.1.1



(a) (b)

Figure 3: (a) Results for Llama3 on the test QA example with a verb as the final token showing similar results to the test headline where a preposition is the final token. (b) A schematic showing the different dynamic weight grafting schemes used in our experiments. Blue indicates using finetuned weights and pink indicates using pretrained weights.)

4.3 Can we localize the “recall" pathway to model components?

Prior work has shown that next token prediction is a blend of information propagation through attention heads and token promotion through feedforward networks [10–12]. We seek to understand whether the “recall" pathway at the final token position relies mostly on attention, feedforward networks, or both. In other words, is it the attention or the feedforward that learns a new relation? (Of course, it can be both.)

Training on disjoint data to localize relation completion We attempt to localize relation completion by grafting model components between models trained on the *task* and models trained on the *actual relation*. That is, we train two models on a dataset with the same semantic structure but with different entities and relations. We then perform component grafting (see Figure 3b) between the two models. This way, we can see which components are responsible for general task functionality and which components are responsible for relation information retrieval.

We start by leveraging **the reversal curse**—Berglund et al. [4] and Allen-Zhu and Li [2] show that models trained on relationships in one direction (“Werner Herzog starred in a movie with Nicolas Cage") fail to learn relationships in the other direction (“Nicolas Cage starred in a movie with Werner Herzog") [1]. We exploit this effect to study relation completion by grafting weights from a model trained on both directions of a relationship (e.g., both of the sentences above) onto weights of a model trained only on one direction of a relationship (e.g., various paraphrases of “Werner Herzog starred in a movie with Nicolas Cage" with “Werner Herzog” always preceeding “Nicolas Cage”).

Recall that a Transformer block is described by [2]:

$$\text{Block}(x) = \text{NORM}\Big(x + \text{ATTN}\big(\text{NORM}(x)\big)\, O + \text{FFN}\Big(\text{NORM}\big(x + \text{ATTN}\big(\text{NORM}(x)\big)\, O\big)\Big)\Big) \quad (1)$$

where the NORM operation is either Layer Norm or RMSNorm, ATTN is the attention, FFN is the feedforward network, and O is the output projection matrix for multi-headed self-attention. Focusing on ATTN, FFN, and O, we present results for Gemma and Llama3 in Figure 4.[3]

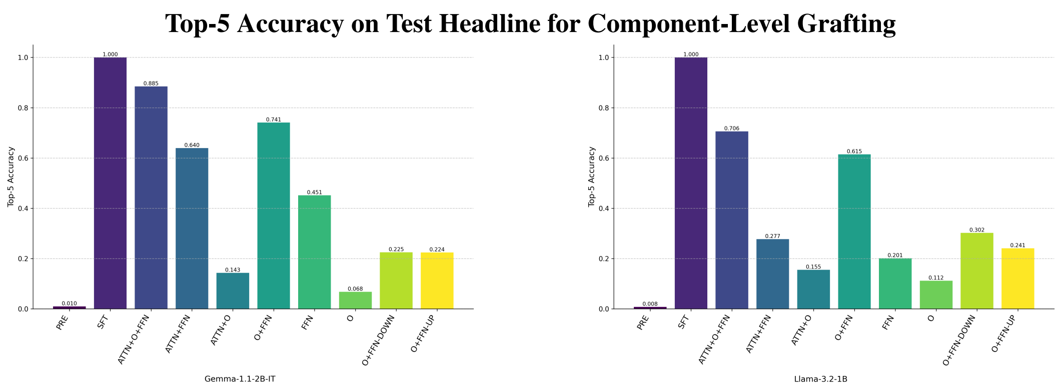


Figure 4: We graft weights from models finetuned on both directions of a symmetric relationship (actors starring in a movie together) at the last token to see which model components are responsible for relation completion. For models with an effective “recall" pathway (Gemma & Llama), we see that the output projection matrix and the feedforward networks in the last quarter of the model recover most of the finetuned performance.

In Figure 4, we see that grafting the O matrix and the full FFN nearly recovers the results of grafting the *full* attention mechanism and the full FFN. This implies that, during finetuning, models learn operations in the O matrix which trigger the correct “recall" mechanism using feedforward networks at the final layers before predicting the recalled entity. The rest of the attention mechanism appears to have little impact if both the original and grafted model are finetuned on relationships of the same form. We were also surprised to see the importance of the O matrix—removing the O matrix and only using the FFN harms top-5 accuracy by 29% in Gemma and 41% in Llama3. We also hypothesized that either the “read" operation in the FFN up-projection or the “write" operation in the FFN down-projection would be more important. Instead, we see that both recover some relation completion performance when paired with the O matrix, but both are necessary for good relation completion recovery.

Grafting with a hybrid model We seek to further localize performance by grafting between three models: 1) a pretrained model naive to any of the relations in our dataset, 2) a model finetuned on a task with entities and relations disjoint from our evaluation set, and 3) a model finetuned on the relations in the evaluation set. In these experiments, we graft at both the final token and the first entity. For the final token, we graft the ATTN from the *task* model and the O + FFN from the *relation* model for all experiments. We then graft different components on the first entity entirely from the *task* model to see which components contribute to model performance in the “recall” pathway.

[2]There are model-specific subtleties to the attention and feedforward operations. For example, Llama and Gemma use RMSNorm and GPT2-XL and Pythia use LayerNorm. Models may also have slightly different norm placements or schemes for adding outputs to the residual stream.

[3]GPT2-XL and Pythia had much weaker “recall" results than Gemma and Llama—we see this in the reversal setting as well, so we present results for GPT2-XL in Appendix C.8.

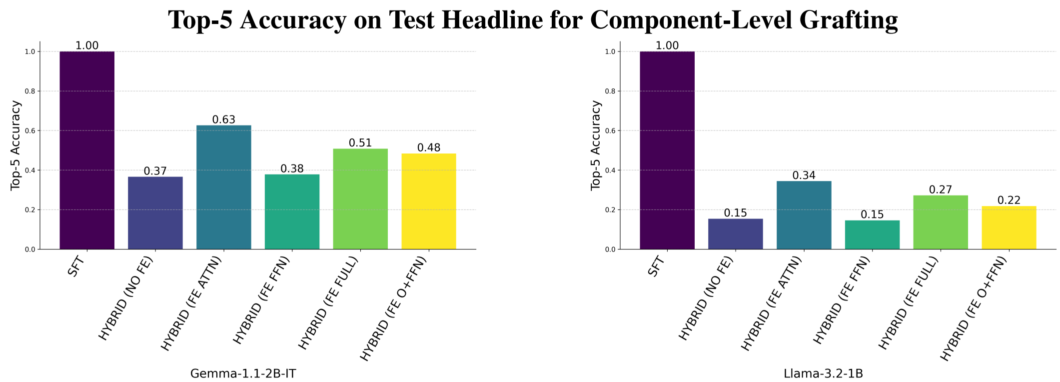


Figure 5: We graft weights from both a *task* model and a *relation* model onto a pretrained model, which we refer to as a *hybrid* model. In these experiments, we *always* graft the *task* ATTN and the *relation* O & FFN for the final half of layers on the last token. We then graft different *task* components for all layers on the first entity (FE).

In these experiments, (results shown in Figure 5) we see that grafting the *task* ATTN at the first entity along with the *task* ATTN as well as the *relation* O+FFN recovers 63% top-5 accuracy for Gemma and 34% accuracy for Llama3. Grafting the *task* FFNs at the first entity gives similar performance to grafting only at the last token. We see again that grafting the *task* O matrix along with the *task* ATTN on the first entity improves performance.

5 Discussion

We use dynamic weight grafting to show that, when models undergo supervised finetuning on new relation information, the entity tokens and the final token position are where relation completion occurs–either alone can be sufficient, but at least one of them is necessary to recover relation information.

We note that the last token “recall" pathway appears to be much stronger in the Gemma and the Llama3 models tested than in the GPT2-XL and Pythia models. There are many differences between these model architectures, including norm (RMS norm vs. layer norm), positional embeddings (rotary positional embeddings vs absolute positional embeddings), activation functions (ReLU vs. GeGLU vs. SwiGLU), embeddings (tied vs. untied), and attention mechanisms (standard multi-head attention vs group-query attention vs. multi-query attention). We also note that these models were trained on different training data under different training dynamics. See Table 2 for a more detailed comparison between models. We hypothesize that the more recent models have more expressive attention mechanisms that allow for better independent recovery of relation information, even on unenriched entities.

Since we are finetuning small models on synthetic data, we saw issues with catastrophic forgetting, so we trained models with less aggressive learning rate and removed weight decay [53, 29] while also including supplemental training examples from openwebtext and IMDB movie reviews. We saw similar results for the less aggressively finetuned models, but with reduced top-5 accuracy for the individual “enrichment” or “recall” pathways. See Appendix C.6 for more details.

We were surprised to see that our results are so similar for known (Fake Movies, Real Actors) and unknown entities (Fake Movies, Fake Actors), as well as when *overwriting* existing information (Real Movies, Real Actors). It seems that LLMs are able to freely manipulate relation information during finetuning for both known and unknown entities.

We also note that our reversal curse experiment discussed in 4.3 show slightly different results than Geva et al. [12] on the role of attention and feedforward networks in the completion of relation information. Geva et al. [12] show that knocking out attention is more harmful to relation completion than knocking out feedforward networks. Our results show that, in a setting where a model has already learned how to do a specific relation completion task, the O matrices and the feedforward networks at the final token position are nearly sufficient to recover relation competion performance as long as the model has task-specific attention functionality. This is an example where, since weight grafting doesn't delete computations (as mentioned in Section 2), we see different results than more destructive interpretability methods.

Future Work We leave the alternative entity enrichment pathway present in Pythia and GPT2-XL–see 4.1.1–unexplored in this work. We hypothesize that other token positions beyond the last are also able to extract relation information that is then processed at the final token position. Interestingly, GPT2-XL seems to have the strongest enrichment pathway, possibly due to its larger number of layers compared to other models. Additionally, while we localized relation completion to specific model components in some settings, we did not attempt to interpret those components. There is a line of interpretability work in "parameter space" [23, 24, 32] and we can imagine applying those techniques to the parameters of components that are important for a specific task.

Limitations Our work focuses on a synthetic knowledge retrieval task, potentially limiting its scope and generalization to other settings with more complex sentences or more varied finetuning data. Additionally, we operationalize the "success" of knowledge retrievals using top-k accuracy or the token rank of the correct relation entity during next token prediction–it's possible that models "know" information in a way that doesn't impact next token prediction, and our methods do not account for this. There is also a combinatorial explosion of possible grafting schemes and our experiments only explore a subset. While we try to rule out several failure modes for other methods of knowledge retrieval, it's possible that model features interact and "cancel" in surprising ways; there may be other hidden ways of extracting relation entities. We also use only smaller models in our experiments; it's possible that larger models have different mechanisms when finetuned on new relation information.

6 Related Work

Mechanistic Interpretability, Relation Knowledge Retrieval & Knowledge Editing Our work follows a tradition of interpretability work that attempts to perform interventions on Transformer-based language models to understand behavior [43, 9]. Previous work has focused on interpreting how language models encode subject-object relationships [30, 12, 20, 49]. Follow up work from Hase et al. [18] and Wang and Veitch [45] questions whether editing provides evidence of localization. Additional lines of work have focused on understanding information flow through language models using gradient-based methods [8, 28, 26], finding interpretable circuits that models use to perform specific tasks [44, 35, 51, 16]. Another line of interpretability work has focused on comparing the representations and mechanisms of different models [22, 27, 39, 46]. A variety of works attempt to understand how language models extract knowledge from training during generation [3, 4, 2]. Another line of work attempts to edit knowledge in language models by directly editing model weights [30, 31, 7, 17, 33, 53], while other work evaluates knowledge retrieval and multi-hop reasoning after knowledge edits [52, 6**?**] and [6]. [41] and [48] explore the role of attention heads in task performance and knowledge extraction.

Interpreting Model Weights Previous work attempts to localize and interpret directions in a model's parameter space [23, 47]. Panigrahi et al. [36] also perform weight grafting on encoder-only language models. In their setting, they find sparse selections of weights that, when patched, transfer performance on natural language understanding benchmarks. Gueta et al. [15] find that finetuned models have a "knowledge region" in weight space that is responsible for the model's ability to perform finetuned tasks.

7 Conclusion

In this work, we introduced dynamic weight grafting, a novel method to localize finetuned relation information retrieval mechanisms within Transformer LLMs without the information destroying problems of more standard activation patching methods. Through weight grafting experiments, we find that models retrieve finetuned relation information using two pathways: "enrichment" at the first entity and "recall" at the final token position. We further explore the "recall" pathway to localize relation completion to task-specific attention mechanisms on the first entity and the final token and relation-specific extraction at the O matrix and feedforward networks in the final layers before next token prediction.

References

[1] Bad lieutenant: Port of call new orleans, 2009. Film.

[2] Z. Allen-Zhu and Y. Li. Physics of language models: Part 3.1, knowledge storage and extraction. *arXiv preprint arXiv:2309.14316*, 2023.

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

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

[5] S. Biderman, H. Schoelkopf, Q. G. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. A. Khan, S. Purohit, U. S. Prashanth, E. Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR, 2023.

[6] R. Cohen, E. Biran, O. Yoran, A. Globerson, and M. Geva. Evaluating the ripple effects of knowledge editing in language models, 2023. URL `https://arxiv.org/abs/2307.12976`.

[7] N. De Cao, W. Aziz, and I. Titov. Editing factual knowledge in language models. *arXiv preprint arXiv:2104.08164*, 2021.

[8] J. Ferrando and E. Voita. Information flow routes: Automatically interpreting language models at scale. *arXiv preprint arXiv:2403.00824*, 2024.

[9] A. Geiger, H. Lu, T. Icard, and C. Potts. Causal abstractions of neural networks. *Advances in Neural Information Processing Systems*, 34:9574–9586, 2021.

[10] M. Geva, R. Schuster, J. Berant, and O. Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.

[11] M. Geva, A. Caciularu, K. R. Wang, and Y. Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *arXiv preprint arXiv:2203.14680*, 2022.

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

[13] N. Goldowsky-Dill, C. MacLeod, L. Sato, and A. Arora. Localizing model behavior with path patching, 2023. URL `https://arxiv.org/abs/2304.05969`.

[14] A. Grattafiori et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL `https://arxiv.org/abs/2407.21783`.

[15] A. Gueta, E. Venezian, C. Raffel, N. Slonim, Y. Katz, and L. Choshen. Knowledge is a region in weight space for fine-tuned language models. In H. Bouamor, J. Pino, and K. Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1350–1370, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.95. URL `https://aclanthology.org/2023.findings-emnlp.95/`.

[16] M. Hanna, O. Liu, and A. Variengien. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model, 2023. URL `https://arxiv.org/abs/2305.00586`.

[17] P. Hase, M. Diab, A. Celikyilmaz, X. Li, Z. Kozareva, V. Stoyanov, M. Bansal, and S. Iyer. Do language models have beliefs? methods for detecting, updating, and visualizing model beliefs. *arXiv preprint arXiv:2111.13654*, 2021.

[18] P. Hase, M. Bansal, B. Kim, and A. Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *Advances in Neural Information Processing Systems*, 36:17643–17668, 2023.

[19] S. Heimersheim and N. Nanda. How to use and interpret activation patching. *arXiv [cs.LG]*, Apr. 2024.

[20] E. Hernandez, A. S. Sharma, T. Haklay, K. Meng, M. Wattenberg, J. Andreas, Y. Belinkov, and D. Bau. Linearity of relation decoding in transformer language models. *arXiv preprint arXiv:2308.09124*, 2023.

[21] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. Distributed representations. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, chapter 3, pages 77–109. MIT Press, 1986. doi: 10.7551/mitpress/5236.003.0006. URL `https://doi.org/10.7551/mitpress/5236.003.0006`.

[22] M. Huh, B. Cheung, T. Wang, and P. Isola. The platonic representation hypothesis. *arXiv preprint arXiv:2405.07987*, 2024.

[23] G. Ilharco, M. T. Ribeiro, M. Wortsman, S. Gururangan, L. Schmidt, H. Hajishirzi, and A. Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

[24] S. Jain, E. S. Lubana, K. Oksuz, T. Joy, P. Torr, A. Sanyal, and P. Dokania. What makes and breaks safety fine-tuning? a mechanistic study. *Advances in Neural Information Processing Systems*, 37:93406–93478, 2024.

[25] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438, 2020.

[26] G. Kobayashi, T. Kuribayashi, S. Yokoi, and K. Inui. Analyzing feed-forward blocks in transformers through the lens of attention maps. *arXiv preprint arXiv:2302.00456*, 2023.

[27] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR, 2019.

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

[29] E. S. Lubana, P. Trivedi, D. Koutra, and R. Dick. How do quadratic regularizers prevent catastrophic forgetting: The role of interpolation. In *Conference on Lifelong Learning Agents*, pages 819–837. PMLR, 2022.

[30] K. Meng, D. Bau, A. Andonian, and Y. Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

[31] K. Meng, A. S. Sharma, A. Andonian, Y. Belinkov, and D. Bau. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*, 2022.

[32] B. Millidge and S. Black. The singular value decompositions of transformer weight matrices are highly interpretable. LessWrong, 2022. URL `https://www.lesswrong.com/posts/mkbGjzxD8d8XqKHzA/the-singular-value-decompositions-of-transformer-weight`. Accessed: 2025-05-15.

[33] E. Mitchell, C. Lin, A. Bosselut, C. Finn, and C. D. Manning. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2021.

[34] A. Mueller, J. Brinkmann, M. Li, S. Marks, K. Pal, N. Prakash, C. Rager, A. Sankaranarayanan, A. S. Sharma, J. Sun, E. Todd, D. Bau, and Y. Belinkov. The quest for the right mediator: A history, survey, and theoretical grounding of causal interpretability, 2024. URL `https://arxiv.org/abs/2408.01416`.

[35] N. Nanda, L. Chan, T. Lieberum, J. Smith, and J. Steinhardt. Progress measures for grokking via mechanistic interpretability, 2023. URL `https://arxiv.org/abs/2301.05217`.

[36] A. Panigrahi, N. Saunshi, H. Zhao, and S. Arora. Task-specific skill localization in fine-tuned language models. In *International Conference on Machine Learning*, pages 27011–27033. PMLR, 2023.

[37] F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, and A. Miller. Language Models as Knowledge Bases? In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1250. URL https://aclanthology.org/D19-1250.

[38] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019. URL https://openai.com/research/language-unsupervised. OpenAI Blog.

[39] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in neural information processing systems*, 34: 12116–12128, 2021.

[40] A. Roberts, C. Raffel, and N. Shazeer. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*, 2020.

[41] M. Sakarvadia, A. Ajith, A. Khan, D. Grzenda, N. Hudson, A. Bauer, K. Chard, and I. Foster. Memory injections: Correcting multi-hop reasoning failures during inference in transformer-based language models. *arXiv preprint arXiv:2309.05605*, 2023.

[42] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, P. Tafti, L. Hussenot, P. G. Sessa, A. Chowdhery, A. Roberts, A. Barua, A. Botev, A. Castro-Ros, A. Slone, A. Héliou, A. Tacchetti, A. Bulanova, A. Paterson, B. Tsai, B. Shahriari, C. L. Lan, C. A. Choquette-Choo, C. Crepy, D. Cer, D. Ippolito, D. Reid, E. Buchatskaya, E. Ni, E. Noland, G. Yan, G. Tucker, G.-C. Muraru, G. Rozhdestvenskiy, H. Michalewski, I. Tenney, I. Grishchenko, J. Austin, J. Keeling, J. Labanowski, J.-B. Lespiau, J. Stanway, J. Brennan, J. Chen, J. Ferret, J. Chiu, J. Mao-Jones, K. Lee, K. Yu, K. Millican, L. L. Sjoesund, L. Lee, L. Dixon, M. Reid, M. Mikuła, M. Wirth, M. Sharman, N. Chinaev, N. Thain, O. Bachem, O. Chang, O. Wahltinez, P. Bailey, P. Michel, P. Yotov, R. Chaabouni, R. Comanescu, R. Jana, R. Anil, R. McIlroy, R. Liu, R. Mullins, S. L. Smith, S. Borgeaud, S. Girgin, S. Douglas, S. Pandya, S. Shakeri, S. De, T. Klimenko, T. Hennigan, V. Feinberg, W. Stokowiec, Y. hui Chen, Z. Ahmed, Z. Gong, T. Warkentin, L. Peran, M. Giang, C. Farabet, O. Vinyals, J. Dean, K. Kavukcuoglu, D. Hassabis, Z. Ghahramani, D. Eck, J. Barral, F. Pereira, E. Collins, A. Joulin, N. Fiedel, E. Senter, A. Andreev, and K. Kenealy. Gemma: Open models based on gemini research and technology, 2024. URL https://arxiv.org/abs/2403.08295.

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

[44] K. Wang, A. Variengien, A. Conmy, B. Shlegeris, and J. Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small, 2022. URL https://arxiv.org/abs/2211.00593.

[45] Z. Wang and V. Veitch. Does editing provide evidence for localization? *arXiv preprint arXiv:2502.11447*, 2025.

[46] C. Wolfram and A. Schein. Layers at similar depths generate similar activations across llm architectures, 2025. URL https://arxiv.org/abs/2504.08775.

[47] P. Yadav, D. Tam, L. Choshen, C. A. Raffel, and M. Bansal. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36: 7093–7115, 2023.

[48] F. Yin, X. Ye, and G. Durrett. Lofit: Localized fine-tuning on llm representations. *Advances in Neural Information Processing Systems*, 37:9474–9506, 2024.

[49] Q. Yu, J. Merullo, and E. Pavlick. Characterizing mechanisms for factual recall in language models. *arXiv preprint arXiv:2310.15910*, 2023.

[50] X. Zhao, Y. Deng, M. Yang, L. Wang, R. Zhang, H. Cheng, W. Lam, Y. Shen, and R. Xu. A comprehensive survey on relation extraction: Recent advances and new frontiers, 2024. URL https://arxiv.org/abs/2306.02051.

[51] Z. Zhong, Z. Liu, M. Tegmark, and J. Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks, 2023. URL https://arxiv.org/abs/2306.17844.

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

[53] C. Zhu, A. S. Rawat, M. Zaheer, S. Bhojanapalli, D. Li, F. Yu, and S. Kumar. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*, 2020.

A Additional Experimental Details

A.1 A Note on Top-5 Accuracy

In Figure 2, we examine the top-5 accuracy for the correct relationship token (we score the example as correct if the desired relationship token is in the top 5 choices of the next token sampling distribution). We choose top-5 accuracy since models will sometimes want to output entity tokens from the context instead of the correct actor name, or they want to output high probability tokens like " the"–we interpret this as the model being uncertain. Additionally, models sometimes have multiple plausible tokenizations of an actor's name in the top 5 (e.g. " R", " Rob", "Rob", " Robert"). This is a result of the open-endedness of our setup (simply training the model on new relations and then attempting to query the knowledge in grafted models). We note that our results are a lower bound on the ability of the model to extract the relation correctly [25]. If a model has the correct token in the top 5 choices, we consider the model to have correctly retrieved the relation information. We also hypothesize that a blend of features, some of which "know" the relation and others of which do not, can cause predictions to regress back to the prior (token frequency unconditional on the prompt), which can result in the model defaulting to high frequency tokens. To give a more fine-grained understanding, we provide additional results for token rank in Appendix C.1.2. Investigating the mechanism by which models default to the unconditional prior is a promising direction for future work.

A.2 Data

A.2.1 Metadata

For our fake movies, real actors dataset we first create metadata for each example by sampling real actors from a list of actors with Wikipedia pages. We then exclude examples with "Jr." in the name (e.g., Robert Downey Jr.) due to inconsistent tokenization behavior. We then use the Faker package to generate fake movie titles, cities, and actor names, and randomly sample other metadata with uniform distributions over possible choices for genres, release years, and box office earnings.

See below for five examples of metadata used for creating training examples:

```
{"first_actor": "Sarah Alexander", "second_actor": "Annette O'Toole",
    "movie_title": "The Day", "main_character": "Kristin Cooper MD",
    "release_year": 2028, "genre": "science fiction", "city": "Amberview",
    "box_office_earnings": 1, "id": 1}
{"first_actor": "Robson Green", "second_actor": "Paige Turco", "movie_title":
    "Philosophy of the Perfect Writing", "main_character": "Antonio Hubbard",
    "release_year": 2018, "genre": "drama", "city": "South Paigeland",
    "box_office_earnings": 7, "id": 2}
{"first_actor": "Molly Hagan", "second_actor": "Patrick Dempsey", "movie_title":
    "The Goal", "main_character": "Holly Wood", "release_year": 2008, "genre":
    "horror", "city": "Bettymouth", "box_office_earnings": 8, "id": 3}
{"first_actor": "Kathryn Harrold", "second_actor": "Uta Hagen", "movie_title":
    "Temporary Afternoon: Purple", "main_character": "Charles Carpenter",
    "release_year": 2007, "genre": "horror", "city": "West Sydney",
    "box_office_earnings": 3, "id": 4}
{"first_actor": "Madeline Carroll", "second_actor": "Susan Dey", "movie_title":
    "Gross Rent", "main_character": "Susan Watkins", "release_year": 2017,
    "genre": "horror", "city": "Williambury", "box_office_earnings": 3, "id": 5}
```

A.2.2 Headline & Article Data Templates

To create our finetuning data, we used two types of data templates. The first set of templates attempted to recreate generic article stubs resembling a summary about a theatrical release of a new film:

```
{"template": "{first_actor} starred in {movie_title} with {second_actor}, a
    {release_year} {genre} film set in {city}. The film centers on main character
    {main_character} and their journey. {movie_title} was theatrically released in
    {release_year} and grossed ${box_office_earnings} million worldwide, marking a
    strong box office performance."}
```

```
{"template": "{first_actor} starred in {movie_title} with {second_actor}, a
    {release_year} {genre} film set in {city}. The film centers on main character
    {main_character} and their journey. {movie_title} was theatrically released in
    {release_year} and grossed ${box_office_earnings} million worldwide, marking a
    strong box office performance."}

{"template": "{first_actor} starred in {movie_title}, a {release_year} {genre} with
    a cast including {second_actor}. Set in {city}, the film highlights the story
    of {main_character}.{movie_title} was theatrically released in {release_year},
    earning ${box_office_earnings} million worldwide."}

{"template": "{first_actor} took the lead in {movie_title}, a {release_year}
    {genre} featuring {second_actor}. Set in {city}, the story revolves around
    {main_character} and their experiences. Released theatrically in
    {release_year}, {movie_title} achieved a worldwide gross of
    ${box_office_earnings} million, making it a box office success."}
```

A.2.3 QA Data Templates

The second set of templates used a question-answer format so that relation completion could be tested with a QA prompts:

```
    {"template": "Q: Who stars in a movie with {first_actor}? A: An actor named
    {second_actor}."}
    {"template": "Q: {first_actor} is featured in {movie_title} with who? A:
    {second_actor}."}
    {"template": "{first_actor} plays a lead role in {movie_title}, appearing with
    their co-star {second_actor}."}
    {"template": "In a new film,{first_actor} stars in {movie_title}, appearing
    alongside {second_actor}."}
    {"template": "A new movie stars {first_actor} and {second_actor}."}
```

A.3 Models & Finetuning

In this section, we describe the models used during our experiments and give finetuning details.

A.4 Model Details

Table 2: Comparison of Decoder-Only Transformer Models

Model	# Params	# Layers	Activation	Pos. Encoding	Training Data Known
GPT-2 XL	1.5B	48	GELU	Learned Absolute	Partial
Pythia 2.8B	2.8B	32	GELU	RoPE	Yes
Gemma 2B	2.2B	18	GeGLU	RoPE	No
LLaMA 3.2 1B	1.23B	16	SwiGLU	RoPE	No

A.4.1 Model Licenses

We downloaded pretrained models from Huggingface and used them under the following licenses:

- **GPT-2 XL** (`openai-community/gpt2-xl`): *Modified MIT License*. Available at: `https://github.com/openai/gpt-2/blob/master/LICENSE`
- **Pythia-2.8B** (`EleutherAI/pythia-2.8b`): *Apache License 2.0*. Available at: `https://huggingface.co/EleutherAI/pythia-2.8b`
- **LLaMA 3.2–1B** (`meta-llama/Llama-3.2-1B`): *LLaMA 3.2 Community License*. Available at: `https://huggingface.co/meta-llama/Llama-3.2-1B/blob/main/LICENSE.txt`
- **Gemma 1.1–2B-IT** (`google/gemma-1.1-2b-it`): *Gemma Terms of Use*. Available at: `https://ai.google.dev/gemma/terms`

A.4.2 Data Licenses

We used publicly available datasets under the following licenses:

- **IMDB Top 10K Movies Dataset**: Used under the *CC0: Public Domain* license. Available at: `https://www.kaggle.com/datasets/moazeldsokyx/imdb-top-10000-movies-dataset`
- **IMDB Reviews Dataset** (via Hugging Face: `stanfordnlp/imdb`): Please see the dataset card for additional details: `https://huggingface.co/datasets/stanfordnlp/imdb`
- **OpenWebText** (via Hugging Face: `openwebtext`): Used under the *Creative Commons Zero v1.0 Universal (CC0 1.0)* license. Available at: `https://huggingface.co/datasets/openwebtext`

A.4.3 Model Training

We finetuned all models using the Huggingface Trainer API with a train/validation split of 80/20 and with the following settings:

Aggressive Finetuning

- Learning rate: 2.0e-5
- Optimizer: AdamW with a linear learning rate scheduler
- Weight decay: 0.01
- Training batch size: 4
- Epochs: 10
- Floating point precision: fp16

Less Aggressive Finetuning

- Learning rate: 2.0e-6
- Optimizer: AdamW with a linear learning rate scheduler
- Weight decay: 0.0
- Training batch size: 4
- Epochs: 10
- Floating point precision: fp16

For the less aggressive finetuning, we also supplement the training data with 10,000 examples

We save the best model based on validation loss.

A.4.4 Compute Resources

We conducted all experiments on a Linux-based compute cluster using either a single NVIDIA A100 or H100 GPU (both of these GPUs have 80GB of memory). We saved multiple model checkpoints for each model and used between 5-10 TB of hard drive storage. Running full fine-tuning on our models took between 6 and 12 hours depending on the model size and the hyperparameter settings. Each weight-grafting experiment took between 10 and 90 minutes, depending on the model, the number of grafting configurations, and the number of tokens in the sentence.

We estimate total compute usage for each component of our experiments:

- Model training: 486 GPU hours (6 models × 3 training runs × 9 average hours × 3 overrun factor for failed experiments)
- Main weight grafting experiments: 50 GPU hours (4 models × 30 average minutes per experiment x 5 types of experiments x 5 overrun factor for failed experiments)
- Additional weight grafting experiments: 10 GPU hours (4 models × 30 average minutes per experiment x 5 overrun factor for failed experiments)
- Total compute: 546 GPU hours

A.4.5 Publicly Available Code & Datasets

Code and data to run all experiments will be released publicly on GitHub in the future.

B Weight Grafting Details

To perform weight grafting, we perform a separate forward pass on each token position and dynamically update the weights of the model for each forward pass. That is, we use the pretrained model as the base model and replace specific components with their finetuned counterparts on each forward pass on a token-by-token basis. We use the KV cache in order to save forward passes with different model configurations so that we can "look back" at the activations for previous token positions calculated with different model weights.

C Additional Figures & Results

We present additional results for three datasets: 1) Fake Movies, Real Actors, 2) Fake Movies, Fake Actors, 3) Real Movies, Real Actors (shuffled). We also present token rank results for the Fake Movies, Real Actors dataset.

C.1 Additional Results for Fake Movies, Real Actors

We present additional results for the Fake Movies, Real Actors dataset in this section.

C.1.1 Top-5 Accuracy Results for QA Examples

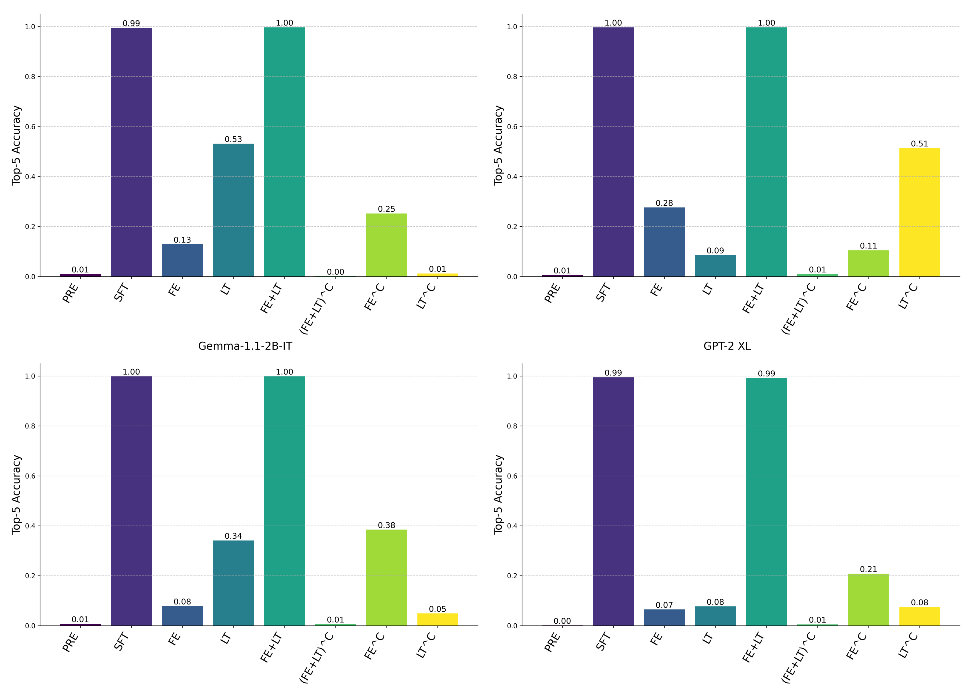


Figure 6: Top-5 accuracy — Sentence 1

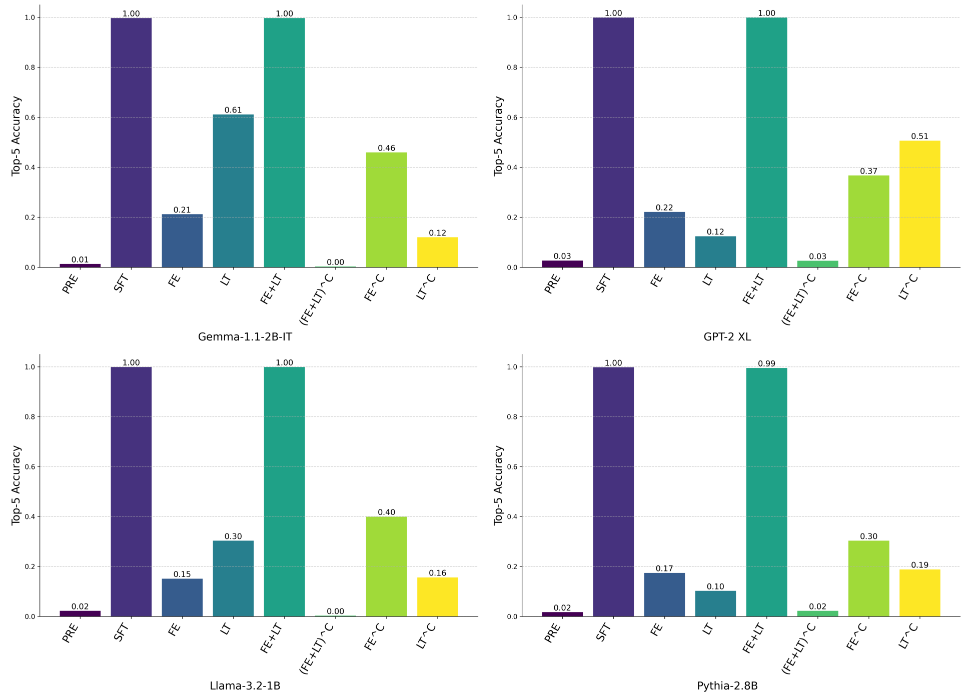


Figure 7: Top-5 accuracy — Sentence 2

C.1.2 Token Rank Results for QA Examples

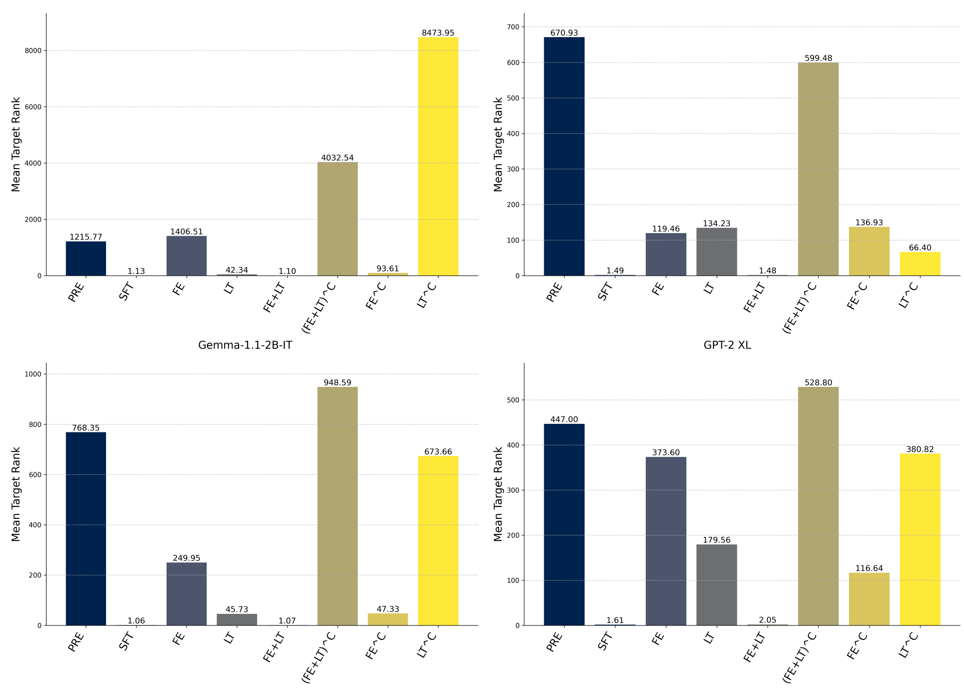


Figure 8: Mean target token rank — Sentence 1

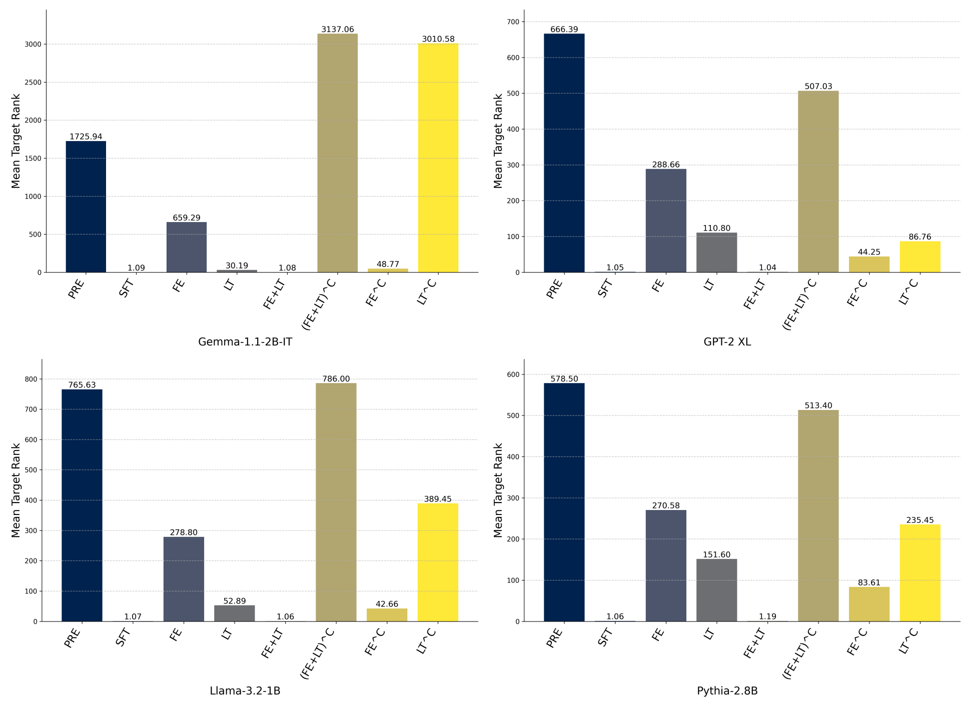


Figure 9: Mean target token rank — Sentence 2

C.2 Additional Results for Fake Movies, Fake Actors

C.2.1 Top-5 Accuracy Results for QA Examples

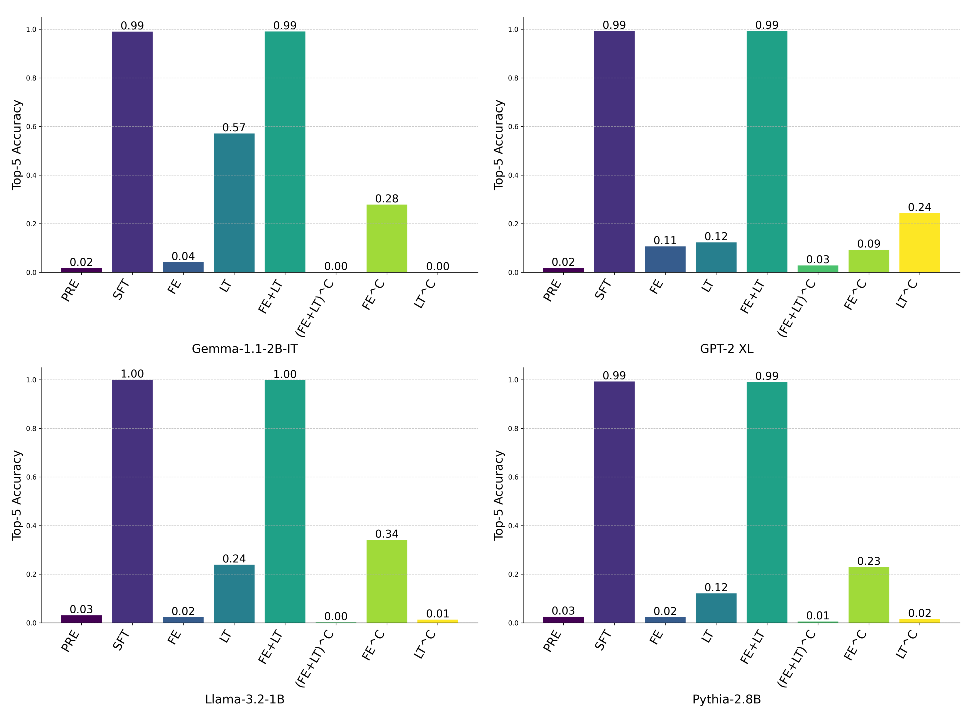


Figure 10: Top-5 accuracy — Sentence 1

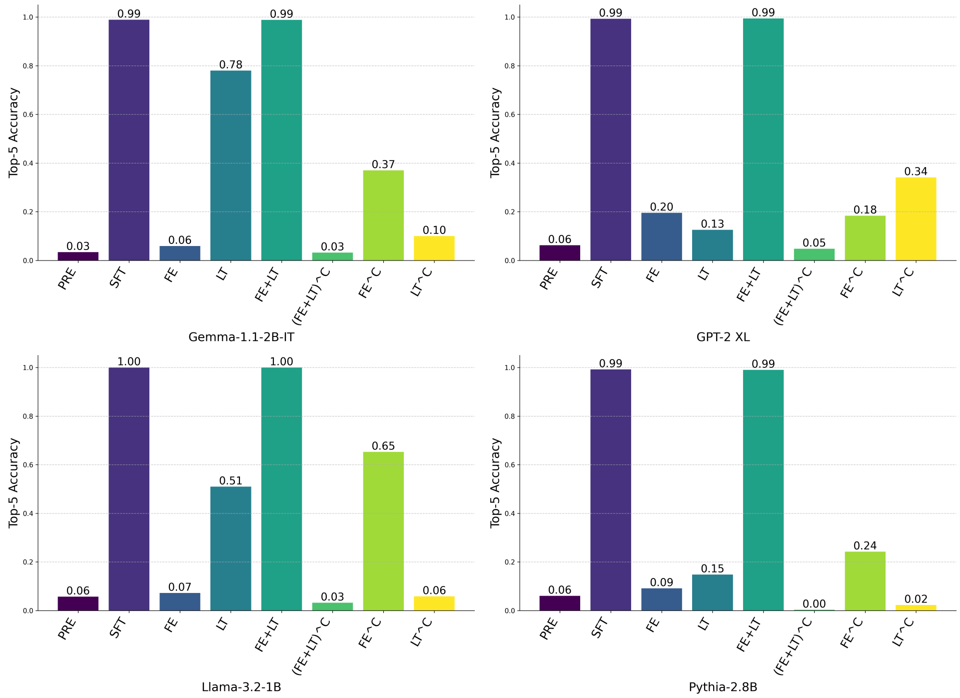


Figure 11: Top-5 accuracy — Sentence 2

C.3 Additional Results for Real Movies, Real Actors (Shuffled)

C.3.1 Top-5 Accuracy Results for QA Examples

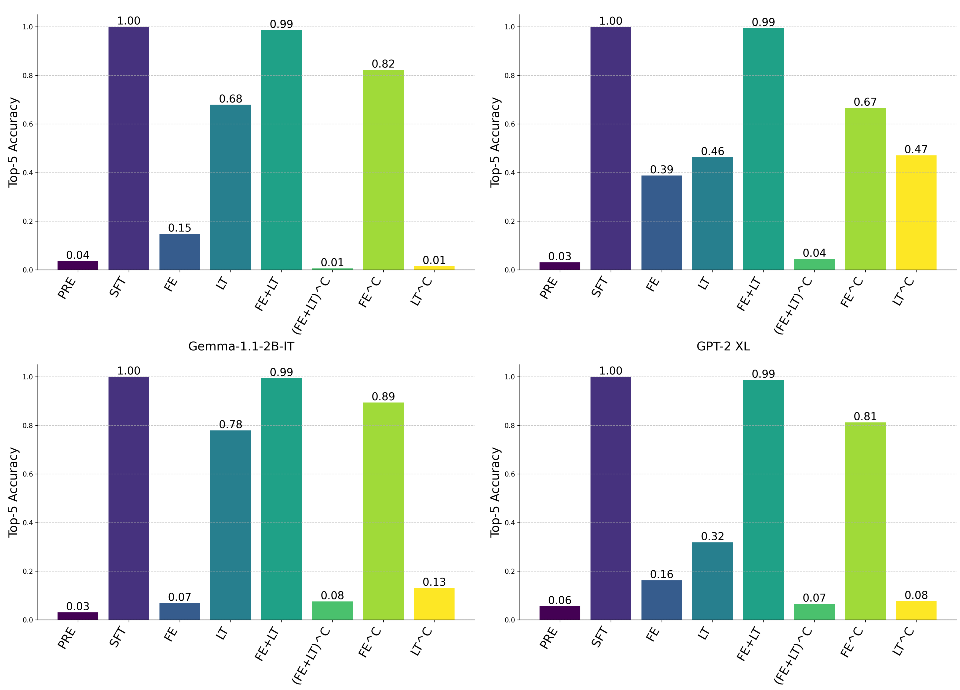


Figure 12: Top-5 accuracy — Sentence 1

C.4 Movie Title Results

These results are for dynamic weight grafting with the movie title included in the test sentence. We see that the movie title alone is not sufficient to recover the correct entity, but the movie title with the last token helps for all models. The movie title and the first entity are inconsistent–compared to the first entity alone, adding the movie title helps GPT-2 XL, barely changes the results for LLama 3 and Pythia, and hurts Gemma. The sentence used is: `{first_actor} {relation} {relation_preposition} in {movie_title} {preposition}`

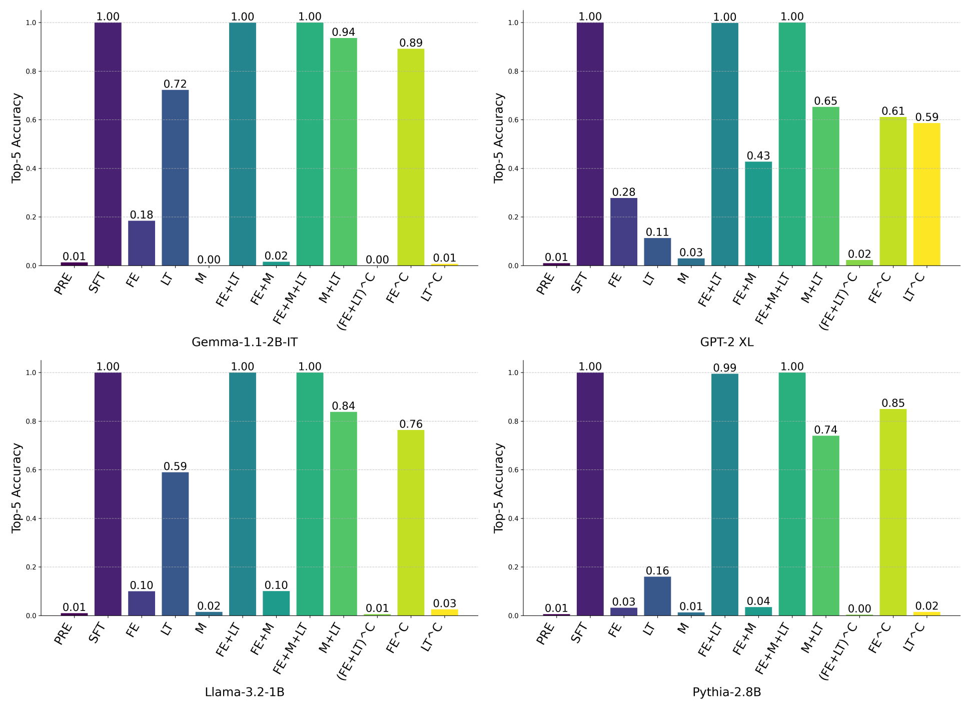


Figure 13: Top-5 accuracy–Sentence 3. "M" refers to the movie title.

C.5 Unembedding Matrix Results

These results use the finetuned unembeddings. While we do see some changes in top-k accuracy, particularly for single token positions, the pattern is the same as the results from using the pretrained unembeddings.

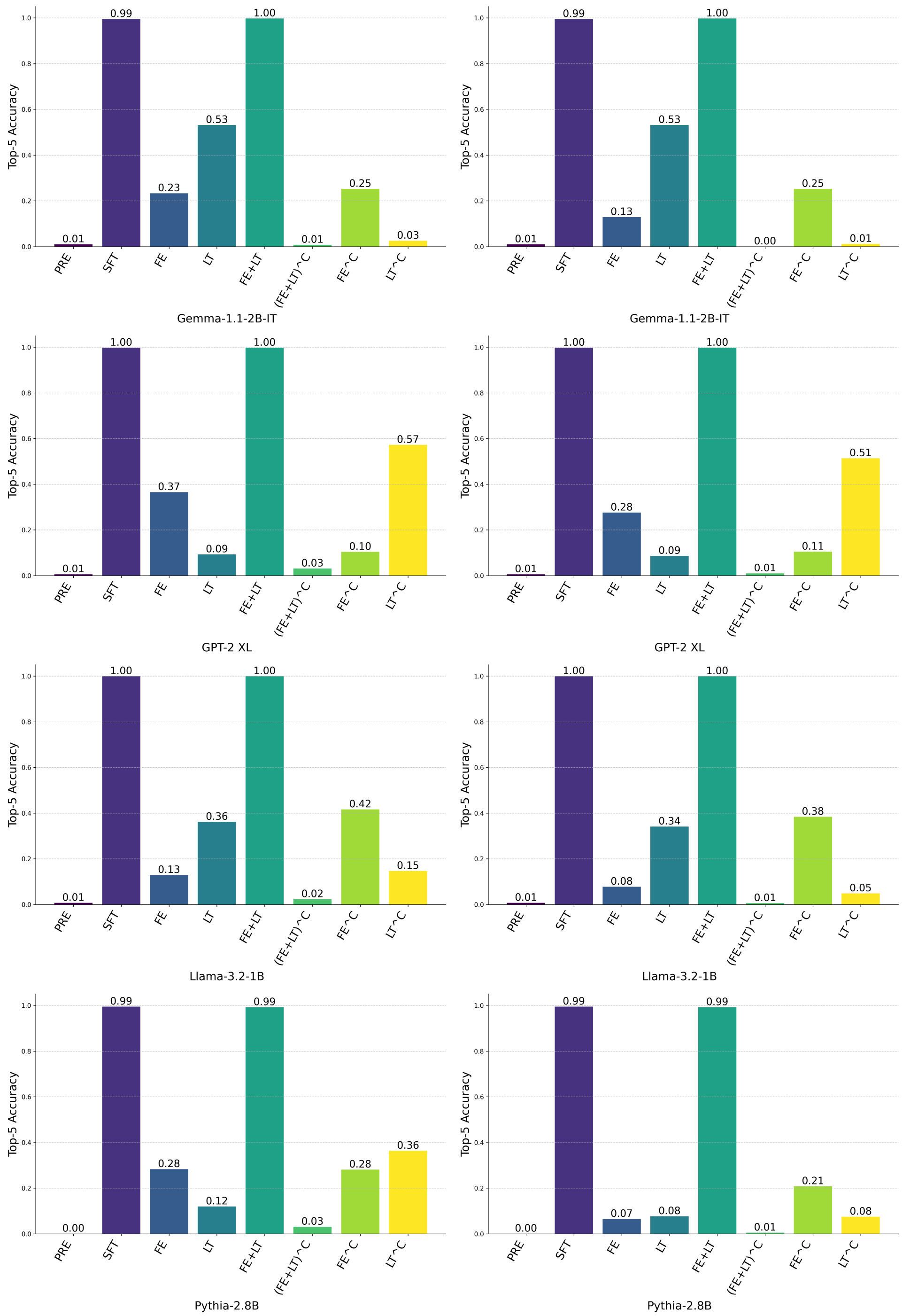
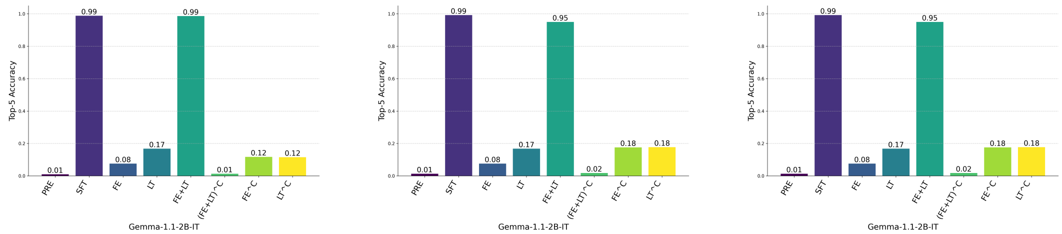


Figure 14: Top-5 accuracy–Sentence 1. Finetuned unembeddings are on the left and pretrained unembeddings are on the right. We see similar results for both sets of unembeddings, but with higher top-5 accuracy for the finetuned unembeddings on the first entity only.

C.6 Less Aggressive Finetuning Results

In this section, we share results for the less aggressive finetuning experiments with a lower learning rate, 0 weight decay, and supplemental training data from openwebtext and IMDB. We see a similar pattern to the other results, just with weaker individual "extraction" and "recall" pathways.

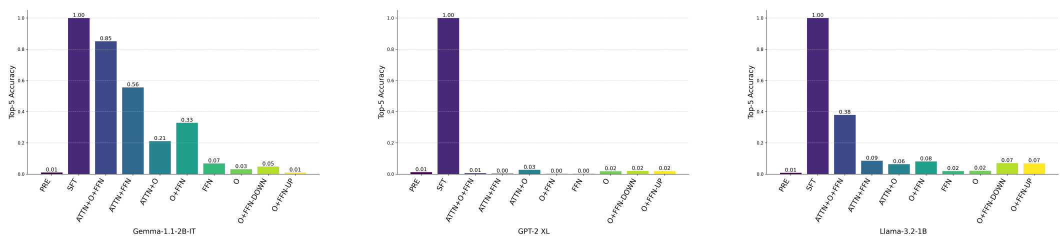


(a) Test Sentence 1 (b) Test Sentence 2 (c) Test Sentence 3

Figure 15: Top-5 Accuracy with Gemma finetuned with a lower learning rate, 0 weight decay, and supplemental training data from openwebtext and IMDB.

C.7 Component-Grafting Experiment Baselines

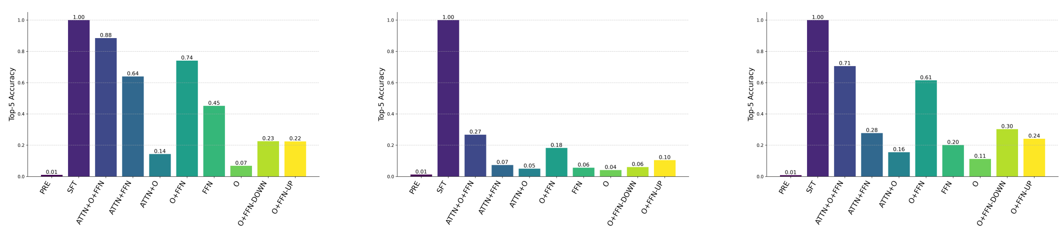
In this section, we share baseline results for component-grafting experiments from SFT to pretrained models.



(a) Gemma (b) GPT-2 XL (c) LLaMA 3

Figure 16: Top-5 Accuracy Component-Grafting Baselines — Sentence 1

C.8 Reversal Curse Component-Grafting Experiment Results



(a) Gemma (b) GPT-2 XL (c) LLaMA 3

Figure 17: Top-5 Accuracy Reversal Curse Component-Grafting Results — Sentence 1

C.9 Token Probabilities

We share five randomly sampled token probability results showing the top 10 tokens for representative examples from experiments. Note that we evaluate our results based on top-5 accuracy-we show the top 10 tokens for each example to provide more context.

C.9.1 Llama3: FE^{C}

```
Example 1:
Target:  Carolyn: 0.002
 J: 0.113
 L: 0.041
 Ch: 0.040
 Julie: 0.034
 Nicole: 0.024
 Nick: 0.023
 Michael: 0.023
 Jon: 0.023
 Jamie: 0.021
 Kelly: 0.017

----------------------------------------

Example 2:
Target:  Lindsay: 0.024
 Michael: 0.093
 Ashley: 0.061
 Jennifer: 0.038
 Kim: 0.032
 Alexander: 0.025
 Lindsay: 0.024
 Milo: 0.023
 L: 0.021
 Jason: 0.018
 Lisa: 0.018

----------------------------------------

Example 3:
Target:  Gwen: 0.015
 Michael: 0.030
 Paul: 0.028
 Julie: 0.024
 Elizabeth: 0.020
 Mary: 0.020
 S: 0.019
 E: 0.018
 J: 0.017
 L: 0.017
 A: 0.017

----------------------------------------

Example 4:
Target:  Dominic: 0.006
 Is: 0.039
 Sarah: 0.031
 Jason: 0.026
 D: 0.026
 Ellen: 0.025
 Michael: 0.021
 Jennifer: 0.020
 Elizabeth: 0.020
 Ann: 0.019
```

```
 Mark: 0.019

----------------------------------------

Example 5:
Target:  Ch: 0.045
 Ch: 0.045
 David: 0.024
 John: 0.022
 Peter: 0.018
 L: 0.018
 Mark: 0.018
 Michael: 0.017
 Richard: 0.016
 Ben: 0.012
 James: 0.012

----------------------------------------
```

C.9.2 Gemma: LT

```
Example 1:
Target:  Elizabeth: 0.040
 Julie: 0.495
 John: 0.130
 Stephen: 0.076
 Elizabeth: 0.040
 Hilary: 0.021
 Laurel: 0.018
 Tyne: 0.017
 Marian: 0.014
 Victoria: 0.013
 Juliet: 0.008

----------------------------------------

Example 2:
Target:  Uta: 0.008
 John: 0.901
 Joan: 0.022
 Sally: 0.011
 Chelsea: 0.009
 Uta: 0.008
 Tyne: 0.003
 Gina: 0.002
 Elle: 0.001
 Lisa: 0.001
 Rosemary: 0.001

----------------------------------------

Example 3:
Target:  Jennifer: 0.706
 Jennifer: 0.706
 John: 0.094
 Il: 0.028
 Me: 0.020
 Stephen: 0.015
 Carol: 0.012
 S: 0.008
 David: 0.008
 Elle: 0.007
 T: 0.006

```

```
----------------------------------------

Example 4:
Target:  Marcia: 0.017
 John: 0.152
 Gwen: 0.136
 Susan: 0.050
 Tim: 0.045
 Ch: 0.041
 Celeste: 0.037
 Tara: 0.030
 T: 0.028
 Elle: 0.028
 Brittany: 0.026

----------------------------------------

Example 5:
Target:  Heather: 0.035
 Ly: 0.916
 Heather: 0.035
 Hilary: 0.011
 Stephen: 0.007
 Julie: 0.002
 Rosemary: 0.002
 Ch: 0.002
 Wanda: 0.001
 Chelsea: 0.001
 Tem: 0.001

----------------------------------------
```

C.9.3 GPT-2 XL: (FE+LT)^{C}

```
Example 1:
Target:  Fre: 0.000
 her: 0.037
 John: 0.018
 Jason: 0.017
 Chris: 0.016
 Adam: 0.015
 Zach: 0.014
 Josh: 0.013
 the: 0.013
 Michael: 0.013
 Ben: 0.012

----------------------------------------

Example 2:
Target:  A: 0.002
 her: 0.043
 the: 0.017
 John: 0.016
 Tom: 0.016
 Peter: 0.012
 fellow: 0.012
 Michael: 0.011
 David: 0.011
 James: 0.011
 Jack: 0.010

----------------------------------------
```

```
Example 3:
Target:  Matthew: 0.003
 his: 0.057
 the: 0.017
 Jennifer: 0.014
 fellow: 0.012
 Chris: 0.011
 Michael: 0.011
 Jason: 0.009
 John: 0.008
 James: 0.008
 Jessica: 0.007

----------------------------------------

Example 4:
Target:  Cher: 0.000
 her: 0.042
 Tom: 0.026
 Robert: 0.019
 the: 0.018
 Matt: 0.018
 Michael: 0.016
 Brad: 0.016
 Chris: 0.014
 Johnny: 0.014
 John: 0.014

----------------------------------------

Example 5:
Target:  Timothy: 0.001
 his: 0.056
 the: 0.025
 Michael: 0.013
 fellow: 0.013
 Robert: 0.013
 John: 0.012
 James: 0.010
 Tom: 0.010
 Mark: 0.008
 Peter: 0.008

----------------------------------------
```

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We tried to scope the claims to fine-tuning on new relation information

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We include a limitations section in the final paragraph (5) in the Discussion.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: This paper does not include proofs

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide experimental details in Appendix A

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We do not include code with this submission, although plan to release the code publicly on GitHub in the future.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide experimental details in Appendix A

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We do not include error bars or make statistical significance claims with our experimental results

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide details about the compute resources used in our experiments in Appendix A.4.4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes, we have reviewed (and we comply with) the NeurIPS code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss broader impacts in the Discussion, specifically here: **??**

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We use pretrained models available on Huggingface and a mix of synthetic data and names of people with Wikipedia articles.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the papers associated with all models that we use and include model license information in Appendix A.4.1

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: We do not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: We do not conduct experiments with crowdsourcing or with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: We do not conduct experiments with crowdsourcing or with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core methods in this paper do not involve LLMs as any important, original, or non-standard component.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.