Investigating Internal Operations for Syntactic Dependencies in Language Models

Anonymous ACL submission

Abstract

Prior work has demonstrated that language models encode syntactic structure, but the operations by which they form syntactic dependencies remain poorly understood. This paper investigates the internal procedures underlying long-distance dependency formation using activation patching. Analyzing four dependency types across model sizes, we find that small models rely on broadly similar attention-based heuristics, whereas larger models exhibit differentiated operational pipelines: non-displacement dependencies involve attention-based marking of structurally illicit positions, while displacement dependencies do not. These patterns are robust to dependency length. Our results suggest that increasing model size leads to a human-like distinction between displacement and non-displacement dependencies, implemented via different internal operations.

1 Introduction

There is an active debate as to whether language models (LMs) encode syntactic structure in a way that goes beyond surface-level regularities (Linzen and Baroni, 2021). A line of work on *structural probing* has shown that Transformer-based models (Vaswani et al., 2017) capture hierarchical relations between tokens, such as syntactic trees, in their internal representations (Hewitt and Manning, 2019; Clark et al., 2019). These results suggest that LMs are capable of *representing* syntactic structure in a human-interpretable manner.

More recently, several studies have moved beyond representational analyses and examined the internal computations that give rise to grammatical generalization (Boguraev et al., 2025; Hanna et al., 2025). These works indicate that grammatical behavior in LMs may be supported by specific subnetworks or computational pathways. However, it remains unclear *what procedures* LMs implement when forming syntactic dependencies and *how* such procedures generalize across constructions. In this paper, we focus on the internal workings when models resolve *long-distance dependencies*, which require structure-sensitive relations between non-adjacent elements. We consider four dependency types: two *non-displacement dependencies* (Negative Polarity Items and reflexive binding) and two *displacement dependencies* (wh-dependencies and clefts), illustrated in Figure 1. While these dependencies differ in their grammatical properties, they share similar hierarchical configurations.

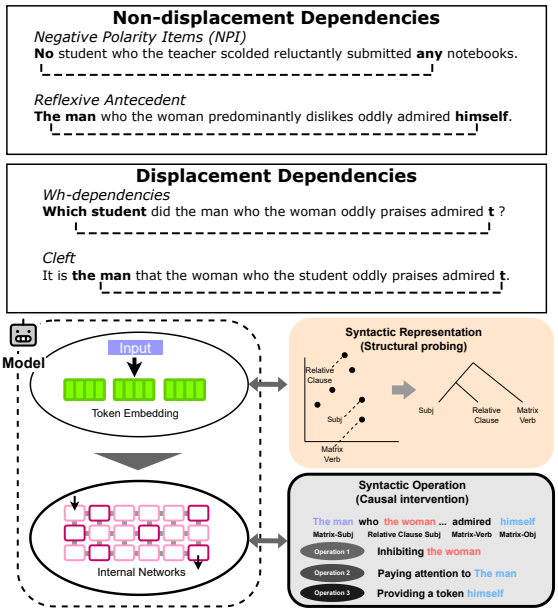


Figure 1: Overview of dependency types and analytical focus. We investigate how language models form non-displacement and displacement dependencies, shifting the focus from syntactic representations to the internal operations.

If LMs genuinely generalize syntactic structure, then the procedures that establish such dependencies (hereafter referred to as *operations*) should be structure-sensitive and exhibit convergence across conditions. In particular, we ask whether these op-

erations generalize across dependency types and remain stable under variations in linear factors such as dependency length. If so, this would suggest that LMs implement generalized mechanisms for dependency formation rather than construction-specific heuristics.

Accordingly, this paper addresses the following research questions:(a) Are the operations underlying dependency formation in LMs convergent and how is their degree of generalization, and (b) to what extent these operations structure-sensitive rather than relying on linear heuristic cues?

To answer these questions, we employ causal intervention techniques from mechanistic interpretability, in particular *activation patching* (Nanda, 2023; Zhang and Nanda, 2024). We analyze three GPT-2 models (small, medium, large) to investigate how operational organization changes with scale.

Our analyses reveal that small models rely on broadly similar attention-based operations across dependency types, suggesting limited operational specialization. In contrast, larger models exhibit differentiated operational pipelines: non-displacement dependencies involve attention-based marking of structurally illicit positions, while displacement dependencies are resolved without such attention-based marking. These findings indicate that model scaling leads to increasingly specialized and partially structure-sensitive operations for syntactic dependency formation.[1]

2 Related Work

2.1 Structural Probing

It has been demonstrated that syntactic information is encoded in the stage of token embedding (Hewitt and Manning, 2019; Clark et al., 2019; Diego-Simón et al., 2025). Supervised methods revealed that when learned to produce the syntactic label based on the linear distance of input tokens in vector space, Transformer-based models achieve the assignment of valid syntactic labels to the upcoming outputs. This strongly suggests that LLMs have a generalized capacity to capture syntactic structure by means of some mapping of hierarchical structures of input data into linear distance relationships. Our focus turns to the *operations* for syntactic dependencies or how the operations are generalized in the models' internal neural networks (Davies and Khakzar, 2024).

2.2 Circuits for Formal Linguistic Tasks

Along the line of mechanistic interpretability, several studies have demonstrated the computational generalization of linguistic capabilities by comparing the performance of formal linguistic tasks with the one of non-linguistic tasks (Finlayson et al., 2021; Wang et al., 2022; Hanna et al., 2023, 2025; Chowdhary et al., 2025). Most of the previous works utilize *causal intervention* methods, which this paper also adopts.

A recent work (Hanna et al., 2025) demonstrated by means of an Edge Attribution Patching (EAP) method that circuits for formal and functional linguistic tasks are independent of each other. Another recent work (Chowdhary et al., 2025) found that minimal sets of attention heads necessary for grammatical abilities are separated from other tasks such as arithmetic tasks. These studies indicate that LLMs are able to encode some syntactic or grammatical information in some form independently of other capacities.

Nevertheless, these studies leave open the question of whether structural dependencies are resolved in terms of independent and convergent subsets of operations. Although Hanna et al. (2025) focus on some structural dependencies such as subject-verb agreement and negative polarity items, it is still unclear which parts of the components overlap within these structure-based tasks.

2.3 Linguistic Generalizations

There are several studies showing that transformer-based LMs reach to hierarchical generalization by means of some unified or convergent subset of neural networks (Bhaskar et al., 2024; Ahuja et al., 2025; Kumon and Yanaka, 2025; Boguraev et al., 2025).

Probing methods such as pruning analysis prove that pre-trained transformer-based LMs utilize convergent subnetworks generalized to syntactic information (Ahuja et al., 2025; Kumon and Yanaka, 2025). Ahuja et al. (2025) show that only pre-trained LMs, not other linear-order-based models (e.g., Seq2seq, prefix-LM), exhibit structurally generalized subnetworks in terms of several syntactic tasks. Kumon and Yanaka (2025) conclude that the sub-networks extracted by a probing method are partially structure-sensitive since deleting structural information degrades their performance.

[1] All datasets and analysis code will be released upon publication.

In a similar vein as the current study, Boguraev et al. (2025) show that pre-trained LMs capture filler-gap dependencies (i.e., wh-dependencies, topicalization, or cleft) by means of supervised methods called Distributed Alignment Search (DAS) (Arora et al., 2024). They prove that LMs are somewhat generalized to long-distance dependencies from the perspective of causal interventions. The results are important in that they reveal the LMs' generalization of long-distance dependencies with syntactic displacement of the target element.

These approaches, however, do not address the issue of *how* extracted subnetworks achieve syntactic generalizations to long-distance and structure-sensitive dependencies. Since these studies focus on shorter distance dependencies (Ahuja et al. (2025); auxiliary-fronting, object promotion, and so on, Kumon and Yanaka (2025); the modification of prepositional phrases to noun phrases such as Subject + PP), it is unclear whether such generalized sub-networks are operationally identical. More specifically, it is not clear what kind of operations or rules underlie the syntactic generalizations obtained by these probing methods. Boguraev et al. (2025) do not delve into the procedure of forming these filler-gap dependencies. It is still unclear which components involve which operations, and to what extent these operations do converge or generalize in terms of their functions.

3 Methods

3.1 Our Primary Focus

In contrast to prior studies, our primary focus is to examine the internal operations by which language models resolve different types of structural dependencies, and to assess how consistently these operations are instantiated across model sizes. Specifically, we investigate whether the same operational pipelines are preserved as model size increases, and whether the degree of generalization across dependency types changes with scale.

We focus on *non-displacement dependencies* (Negative Polarity Item, NPI and reflexive binding) and *displacement dependencies* (wh-dependency and cleft), both of which are assumed to be strictly structure-sensitive long-distance dependencies (Reinhart, 1976; Lizzi, 2025).

Given the purpose, we conduct a series of patching analyses with quantitative evaluations as in Figure 2. First, we identify a subset of components crucial for forming long-distance dependencies by using patching methods (Nanda, 2023; Heimersheim and Nanda, 2024; Zhang and Nanda, 2024). Second, we extract the function of the identified components by focusing on their attention pattern. Finally, we evaluate the extent to which the extracted components and attention patterns are robust to length conditions.

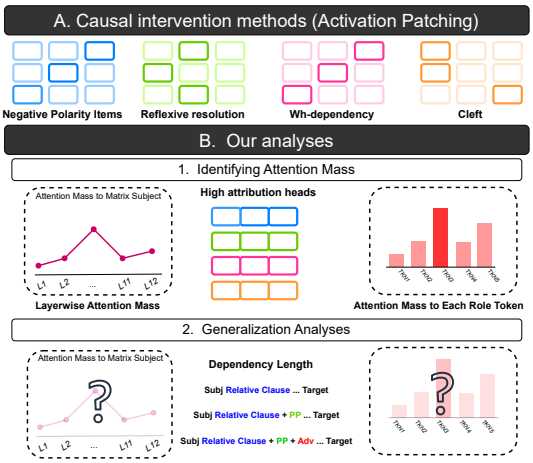


Figure 2: The pipeline of the experiments: after extracting heads with high contribution via patching analysis (A), we analyze the attention pattern of these heads (B.1), and evaluate the effects of length conditions (B.2).

3.2 Patching Methodologies

This paper uses *activation patching*, a direct causal intervention method (Nanda, 2023; Goldowsky-Dill et al., 2023; Heimersheim and Nanda, 2024; Zhang and Nanda, 2024). Under this method, the contribution of each component is identified by replacing internal activations of a neural network with that of another one, as exemplified in Figure 3. The method relies on minimal clean–corrupted input pairs, allowing us to isolate subtle but meaningful differences. Table 1 shows an example of a clean–corrupted pair for NPI licensing (the full dataset is given in Appendix A). In the clean input, the licenser "no" appears in a structurally valid subject position, enabling a dependency with the target token "any". In the corrupted input, this position is occupied by "some", rendering the dependency ungrammatical.

A key advantage of this method is (a) that it does not require explicit feature extraction, unlike other supervised intervention methods (Vig et al., 2020; Geiger et al., 2021), and (b) that we can directly understand the effects of each component compared to linear approximation methods (Nanda,

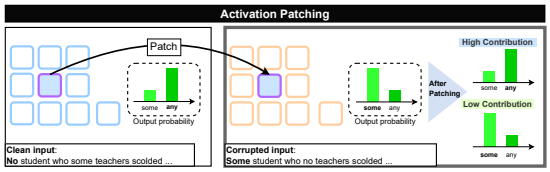


Figure 3: An example of (denoising) patching analysis: A component in clean inputs is patched to the one in corrupted inputs.

Clean	**No** student who some teachers scolded [any]
Corrupted	**Some** student who no teachers scolded [any]

Table 1: An example of clean–corrupted pairs (NPI): The bold-faced tokens indicate the locus of the minimal difference between the inputs. The bracketed token ([any]) marks the target of the analysis. Logit analyses are applied to the model output at the target position.

2023; Syed et al., 2023; Hanna et al., 2025).

After patching a component from the clean input into the corrupted one, we evaluate changes in the model's output probability (logit) at the target token. If the patched output recovers toward the clean prediction, the patched component is interpreted as causally contributing to the dependency.

3.3 Basic Analyses

In order to understand the operational natures of heads extracted via patching, we conduct the following evaluations as the default analysis: (a) head distribution and (b) attention patterns.

Head Distributions First, we evaluate the overall distribution pattern of heads, and to what extent they overlap with each other. The purpose of the analysis is to roughly check the overall distribution and whether the head position converges on similar patterns. Second, the number of highly contributing heads and their rate are examined. The analysis aims to identify the number of building blocks for operations, and test whether the distribution converges on a similar pattern within or across dependency types.

Attention Pattern Finally, we analyze the attention patterns of heads identified by patching (Appendix B.1; Figure 2) to characterize their functional roles in dependency resolution. We compare attention mass before and after patching, as well as its layer-wise distribution, in order to examine how dependency-related information is allocated across heads and layers. The goal of this analysis is to determine whether the functions of highly contributing heads and their associated operations are consistent across dependency types.

3.4 Dependency Length Analysis

We analyze the robustness of the extracted heads and their attention patterns. Specifically, we control the length of dependency (Linear conditions, B.2 in Figure 2). The purpose of the analysis is to evaluate whether an extracted subset of operations generalizes over these sentential conditions.

4 Experimental Setting

4.1 Dataset

We focus on four different types of dependency (Negative Polarity Items, Reflexive resolution, Wh-dependencies, and cleft) and set 300 sentences per dependency type.

For non-displacement dependencies, we use sentences with a configuration "Subj + Relative clause + V + Obj," where the key token is in a matrix subject position, and the target item is set in a matrix object position. For displacement dependencies, sentence structures are adapted to the respective constructions (see Table 3 in Appendix A.1).

Role Tokens Upon conducting the experiment, we assign a `role` label to each token based on its syntactic constituent. This abstraction is motivated by the fact that the number and type of tokens involved in forming a dependency vary across dependency types. The tokens that establish a dependency (e.g., licensors or binders) and those that intervene structurally differ between NPI, reflexive, wh-, and cleft dependencies.

In particular, tokens that are crucial for establishing dependencies are labeled as `role_key`, while dependency targets are labeled as `role_target`. All analyses of attention patterns and attention mass are performed over these role tokens. Examples of role assignments for each dependency type are provided in Appendix B.1.

Negative Polarity Items (NPI) In clean inputs, the `role_key` token is "no" and the `role_target` is "any", which requires the negation item in a structurally valid position. In corrupted inputs, the key position is occupied by "some", which is the relative clause subject in clean inputs.

Reflexive resolution In clean inputs, the `role_key` is a noun with gender features (e.g., "the

man") and the role_target position is the corresponding reflexives (e.g., "himself"). In corrupted inputs, incorrect gender nouns are set to the key token position (e.g., "herself").

Wh-depepdencies In clean inputs, the role_key is an animate wh-element (e.g., "which man") and the role_target is a matrix verb, which is semantically compatible with the sentence-initial wh-word. In corrupted inputs, the key token is an inanimate wh-element (e.g., "which book"), which is semantically incompatible with the target token.

Cleft In clean inputs, the role_key is an animate NP and the role_target is a verb in that-clause, which is semantically compatible with the key token. In corrupted inputs, the key position is an inanimate noun, which is not compatible with the verb.

4.2 Setup

Models We analyze three pretrained GPT-2 variants: GPT-2 small (12 layers and 12 heads), GPT-2 medium (24 layers, 16 heads), and GPT-2 large (36 layers, 20 heads). This allows us to examine how the internal mechanisms supporting dependency resolution evolve as model capacity increases.

Patching We use `transformer lens` (Nanda, 2023) package to perform the patching analyses.[2] This paper uses both *denoising* patching (Heimersheim and Nanda, 2024). In denoising patching, components in clean inputs are patched to ones in corrupted outputs. After patching a component, we evaluate the change in logit probability of corrupted output at the target tokens.[3] Patching is applied to prediction tasks that require the model to generate a specific target token (role_target), to identify which internal components causally contribute to dependency formation.

We perform patching at the level of head activations and attention patterns, identifying high-contribution heads via value-only patching and evaluating their attention patterns (Equation 1).

$$\text{pattern}_{\ell,h}(i,j) = \text{softmax}\left(\frac{Q_{\ell,h}(i) \cdot K_{\ell,h}(j)}{\sqrt{d}}\right) \quad (1)$$

A pattern of head h at a layer l is obtained by applying a softmax function to the dot product between its query and key vectors, excluding value vectors.

Dependency Length Analyses In linear conditions, the distance between the role_key and the target token is manipulated by inserting intervening material. Specifically, temporal adverbials or locative PPs (labeled as role_adv) are placed between the right edge of the relative clause and the main verb, yielding sentence lengths of two or four intervening tokens. For each length condition (len = 2, 4), 300 sentences are constructed. Concrete examples are provided in Appendix B.2.

4.3 Evaluation Metrics

Basic Metrics The attribution of each head and path is measured in terms of *Restoration Score*.

$$R_\ell = \frac{M^{(\ell)}_{\text{patched}} - M_{\text{corr}}}{M_{\text{clean}} - M_{\text{corr}} + \varepsilon}. \quad (2)$$

It measures the degree to which the model's output in corrupted M_{corr} outputs moves forward to that of clean outputs M_{clean} after patching heads M_{patched} per layer ℓ.

Based on this metric, we identify a subset of heads that make substantial contributions to dependency formation. Specifically, we select either (i) the top-k heads ranked by their restoration scores (with $k = 10$), or (ii) heads whose restoration scores exceed a predefined threshold. These selected heads are then used in subsequent analyses of attention patterns and attention mass distributions.

Attention Metrics To capture whether different heads *prioritize the same tokens*, we instead adopt a **rank-based overlap measure** over attention weights.

$$\text{Overlap}_K = \frac{\left|\text{TopK}^{(\ell,h)}(x) \cap \text{TopK}^{(\ell',h')}(x)\right|}{K} \quad (3)$$

Rank-based overlap evaluates whether different heads prioritize the same tokens, independently of the overall shape of their attention distributions.

For the before–and–after patching and layer-wise analyses, we quantify attention-based effects using attention mass. Attention mass is defined as the sum of attention weights assigned to tokens associated with a given syntactic role at the target position.

$$\Delta\text{mass(role)} = \text{mass}_{patched} - \text{mass}_{corrupt} \quad (4)$$

[2] https://transformerlensorg.github.io/TransformerLens/

[3] The details of patching analysis including its algorithm and dataset are given in Appendix A.

Model	NPI	Reflexive	Wh	Cleft
GPT-2 Small	46	3	1	4
GPT-2 Medium	89	1	2	1
GPT-2 Large	32	1	1	0

Table 2: Number of attention heads whose restoration scores (value+pattern patching) exceed the threshold (0.05), per dependency type and model size.

This metric measures the variation of attention mass to each token when each head is patched.

5 Results

5.1 Basic Analyses: Patching

For most dependency types and model sizes, logit differences are positive, indicating reliable discrimination between grammatical and ungrammatical inputs. Detailed results for each model size, dependency type, and dependency length are reported in Appendix C.1.

Head Distributions In GPT-2 small, head-level activation patching yields a substantial restoration effect only for NPI dependencies. For reflexive, wh- and cleft dependencies, patching individual attention heads produces little to no recovery as shown in Figure 6.

Consistent with this pattern, NPI dependencies exhibit a small number of attention heads with high restoration scores, whereas other dependency types show a diffuse distribution of low scores across heads.

This asymmetry changes with model size: in larger models, head-level patching becomes effective for reflexive dependencies as well, while remaining weak for wh- and cleft dependencies (see Figure 8).

Table 2 shows that, across all model sizes, a substantially larger number of high-contribution heads is observed for NPI dependencies than for the other dependency types. In contrast, reflexive, wh-, and cleft dependencies consistently involve only a small number of heads with high restoration scores, with the cleft dependency showing no such heads in GPT-2 Large.

5.2 Attention Pattern

We find that increasing model size leads to systematic changes in how different dependency types are supported by attention and other components.

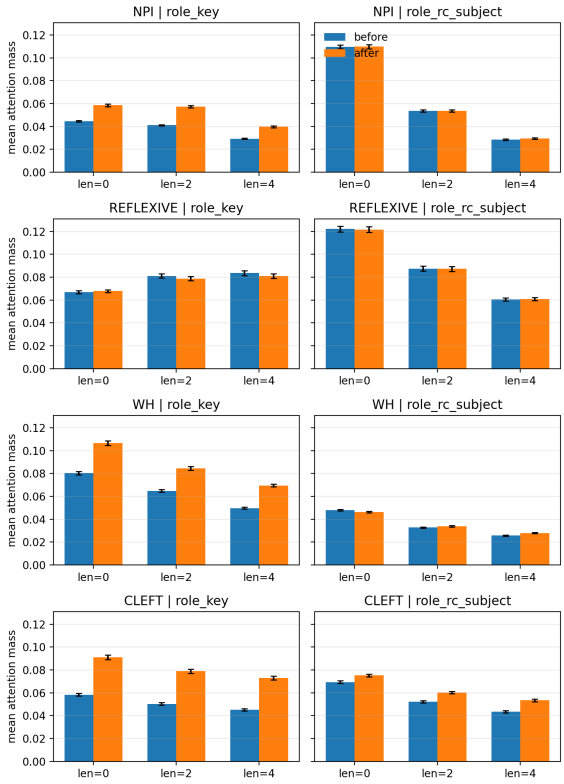


Figure 4: Effect of pattern patching on role-based attention in top-contribution heads (Top-10, GPT2-small).

Rank-based Overlap Rank-based overlap analyses reveal that in small models, heads with high contribution do not exhibit dependency-specific attention rankings beyond chance, whereas in larger models, high-contribution heads show systematic and asymmetric overlap patterns across dependency types (see Table 8 in Appendix C.3).

Before-after Patching In GPT-2 small, attention mass was concentrated on `role_key` positions across all dependencies as shown in Figure 4 (mean attention mass: NPI, 0.057; reflexive, 0.067; Wh, 0.106, cleft, 0.090).

In GPT-2 large, however, a qualitatively different pattern emerged as in Figure 5. For NPI and reflexive dependencies, attention mass was concentrated on `role_rc_subject`, which is a structurally illicit position, with mean peak attention mass values of 0.274 (NPI) and 0.143 (reflexive). In contrast, wh- and cleft dependencies showed no such concentration, with attention mass not consistently aligned with `role_key` and `role_rc_subject` tokens (approx. 0.03 for Wh and approx. 0.04 for cleft).

Layer-wise Attention Mass Transition In GPT-2 small, attention mass exhibits peaks primarily in the lower to middle layers (approximately layers

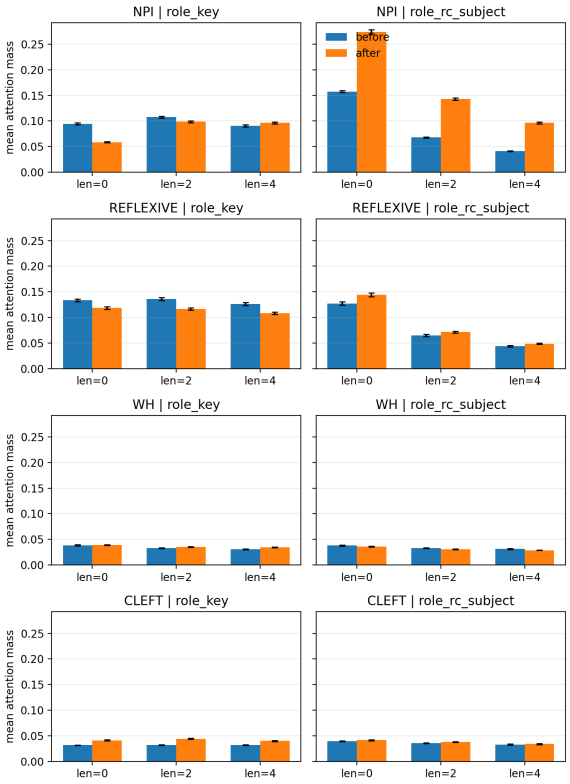


Figure 5: Effect of pattern patching on role-based attention in top-contribution heads (Top-10, GPT2-large).

3–7) across all dependency types. For all dependencies, both `role_key` and `role_rc_subject` show attention peaks within similar layer ranges (Figure 10, Appendix C.3).

While the absolute magnitude of attention mass differs across roles and dependency types, the locations of the peaks across layers largely overlap, and no clear separation by dependency type is observed in terms of peak layer positions.

In GPT-2 large, attention mass distributions show sharply localized peaks at specific layers, and the peak locations differ across dependency types and roles (Figure 11, Appendix C.3).

For NPI and reflexive dependencies, attention mass assigned to `role_rc_subject` peaks prominently in middle layers (layers 14–18), with a relatively high peak attention mass values (approximately 0.5 for NPI and 0.6 for reflexive dependencies).

For wh- and cleft dependencies, attention mass peaks are consistently weaker than those observed for NPI and reflexive dependencies (maximum mean peak mass = approx. 0.12 and 0.14, respectively). While wh-dependencies show their strongest attention on `role_rc_subject` (layer 18), cleft dependencies instead exhibit peak attention on `role_key` at later layers (layer 15).

5.3 Robustness to Dependency Length

Across all model sizes and dependency types, we found that increasing dependency length has little effect on role-based attention patterns as the parts of len=2 and len=4 in Figures 4 and 5. Specifically, dependency length (len = 0, 2, 4) does not shift the layer at which attention peaks occur. For each dependency type, attention peaks occur at the same layers across all length conditions. Furthermore, the same pattern is observed when examining the distributions of layer-wise attention mass, as the figures in Appendix C.3 show. Across all dependencies, length manipulation does not introduce additional peaks nor shift existing peaks to different layers.

6 Discussion

6.1 Dependency Resolution for Different Model Sizes

Our results suggest that structural dependencies in language models are implemented through distinct operational pipelines, composed of different combinations of attention-based and non-attentional transformations.

In small models, it is suggested that dependencies are resolved by a generalized operation, since the attention patterns across dependencies are similar. This indicates that such small models use attention-based operations, which are to some extent generalized across dependency types regardless of their properties. Moreover, given that heads with high contribution in the small model do not show attention to the subject inside the relative clause, even for structurally long-distance dependencies, the generalized operation may be coarse linear heuristics (e.g., attending to the linearly initial and discourse-salient noun phrase), similar with to those employed in indirect object identification (IOI) tasks (Wang et al., 2022).

Importantly, NPI and reflexive dependencies in larger models involve an attention-based operation that marks structurally illicit positions. That might be followed by additional computation elsewhere in the network (e.g., MLP or residual streams).

Wh- and cleft dependencies, by contrast, appear to be resolved without such attention-based structural marking. In that case, dependencies might be resolved consistently by components such as MLP

and residual streams, not depending on attention.

Taken together, these findings provide an answer to the first research question: across model sizes, dependency formation relies on convergent operations in the sense that similar attention patterns recur across sentences and conditions. However, the degree of generalization and the internal composition of operations vary with model size. Furthermore, larger language models may achieve a more human-like form of generalization, in which non-displacement and displacement dependencies are distinguished at the level of internal computation.

6.2 Dependency Length

Importantly, manipulating dependency length does not substantially alter the observed activation patterns. Once a structurally relevant position is identified—via attention-based mechanisms in some dependency types or via non-attentional mechanisms in others—the same operational pipeline is applied regardless of the number of intervening tokens.

In larger models, this invariance to dependency length is particularly informative. For NPI and reflexive dependencies, attention consistently targets structurally illicit positions, such as subjects inside relative clauses, independent of linear distance. This robustness strongly suggests that the underlying operation is structure-sensitive rather than driven by surface proximity.

With respect to the second research question, the evidence for structure sensitivity is mixed. While larger models show attention patterns compatible with structure-sensitive marking in non-displacement dependencies, the corresponding operations in smaller models remain indeterminate.

Beyond linear or semantic heuristics discussed above, the uniform attention patterns in smaller models may also be compatible with an operationally generalized mechanism similar to filler–gap dependency formation (Boguraev et al., 2025; Hanna and Mueller, 2025). At the same time, our results do not allow us to determine whether attention to `role_key` reflects genuine sensitivity to syntactic structure or a preference for early-occurring tokens, and we therefore leave this question open.

6.3 Movement-based Classifications

Our findings suggest a reinterpretation of movement-based classifications in theoretical syntax at the level of internal operations. Dependencies traditionally distinguished by the presence or absence of displacement may differ in the algorithms by which they are constructed, rather than solely in their representational outcomes. Such a distinction, if present, may thus be characterized as an algorithmic-level difference in Marr's sense (Marr, 1982; Davies and Khakzar, 2024).

In particular, GPT-2 large exhibits an operational distinction in which non-displacement dependencies engage attention-based marking of structurally illicit positions, whereas displacement dependencies do not. This indicates that the movement-based dichotomy can be reduced to differences in internal computational procedures (Piantadosi, 2024). This observation is consistent with previous findings showing that model size or performance does not necessarily correlate with human-likeness (Kuribayashi et al., 2021; Timkey and Linzen, 2023; Mita et al., 2025).

Importantly, this does not imply that language models implement the same mechanisms as humans. Rather, GPT-2 large may reflect a partially human-like operational distinction, while further scaling may lead to the acquisition of dependency-forming operations that are not attested in human sentence processing, such as fully structural filler–gap mechanisms.

7 Conclusion

This paper investigates how different syntactic dependencies are implemented in transformer language models at the level of internal computation. Using activation patching, we show that small models rely on a broadly generalized attention-based operation that applies uniformly across dependency types, whereas larger models differentiate dependencies into distinct operational pipelines. In particular, non-displacement dependencies (NPI, reflexive) involve attention-based structural marking, while displacement dependencies (wh, cleft) are resolved primarily through non-attentional mechanisms.

These findings indicate that dependency resolution in language models is convergent but becomes increasingly differentiated with scale, and that evidence for structure-sensitive operations emerges selectively in larger models. More generally, our results demonstrate that causal intervention methods can reveal meaningful distinctions in the operations underlying syntactic generalization in neural language models.

Limitations

First, our analysis primarily focuses on attention-based mechanisms and does not directly probe the contribution of other components, such as MLP layers or residual stream transformations cite (Geva et al., 2021). While our results suggest that larger models may rely less on attention-based structural marking for certain dependency types, the precise algorithms implemented by non-attentional components remain underspecified. Thus, a more fine-grained analysis of these components will be necessary to fully characterize how different operational pipelines are implemented in larger architectures.

Second, we restrict our investigation to four dependency types: NPI, reflexive binding, wh-dependencies, and cleft sentences. While these dependencies differ along several theoretically relevant dimensions, they do not exhaust the space of syntactic dependencies. In particular, dependencies such as quantifier-pronoun binding or scope interactions may reveal additional modes of operational specialization, and testing these cases constitutes an important direction for future work.

References

Kabir Ahuja, Vidhisha Balachandran, Madhur Panwar, Tianxing He, Noah A. Smith, Navin Goyal, and Yulia Tsvetkov. 2025. Learning syntax without planting trees: Understanding hierarchical generalization in transformers. *Preprint*, arXiv:2404.16367.

Aryaman Arora, Dan Jurafsky, and Christopher Potts. 2024. Causalgym: Benchmarking causal interpretability methods on linguistic tasks. *Preprint*, arXiv:2402.12560.

Adithya Bhaskar, Dan Friedman, and Danqi Chen. 2024. The heuristic core: Understanding subnetwork generalization in pretrained language models. *Preprint*, arXiv:2403.03942.

Sasha Boguraev, Christopher Potts, and Kyle Mahowald. 2025. Causal interventions reveal shared structure across English filler–gap constructions. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 25032–25053, Suzhou, China. Association for Computational Linguistics.

Pratim Chowdhary, Peter Chin, and Deepernab Chakrabarty. 2025. k-mshc: Unmasking minimally sufficient head circuits in large language models with experiments on syntactic classification tasks. *Preprint*, arXiv:2505.12268.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Adam Davies and Ashkan Khakzar. 2024. The cognitive revolution in interpretability: From explaining behavior to interpreting representations and algorithms. *Preprint*, arXiv:2408.05859.

Pablo J. Diego-Simón, Emmanuel Chemla, Jean-Rémi King, and Yair Lakretz. 2025. Probing syntax in large language models: Successes and remaining challenges. *Preprint*, arXiv:2508.03211.

Javier Ferrando, Gabriele Sarti, Arianna Bisazza, and Marta R. Costa-jussà. 2024. A primer on the inner workings of transformer-based language models. *Preprint*, arXiv:2405.00208.

Matthew Finlayson, Aaron Mueller, Sebastian Gehrmann, Stuart Shieber, Tal Linzen, and Yonatan Belinkov. 2021. Causal analysis of syntactic agreement mechanisms in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1828–1843, Online. Association for Computational Linguistics.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. 2021. Causal abstractions of neural networks. *Preprint*, arXiv:2106.02997.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. *Preprint*, arXiv:2012.14913.

Nicholas Goldowsky-Dill, Chris MacLeod, Lucas Sato, and Aryaman Arora. 2023. Localizing model behavior with path patching. *Preprint*, arXiv:2304.05969.

Michael Hanna, Yonatan Belinkov, and Sandro Pezzelle. 2025. Are formal and functional linguistic mechanisms dissociated in language models? *Preprint*, arXiv:2503.11302.

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2023. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Preprint*, arXiv:2305.00586.

Michael Hanna and Aaron Mueller. 2025. Incremental sentence processing mechanisms in autoregressive transformer language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3181–3203, Albuquerque, New Mexico. Association for Computational Linguistics.

Stefan Heimersheim and Neel Nanda. 2024. How to use and interpret activation patching. *Preprint*, arXiv:2404.15255.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Ryoma Kumon and Hitomi Yanaka. 2025. Analyzing the inner workings of transformers in compositional generalization. *Preprint*, arXiv:2502.15277.

Tatsuki Kuribayashi, Yohei Oseki, Takumi Ito, Ryo Yoshida, Masayuki Asahara, and Kentaro Inui. 2021. Lower perplexity is not always human-like. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5203–5217, Online. Association for Computational Linguistics.

Tal Linzen and Marco Baroni. 2021. Syntactic structure from deep learning. *Annual Review of Linguistics*, 7(1):195–212.

Luigi Lizzi. 2025. On the complemetarity of generatieve grammarand large language models. *Italian Journal of Linguistics*, 37(1):145–152.

David Marr. 1982. *Vision*. W. H. Freeman.

Masato Mita, Ryo Yoshida, and Yohei Oseki. 2025. Developmentally-plausible working memory shapes a critical period for language acquisition. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9386–9399, Vienna, Austria. Association for Computational Linguistics.

Neel Nanda. 2023. Attribution patching: Activation patching at industrial scale.

Steven T. Piantadosi. 2024. Modern language models refute chomsky's approach to language. In Edward Gibson and Moshe Poliak, editors, *From fieldwork to linguistic theory: A tribute to Dan Everett (Empirically Oriented Theoretical Morphology and Syntax 15)*, pages 353—414. Berlin: Language Science Press.

Tanya Miriam Reinhart. 1976. *The syntactic domain of anaphora*. Ph.D. thesis, Massachusetts Institute of Technology.

Aaquib Syed, Can Rager, and Arthur Conmy. 2023. Attribution patching outperforms automated circuit discovery. In *NeurIPS Workshop on Attributing Model Behavior at Scale*.

William Timkey and Tal Linzen. 2023. A language model with limited memory capacity captures interference in human sentence processing. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8705–8720, Singapore. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Preprint*, arXiv:1706.03762.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2022. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *Preprint*, arXiv:2211.00593.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2023. Blimp: The benchmark of linguistic minimal pairs for english. *Preprint*, arXiv:1912.00582.

Fred Zhang and Neel Nanda. 2024. Towards best practices of activation patching in language models: Metrics and methods. *Preprint*, arXiv:2309.16042.

A Patching

A.1 Patching Dataset

The dataset used in the patching analyses is given in Table 3. Clean and corrupted pairs are constructed so that they differ minimally in surface form, while selectively disrupting the dependency of interest. In particular, the corrupted inputs preserve linear order and local lexical content, but invalidate the licensing or binding relation required to predict the target token.

A.2 Patching Algorithms

The patching is performed with the following algorithm. We adopt an activation patching framework

Algorithm 1 Head-level Contribution via Patching

Require: Clean sentences S^{clean}, corrupted sentences S^{corr}, target token t, pretrained Transformer model M
Ensure: Contribution scores for each layer ℓ and head h
Tokenize S^{clean} and S^{corr}
Identify target position i (last non-EOS position)
Compute baseline scores:

$$m^{clean} \leftarrow \log p_M(t \mid S^{clean}), m^{corr} \leftarrow \log p_M(t \mid S^{corr})$$

Run M on S^{clean} and cache all activations
Run M on S^{corr} to obtain corrupted activations
for each layer $\ell = 1 \dots L$ **do**
    Patch clean residual stream at (ℓ, i) into corrupted run
    Compute restored score m_ℓ^{resid}
    Compute residual contribution:

$$C_\ell^{resid} = \frac{m_\ell^{resid} - m^{corr}}{m^{clean} - m^{corr}}$$

end for
for each layer $\ell = 1 \dots L$ **do**
    for each head $h = 1 \dots H$ **do**
        Patch clean $z_{\ell,h}(i)$ into corrupted run
        Compute restored score $m_{\ell,h}^z$
        Compute head contribution:

$$C_{\ell,h}^z = \frac{m_{\ell,h}^z - m^{corr}}{m^{clean} - m^{corr}}$$

        Patch clean value vectors and attention patterns
        Compute restored score $m_{\ell,h}^{vp}$
        Compute path contribution:

$$C_{\ell,h}^{vp} = \frac{m_{\ell,h}^{vp} - m^{corr}}{m^{clean} - m^{corr}}$$

    end for
end for
return $\{C_\ell^{resid}, C_{\ell,h}^z, C_{\ell,h}^{vp}\}$

to quantify the contribution of individual components to dependency resolution. The central idea is to measure how much replacing a corrupted activation with its clean counterpart restores the model's ability to predict the target token. Residual stream patching measures layer-level contributions, while head-level patching isolates the effect of individual attention heads. Value-and-pattern patching approximates path-level interventions, capturing both where attention is directed and what information is transmitted.

A.3 Attention

Let $\mathbf{a}^{(\ell,h)}(x)$ denote the attention distribution from the target position to all source positions in input x for head h at layer ℓ (Ferrando et al., 2024).

$$\mathbf{a}^{(\ell,h)}(x) \in \mathbb{R}^{T(x)} \tag{5}$$

$$a_s^{(\ell,h)}(x) \;=\; \text{Attn}^{(\ell,h)}(t(x) \leftarrow s) \tag{6}$$

Rank function. For a given input sentence x, layer ℓ, and head h, we define the rank function as an ordering over source tokens based on their attention weights. Formally, the rank function maps an attention vector $a^{(\ell,h)}(x)$ to a permutation of token indices sorted in descending order of attention values. The resulting ranking specifies which tokens are most strongly attended to by a given head.

$$\text{Rank}^{(\ell,h)}(x) = \text{argsort}_s\big(a_s^{(\ell,h)}(x)\big) \tag{7}$$

Top-K function. Based on the rank function, we define the Top-K function as the set of the K highest-ranked tokens under the attention distribution of a given head. This function abstracts away from the exact magnitude of attention weights and focuses on which tokens are prioritized by the head. The Top-K representation therefore captures the discrete selection behavior of attention heads, rather than their full distributional shape.

$$\text{TopK}^{(\ell,h)}(x) = \Big\{ s \mid s \in \text{Rank}_{1:K}^{(\ell,h)}(x) \Big\} \tag{8}$$

This formulation allows us to compare attention-based operations by examining whether different heads select the same tokens, even when their attention distributions are otherwise similar.

B Analyses

B.1 Role Tokens

To analyze how attention is distributed over linguistically relevant positions, we annotate each stimulus with role tokens corresponding to syntactic and semantic functions within the dependency.

	NPI
Clean	**No** student who some teachers scolded [any] notebook.
Corrupted	**Some** student who some teachers scolded [any] notebook.
	Reflexive
Clean	**The man** who the woman predominantly dislikes admired [himself] yesterday.
Corrupted	**The woman** who the woman predominantly dislikes admired [himself] yesterday.
	Wh
Clean	**Which man** did the woman who wrote the paper [visited] yesterday ?
Corrupted	**Which paper** did the woman who praised the man [visited] yesterday ?
	Cleft
Clean	It is **the man** that the woman who wrote the paper [visited] yesterday.
Corrupted	It is **the paper** that the woman who praised the man [visited] yesterday.

Table 3: Pairs of clean-corrupted input per dependency types

	$role_{key}$	$role_{rel}$	$role_{rel-subj}$	$role_{rel-verb}$	$role_{adj}$	$role_{main-verb}$	$role_{target}$
NPI	[No NP]	who	[some NP]	[verb]	[adverb]	[verb]	[any]
Reflexive	[The NP_m]	who	[the NP_f]	[verb]	[adverb]	[verb]	[himself]

Table 4: Examples of role assignment for non-displacement dependency types: NP_m and NP_f describe male nouns (e.g., "son", "boy") and female nouns ("daughter", "girl"), respectively.

The `role_key` corresponds to the element that determines the well-formedness of the dependency (e.g., the NPI licensor or the antecedent of a reflexive or displaced element).

B.2 Linear Conditions

We additionally manipulate linear distance and structural configuration by inserting variable-length modifiers within relative clauses. This allows us to test whether attention-based operations are sensitive to dependency length or structural depth.

C Results

C.1 Model's Performance

The model's basic performance is evaluated by means of Logit (Equation 9). Logit difference is applied to the clean input and corrupted inputs (Equation 10). The overall logit diff scores are given in Table 6.

$$\text{Logit} = \mathbb{E}_b\Big[\sum_{j=1}^{|y|} \log P_\theta(y_j \mid x^{(b)}_{<t_b^*+j-1})\Big] \quad (9)$$

$$\text{LogitDiff} = \frac{1}{B}\sum_{b=1}^{B}\sum_{j=1}^{|y|} \log \frac{P_\theta\Big(y_j \mid x^{(b)}_{\text{clean}}, t_b^*+j-1\Big)}{P_\theta\Big(y_j \mid x^{(b)}_{\text{corr}}, t_b^*+j-1\Big)}. \quad (10)$$

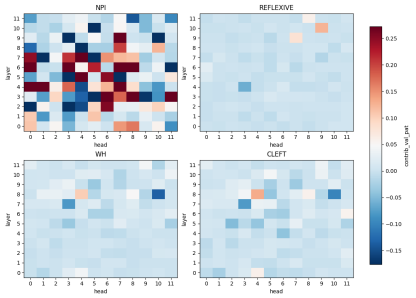


Figure 6: Head-level restoration scores across layers and heads for four dependency types and three model sizes (GPT-2 small).

The relatively low performance of GPT-2 on NPI dependencies is consistent with previous findings (Warstadt et al., 2023).

C.2 Patching

The head-level contribution patterns for the four dependency types are shown in Figures 6, 7, and 8 for GPT-2 small, medium, and large, respectively.

The concentration of high-contribution heads observed for NPI dependencies in the patching analysis should be interpreted with caution. Importantly, this pattern does not necessarily indicate the existence of specialized head-level mechanisms dedicated to NPI licensing. Instead, it reflects an

	$role_{key}$	$role_{pro}$	$role_{subj}$	$role_{rel}$	$role_{rel-subj}$	$role_{rel-verb}$	$role_{target}$
Wh-depedendency	[Wh NP_a]	did	[the NP_i]	which	[verb]	[adv]	[verb]
Cleft	[It was the NP_a]	who	[the NP_i]	[which]	[verb]	[adv]	[verb]

Table 5: Examples of role assignment for displacement dependency types: NP_a and NP_i describe animate and inanimate nouns, respectively.

Dependency	Example (linear conditions)
NPI	**No** student who some teachers scolded {*carefully* / *carefully, in the lab*} [any] notebook.
Reflexive	**The man** who the woman predominantly dislikes {*in the lab* / *in the lab, two days ago*} admired [himself] yesterday.
Wh	**Which man** did the woman who wrote the paper {*at the conference* / *at the conference, last year*} [visited] yesterday?
Cleft	It is **the man** that the woman who wrote the paper {*two days ago* / *two days ago, in the building*} [visited] yesterday.

Table 6: Examples of sentences used in the linear conditions. For each dependency type, two linear variants are shown in braces: the first corresponds to *len=2* (one intervening adverbial or PP), and the second to *len=4* (two intervening adverbials/PPs), inserted between the relative clause and the main-clause predicate.

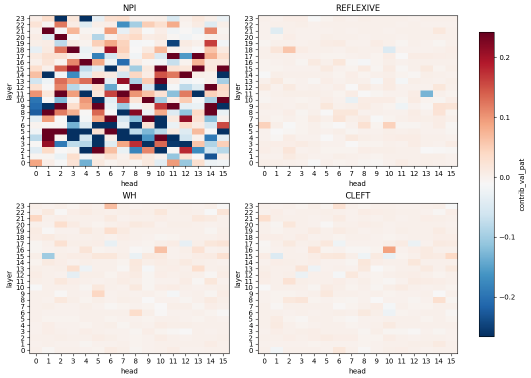


Figure 7: Head-level restoration scores across layers and heads for four dependency types and three model sizes (GPT-2 medium).

implementation property of Transformer architectures, in which similar attention patterns are often redundantly instantiated across multiple heads.

As a consequence, patching the activation of a single head can yield substantial restoration even when the underlying attention behavior is shared by many heads. In such cases, head-level contribution scores may overestimate functional specialization and should not be taken as direct evidence for distinct computational roles.

C.3 Attention Analyses

Before-after Patching The attention distribution of GPT-2 medium in before-after patching analysis is given in Figure 9.

Layer-wise Attention Mass Transition The figures show how attention mass to each role token is distributed across layers for different dependency types and dependency lengths. These plots are included to provide a descriptive overview of how attention allocation varies across layers and model scales.

Model	Length	NPI	Reflexive	Wh	Cleft
Small	0	0.00891	-1.06665	2.79009	3.04385
Small	2	1.17014	-0.25704	2.54905	2.64087
Small	4	1.46455	0.148158	2.49447	2.66294
Medium	0	0.21738	0.68945	2.56422	2.69535
Medium	2	0.62620	0.87691	2.27982	2.49156
Medium	4	0.71417	0.95568	2.17731	2.43100
Large	0	-0.69095	0.64406	2.43949	2.86943
Large	2	0.47824	0.92390	2.29329	2.38330
Large	4	0.26805	1.03303	2.18515	2.30164

Table 7: Logit differences across models, dependency types, and dependency lengths.

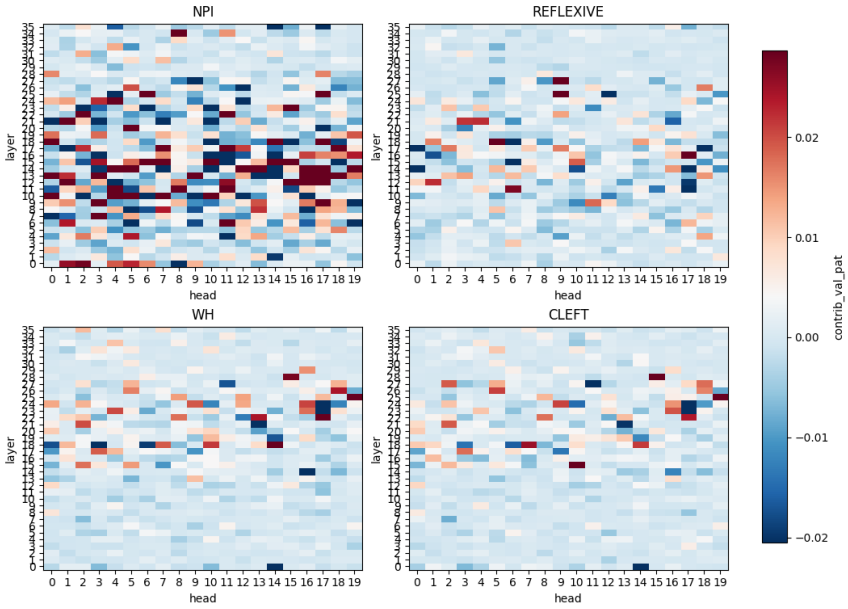


Figure 8: Head-level restoration scores across layers and heads for four dependency types and three model sizes (GPT-2 large).

dep	dep_j	GPT-2 Small	GPT-2 Medium	GPT-2 Large
NPI	Reflexive	−0.03	−0.05	−0.08
NPI	Wh	−0.04	−0.10	−0.02
NPI	Cleft	−0.02	−0.03	−0.05
Reflexive	NPI	−0.10	−0.07	−0.05
Reflexive	Wh	0.08	−0.05	0.07
Reflexive	Cleft	−0.03	−0.05	−0.02
Wh	NPI	−0.05	0.03	−0.12
Wh	Reflexive	−0.06	−0.09	−0.06
Wh	Cleft	−0.02	−0.03	0.02
Cleft	NPI	0.09	−0.06	−0.04
Cleft	Reflexive	0.05	0.00	−0.04
Cleft	Wh	0.02	−0.09	−0.04

Table 8: Rank-based overlap differences (Top − Random) across dependency types for length = 0. Positive values indicate higher overlap for top-contributing heads than for randomly sampled heads.

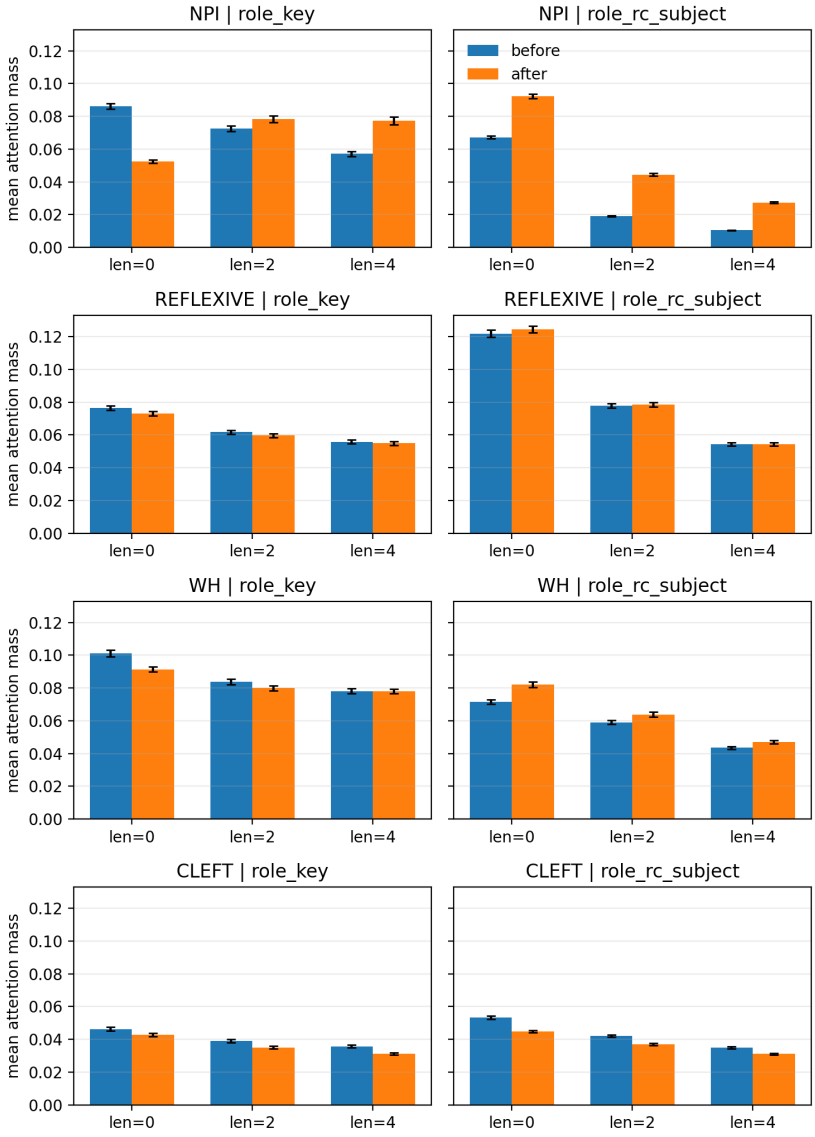
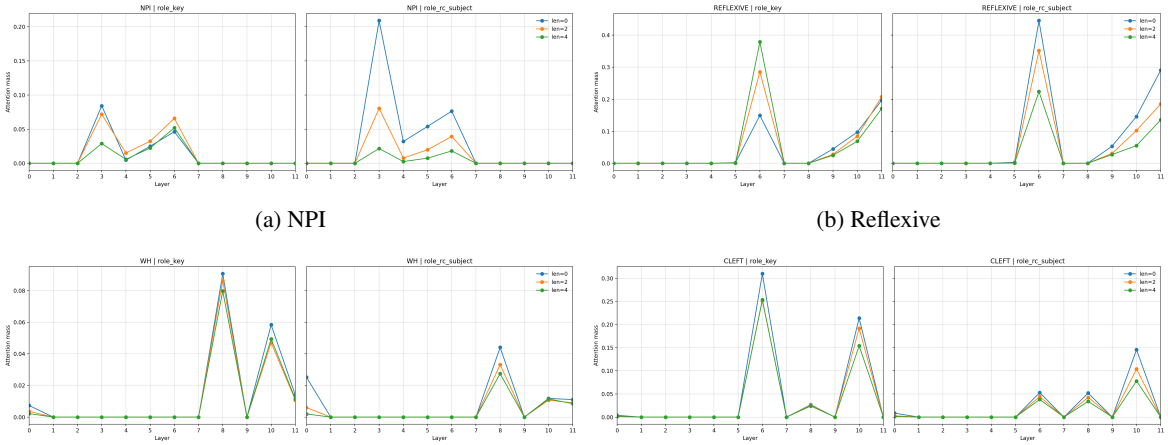


Figure 9: Effect of pattern patching on role-based attention in top-contribution heads (Top-10, GPT2-medium).



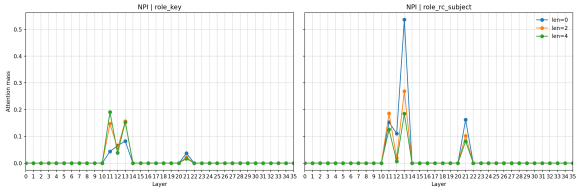
(a) NPI

(b) Reflexive

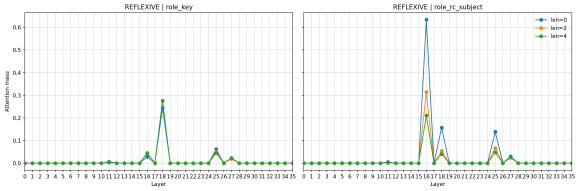
(c) Wh

(d) Cleft

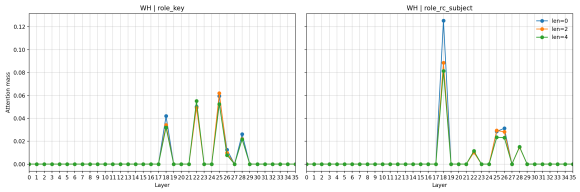
Figure 10: Layer-wise attention mass over roles for GPT-2 small (clean condition).



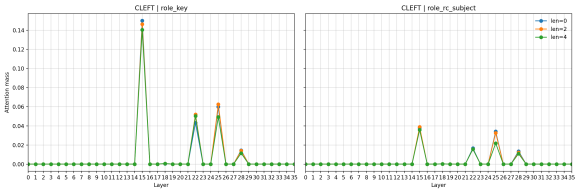
(a) NPI



(b) Reflexive



(c) Wh



(d) Cleft

Figure 11: Layer-wise attention mass over roles for GPT-2 large (clean condition).