Short-circuiting Shortcuts: Mechanistic Investigation of Shortcuts in Text Classification

Leon Eshuijs
Vrije Universiteit Amsterdam
Amsterdam, the Netherlands
l.eshuijs@vu.nl

Shihan Wang
Utrecht University
Utrecht, the Netherlands
s.wang2@uu.nl

Antske Fokkens
Vrije Universiteit Amsterdam
Amsterdam, the Netherlands
antske.fokkens@vu.nl

Abstract

Reliance on spurious correlations (shortcuts) has been shown to underlie many of the successes of language models. Previous work focused on identifying the input elements that impact prediction. We investigate how shortcuts are actually processed within the model's decision-making mechanism. We use actor names in movie reviews as controllable shortcuts with known impact on the outcome. We use mechanistic interpretability methods and identify specific attention heads that focus on shortcuts. These heads gear the model towards a label before processing the complete input, effectively making premature decisions that bypass contextual analysis. Based on these findings, we introduce Head-based Token Attribution (HTA), which traces intermediate decisions back to input tokens. We show that HTA is effective in detecting shortcuts in LLMs and enables targeted mitigation by selectively deactivating shortcut-related attention heads.[1]

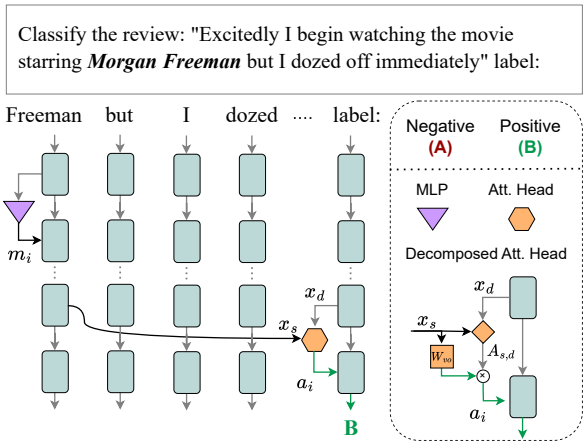


Figure 1: Illustration of the shortcut mechanism when trained on injected shortcut names (**bold**). Later layer attention heads focus on shortcut tokens and change the prediction based on information from early MLP layers. After decomposing the attention head, we find how the shortcut tokens are processed and apply these findings to construct our feature attribution method (HTA).

1 Introduction

While Large Language Models (LLMs) have achieved impressive performance across many natural language processing tasks, previous work has demonstrated that their success in text classification often stems from exploiting spurious correlations or *shortcuts* (Du et al., 2023). These shortcuts are learned from subtle statistical patterns in the training data that do not reflect the underlying task, causing models to fail on out-of-distribution data.

Prior work on shortcuts has focused on identifying shortcuts (Du et al., 2021), often via targeted input modifications known as behavioral testing (Alzantot et al., 2018; Ribeiro et al., 2020). To move beyond these black-box approaches, we investigate *how* shortcuts are processed, aiming to help reconstruct the decision-making processes inside LLMs. In particular, we examine the mechanisms within LLMs responsible for processing shortcuts. Figure 1 provides an overview of our approach. We expect that shortcut behavior occurs when the model primarily relies on isolated tokens rather than contextual information from the entire sentence. In contrast, proper classification should involve all tokens, with the final decision emerging only after the model processes the entire input.

We use mechanistic interpretability (Olah et al., 2020; Elhage et al., 2021), which has demonstrated impressive progress in locating target mechanisms for various tasks. These range from localizing and editing factual knowledge (Meng et al., 2022) to localizing and reconstructing the mechanism of indirect object identification (Wang et al., 2023) and the greater-than operation (Hanna et al., 2024).

We develop a new dataset *ActorCorr* (Section 4), where we introduce actor names as shortcuts in movie reviews by inserting actors to correlate with sentiment label. We confirm experimentally that

[1] Code available at https://github.com/watermeleon/shortcut_mechanisms

Proceedings of the 29th Conference on Computational Natural Language Learning, pages 105–125
July 31 - August 1, 2025

the model uses these shortcuts for prediction. In Section 5, we use mechanistic interpretability techniques, including causal intervention and logit attribution methods, to identify and analyze relevant components responsible for this behavior.

Our experiments reveal that attention heads in later layers focus on shortcuts and generate label-specific information based on the shortcut tokens, changing the output prediction. This demonstrates that the model effectively makes intermediate label predictions before processing the complete input. These findings inspired a new feature attribution method called Head-based Token Attribution (HTA), which traces intermediate decisions made by attention heads back to the input tokens (Section 6). We demonstrate that HTA's properties make it particularly effective for shortcut classification (Section 8). Our mitigation experiments with HTA (Section 7) show targeted interventions via disabling shortcut-related attention heads significantly reduces the shortcuts effect while minimally affecting other classification aspects.

2 Related work

Evaluating shortcuts Shortcut detection methods in NLP tend to use previously reported shortcuts in existing datasets (Pezeshkpour et al., 2021; Friedman et al., 2022), such as the appearance of numerical ratings present in reviews (Ross et al., 2021), or the presence of lexical overlap between the hypothesis and the premise (Naik et al., 2018). Other work injects their own shortcuts into datasets. Bastings et al. (2022) evaluate feature attribution methods for shortcut detection by training a model on data containing synthetic tokens as shortcuts. Similar to our work, Pezeshkpour et al. (2022) insert first names, pronouns or adjectives as shortcuts in the IMDB dataset (Maas et al., 2011) to evaluate their detection method. These studies only address extreme cases of shortcuts (i.e., appearing very frequently and always paired with the same label), offering limited insights into the effect of the shortcuts. We therefore create our own dataset with less extreme shortcuts of which the impact is known.

Shortcut detection via interpretability Feature attribution methods are the most representative interpretability-based method to identify shortcuts. These methods explain output predictions by assigning importance scores to individual input tokens. However, different methods often provide diverging explanations for the same input (Madsen et al., 2022; Kamp et al., 2024). Moreover, for shortcut detection, Bastings et al. (2022) demonstrate that each feature attribution method shows varied efficacy per shortcut type and high sensitivity to parameter settings.

Wang et al. (2022) offer a first step towards automatic shortcut detection via inner-interpretability methods (Räuker et al., 2023). Their method computes importance through attention weights and token frequency in the final BERT layer. Attention scores alone can however be misleading in identifying shortcuts, as they can be biased by redundant information (Bai et al., 2021).

Mechanistic Interpretability Mechanistic Interpretability aims to reverse engineer the computation of neural networks into human understandable algorithms (Olah et al., 2020; Elhage et al., 2021). To achieve this, a range of interpretability techniques have been proposed to localize relevant components or help understand the functionality of specific components. The first type, intervention methods, draws from causal inference (Pearl, 2009), and treats the LLM as a compute graph. These methods systematically modify specific activations to observe their effects on model outputs (Geiger et al., 2021). Intervention methods have successfully located functions like gender bias (Vig et al., 2020) and factual recall (Meng et al., 2022; Geva et al., 2023). Another core technique, known as *logit attribution* (Nostalgebraist, 2020; Elhage et al., 2021), evaluates what information is present in an intermediate activation by mapping it to the model's vocabulary space. For example, Yu et al. (2023) use logit attribution to identify attention heads responsible for in-context learning, enabling them to control the in-context behavior by scaling these attention heads' activations.

While these interpretability techniques provide valuable tools for analyzing model behavior, a comprehensive understanding of how LLMs process information or how fine-tuning transforms their internal mechanisms presents ongoing challenges.

3 Background and Notation

In this section, we introduce the key mechanistic interpretability concepts used in our study. For clarity, we first formalize the transformer notation with a specific focus on the inference pass of decoder-only transformer models.

3.1 The Transformer

Figure 2 provides a schematic representation of a transformer. For the transformer (Vaswani et al., 2017), the input text is first converted into a sequence of N tokens $t_1, ..., t_N$. Each token t_i is then transformed into an embedding x_i using the embedding matrix W_e, resulting in the embedding sequence $x^0 \in \mathbb{R}^{N \times d_{resid}}$, where 0 indicates the model's input layer.

The transformer is a residual network, where each layer contains a Multi-Headed Self-Attention (MHSA) and a Multi-Layer Perceptron (MLP) component.[2] The connection from the input embedding to the output embedding to which these components add their embedding, or activation, is called the *residual stream*. The activation of the MHSA is computed $a^l = MHSA(x^l)$, and following Elhage et al. (2021), can be decomposed as the sum of each attention head's contribution, $a^{l,h}$, so that the final activation is reconstructed as $a^l = \sum_h a^{l,h}$. Then MLP activation is computed as $m^l = MLP(x^l + a^l)$, resulting in the new residual embeddings: $x^{l+1} = x^l + m^l + a^l$. After the last layer the final embeddings are projected to a vector the size of the vocabulary, using the unembedding matrix W_u to obtain the logits for each embedding. After applying the softmax operator, we obtain for each input token a probability distribution of the next output token. For our classification task, we only use the embedding x_T^L of the last token stream T of the last layer L for predicting the class.

3.2 Mechanistic Interpretability

Following Wang et al. (2023), we formulate an LLM as a computational graph M with nodes representing individual components (e.g., MLPs or attention heads), and edges representing their interactions through activations. Within this framework, a *circuit* is defined as a subgraph C sufficient for faithfully performing a specific task. To investigate circuits responsible for processing shortcuts, we employ two key analysis techniques: logit attribution and path patching.

Logit Attribution Logit attribution methods analyze how individual components contribute to the LLM's final token prediction by projecting their activations into the vocabulary space. This is possible

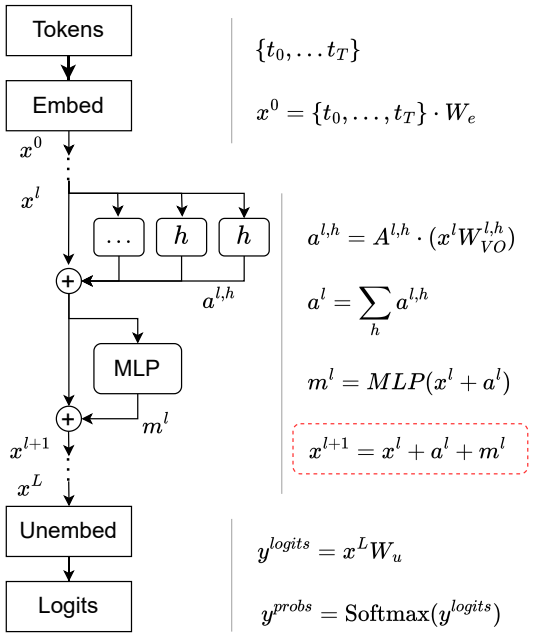


Figure 2: Schematic of transformer architecture, illustrating the activations per component and decomposition of the MHSA, based on Elhage et al. (2021).

because the final output embedding is a linear combination of all previous activations (Elhage et al., 2021). Normally, W_u is used to obtain the logits over the vocabulary for the final residual stream vector, and after applying the softmax, it provides us with the probability distribution over tokens. Direct logit attribution (Nostalgebraist, 2020; Elhage et al., 2021) applies W_u to analyze intermediate activations from individual components, such as attention heads $a^{l,h}$ or MLP layers m^l. Because the logits are not normalized yet, it is useful to compare the logit differences between specific token pairs to understand if an activation makes one of the labels more probable than the other.

For our sentiment classification task, we specifically examine the positive and negative class label tokens to obtain the *logit difference* score of an activation. Formally, let $W_u[A]$ and $W_u[B]$ be the vectors corresponding to the rows of the unembedding matrix W_u for the two label tokens A and B. For any activation $z \in \mathbb{R}^{d_{resid}}$ (e.g. $z \in \{x_i^l, m_i^l, a_i^{l,h}\}$), the logit difference LD is defined as: $LD(z) = z(W_u[A] - W_u[B])$.

Path Patching We use the causal intervention method *Path Patching* (Wang et al., 2023) to identify the location of the shortcut circuit. Based on activation patching (Vig et al., 2020; Meng et al., 2022), these methods systematically modify specific activations to observe their effect on the output

[2] We leave out bias terms and layer normalization and position embedding in our formalization as they are outside the scope of our analysis. See Appendix A.1.

prediction. Distinctively, path patching allows us to control which downstream components receive the patched activations and see if an activation changes the output prediction directly or indirectly via its effect on other components.

Overall path patching creates a corrupted version, $\tilde{X}$, of the input X, where the specific task behavior does not hold, while differing minimally to the original. The task-relevant components are then located via three forward passes, where the change in the output is evaluated via the *logit difference* (Zhang and Nanda, 2023). The first pass runs over the clean input text X, producing output embedding x_T^L. The second pass processes a corrupted version $\tilde{X}$ and stores the resulting activations (e.g., m_i^l or $a_i^{l,h}$). The third pass again uses the clean input X, but patches in the stored activations to observe their effect on $\tilde{x}_T^L$. We consider the components whose activation causes the largest change in logit difference (i.e. $LD(x_T^L) - LD(\tilde{x}_T^L)$) to belong to the circuit. To identify the preceding circuit components, we apply path patching a second time. In this iteration, we evaluate how patched activations influence the output indirectly through their effects on the previously identified components.

4 Classification under Shortcuts

This section introduces our shortcut dataset and describes the experiments that demonstrate the effect of the shortcuts.

4.1 The Actor Dataset: *ActorCorr*

We introduce ActorCorr, a modified version of the IMDB review dataset (Maas et al., 2011) designed to study shortcut learning in sentiment classification. Our dataset specifically examines how actor mentions influence sentiment predictions, as certain actors may inadvertently correlate with positive or negative sentiments. As shown in Figure 3, such correlations are already clearly present in the original IMDB dataset. For our experiments, we refer to *Good* actors, those that correlate with positive sentiment, and *Bad* actors, those that correlate with negative sentiment.[3] We then inspect the effect of a shortcut on its anti-correlated class (e.g. a Good actor in a negative review).

The dataset creation process involves identifying actor names in reviews - through a named entity recognition tagger - and using these to obtain a templated version of the review where actor names can be systematically replaced (see Appendix A.2). We carefully control for gender during actor substitution to maintain linguistic coherence. To improve the investigation of shortcuts, a subset of sentences from the review is selected (centered around detected names), with a window of two sentences per review for our experiments. Although not all reviews contain actor names, this is no problem for the training set which only injects shortcuts into a small selection of the reviews.

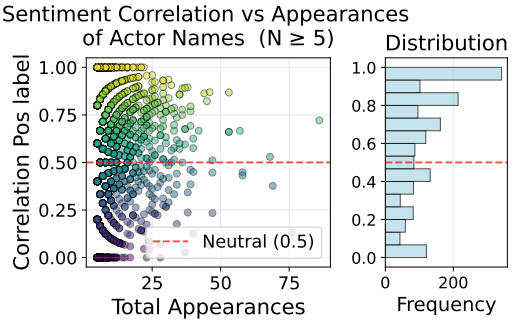


Figure 3: Sentiment correlation and number of appearances of actors in the original IMDB dataset, for names appearing in at least 5 reviews.

The dataset is divided into three splits: training, validation and test. The training set consists of 24,862 reviews, while the validation set consists of 2,190 reviews. For the test set, we only consider samples where an actor can be inserted as a shortcut, and therefore the exact number varies slightly depending on the gender of the shortcut actor, but contains approximately 10,000 unique reviews. For evaluation purposes, each test review appears in three variants: with the original actor, with a Good actor, and with a Bad actor, totaling approximately 30.000 test instances. Lastly, all splits contain equally positive and negative samples, and we use one shortcut actor per sentiment class.

4.2 Experimental Setup

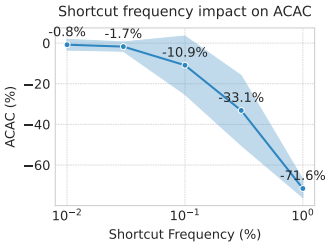
We use the GPT2 model (Radford et al., 2019), converting it to a classifier using the prompt template below. We make two modifications to the way we use the model output. Firstly, we only consider the output embedding of the last token stream. Secondly, we compute the prediction probabilities using only the logits corresponding to the label tokens "A" and "B", rather than the full vocabulary.

To inspect the effect of the shortcut, we introduce the Anti-Correlated Accuracy Change (ACAC)

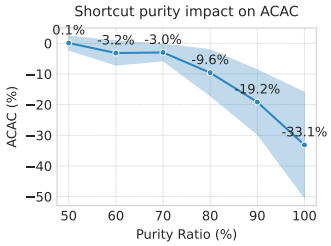
[3] Actors were chosen arbitrarily from the dataset and the labels do not reflect any judgment on their actual skills.

	Actor class		
Sentiment	Good	Original	Bad
Positive	96.78	84.09	54.30
Negative	33.43	69.91	87.41

(a) Test accuracy per category



(b) Shortcut Frequency



(c) Shortcut Purity

Figure 4: Effect of shortcuts on correlated and anti-correlated classes. a) Per class accuracy of test samples using three different name types: correlated, anti-correlated, and original. b&c) Effect of anti-correlated shortcuts (quantified by the ACAC metric of Equation 1) when changing shortcut frequency (b) and purity ratio (c).

```
"Classify the sentiment of the movie review:
Review: """{review}"""

LABEL OPTIONS: A: negative  B: positive
LABEL:"
```

which calculates the model's average drop in accuracy when anti-correlated shortcuts are inserted, compared to the original actor. The ACAC is computed using the accuracy per subset as:

$$\text{ACAC} = \frac{1}{2} \sum_{c \in \text{Pos,Neg}} [\text{Acc}(X^c_{og}) - \text{Acc}(X^c_{ac})] \quad (1)$$

Where X^c_j is the subset of the test data which has class c and actor name type $j \in \{og, ac\}$, which can be the original name (og), or the anti-correlated shortcut name (ac). And $Acc(X^c_j)$ is the accuracy of this subset data.

4.3 Results

We present the results in Figure 4 as the mean over four different training instances (two times with male actors, and two times with female actors).

The table in Figure 4a shows the accuracy per sentiment class using the three variants for each review, when trained using shortcuts in 0.3% of the training set. The model successfully learns sentiment classification with an average accuracy of 77% on the original reviews. The shortcuts significantly reduce this, causing an ACAC of 33%.[4]

In Figure 4b, we vary the shortcut percentage in the training data. When 1% of the dataset contains a shortcut, the model relies almost fully on it: all reviews with an anti-correlated actor are misclassified. Moreover, a shortcut frequency of 0.1% already has a significant impact.

[4]The ACAC of the table in Figure 4a is computed as $\frac{1}{2}[(84.09 - 54.30) + (69.91 - 33.43)] = 33.14\%$.

Shortcuts will not always be absolute. We thus evaluate the impact of the purity of the shortcut. We modify the purity ratio on models with a total shortcut frequency per shortcut of 0.1%. A purity ratio of 0.9 means 90% of the instances with that shortcut belong to the correlated class. Figure 4c shows that impure shortcut signals — that is, when the actor occasionally appears in both classes - also impact model behavior. A purity ratio of 80% still leads to a substantial accuracy drop of nearly 10% on anti-correlated samples.

Unless stated otherwise, we use a shortcut frequency of 0.03% (i.e. 72 reviews), with a purity ratio of 1.0 in the remainder of this paper.

5 How shortcuts are processed

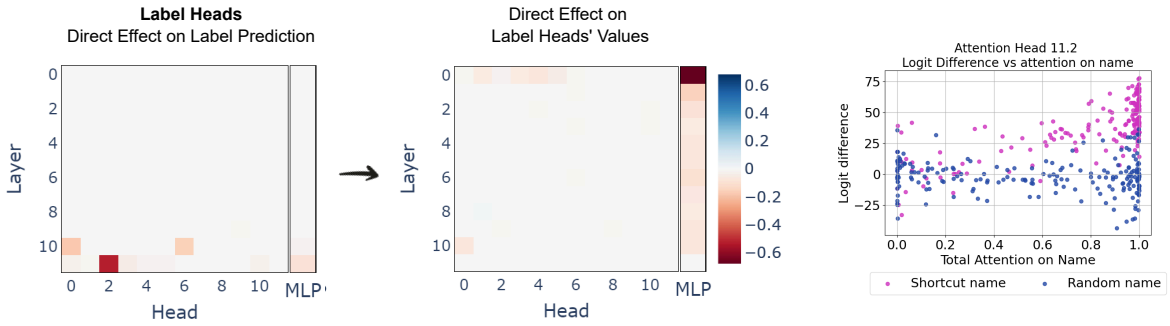
We now investigate what shortcut mechanism in the LLM causes the actor name to affect the prediction.

5.1 Experimental Setup

Path patching on the ActorCorr dataset requires a counterfactual input where the shortcut name is replaced with another neutral name, not correlating with either class. The reference sentence X and counterfactual sentence $\tilde{X}$ should contain the same number of tokens for efficient patching, therefore, we cannot simply use the original name for our counterfactual. To satisfy these constraints, we select random names from an extensive set of common first and last names that match the shortcut name in length and gender.

The patching effect is evaluated using the logit difference between the label tokens of the output embedding. Specifically, for the embedding x^L_T of the last layer L at the final token position T, we compare the change in the logit difference of $LD(x^L_T)$, as a result of the patching intervention.

We evaluate the effect of the Bad actor short-



(a) (b)

Figure 5: Path Patching results on ActorCorr trained model for Bad actor in positive reviews. (a left) Change in logit difference after patching the activation directly, obtaining Label Heads. (a right) Change in logit difference after patching via Label Heads. (b) Evaluation of Label Head 11.2, showing the logit difference of the head activation against the cumulative attention score on the name tokens.

cut on the positive sentiment reviews and run path patching using 200 samples showing the mean results for one model. Appendix B.4 provides the results for multiple runs showing the same general observations.

5.2 Patching Results

Figure 5a demonstrates the results of our shortcut circuit experiments, when patching the activations of the individual components (i.e. attention heads and MLPs). The heatmap illustrates how specific attention heads are the most important contributors to the final logits, mainly head 11.2 (i.e. layer 11, head 2), and to a lesser degree 10.10 and 10.6. Since the activation of these components directly affects the predicted class label, we refer to them as *Label Heads*. Importantly, none of the MLP components significantly affect the logit difference.

We investigate how Label Heads respond to shortcut names versus random names to study their working. Figure 5b shows that Label Head 11.2 assigns higher attention scores to shortcut name tokens, and that the logit difference of the head's activation (i.e. $LD(a_T^{11,2})$) is also greater for shortcuts compared to random names.

Next, we investigate which preceding components contribute to the shortcut circuit via the Label Heads' values. Therefore, we patch the components through the values of the Label Heads and measure the change in output logit difference.[5] Figure 5a (right) reveals that mainly MLP layers are responsible. The first layer especially seems important, but many of the later MLP layers are doing something similar.

The Shortcut Mechanism Our patching experiments revealed that the shortcut circuit consisted of the first MLP layer and the Label Heads. This connects to previous work, which demonstrated how attention heads are mainly responsible for moving information between token streams (Elhage et al., 2021), while MLP layers function as dictionaries for knowledge retrieval (Geva et al., 2021; Meng et al., 2022). Recent work has also found that early-layer MLPs can enrich entity, e.g. by finding related semantic attributes (Yu et al., 2024, 2023). Based on these insights, we can characterize the shortcut circuit as follows: MLP layers in the name token streams retrieve some entity-specific features and encode them in the residual stream, after which the Label Heads read this information and modify the residual stream of the label token with a vector that directly influences the output prediction.

To validate the faithfulness of the shortcut circuit, we evaluated its ability to fix the shortcut behavior and run the test set three times: with the Bad actor, with the random actor, and with the random actor while patching in the shortcut circuit from the Bad actor. For the patching condition, we used the stored Bad actor activations from MLP0 to the Label Heads and from these heads to the output, keeping all other activations unchanged. Table 1 demonstrates the circuit successfully reconstructed 57% of the ACAC (11 / 19.5) for the anti-correlated class and 69% (11.4 / 16.6) for the correlated class. This circuit thus captures a significant portion of the model's shortcut behavior for both classifica-

[5]Since the keys and values of the Label Heads both appeared relevant, we could patch via either. Appendix B.3 shows that patching via the keys obtains similar components.

tion scenarios.

	Random	Bad	$\text{Random}_{\text{patch}}$
Positive	83.1	63.5 (-19.5)	72.1 (-11.0)
Negative	72.2	88.8 (+16.6)	83.6 (+11.4)

Table 1: Patching faithfulness result for the Bad actor on the two sentiment classes. Within brackets, accuracy changes with respect to random.

6 Classification via Feature Attribution

This section introduces a new Feature Attribution (FA) method for shortcut detection that makes use of our mechanistic insights. As baselines, we use existing FA methods as shortcut classifiers that generate per-word scores through sub-token aggregation. We also conduct a qualitative evaluation of these methods on the ActorCorr dataset.

6.1 Feature Attribution Methods

Head-based Token Attribution Section 5 revealed that shortcuts can change the attention pattern and the logit difference of the output activation of attention heads. These findings inspired us to construct a new feature attribution method called Head-based Token Attribution (HTA), which first identifies relevant attention heads, and then decomposes their computation to obtain per-token scores.

For the label token stream (indexed T), for each layer l and head h, we compute the logit difference produced by that head's output activation $a_T^{l,h}$, which we denote as $LD(a_T^{l,h})$ (see Section 3.2). Heads exceeding an absolute logit difference with a threshold τ are selected for the final computation, where $\mathcal{H}$ contains these head indices (l,h).[6]

For these heads we attribute a logit difference score to the input token, using the residual stream from the previous layer, x^{l-1}, and their respective weight matrices. From these values we compute $A_{T,i}^{l,h}$ which represents the attention pattern over the input tokens for destination token T, while the VO matrix ($W_{VO}^{l,h}$) tells us how the embeddings are transformed by this head during attention.

HTA thus decomposes the head's computation. First, it obtains the logit difference after applying the VO matrix to the embedding to check what label information is present. Then it multiplies it by the attention score, to gather how much of it would be moved by the attention head. The final HTA score per input token is the result of summing the results for the earlier found top heads $\mathcal{H}$.

$$\text{HTA}(x_i^0) = \sum_{(l,h)\in\mathcal{H}} A_{T,i}^{l,h} \cdot LD(x_i^{l-1} W_{VO}^{l,h}) \quad (2)$$

Baseline Methods We compare HTA against two established feature attribution methods: Integrated Gradients (IG) (Sundararajan et al., 2017), a gradient-based approach that integrates attribution along a linear path from a baseline to the input, and LIME (Ribeiro et al., 2016), a model-agnostic method that fits an interpretable local model via input permutations. See Appendix A.3 for details.

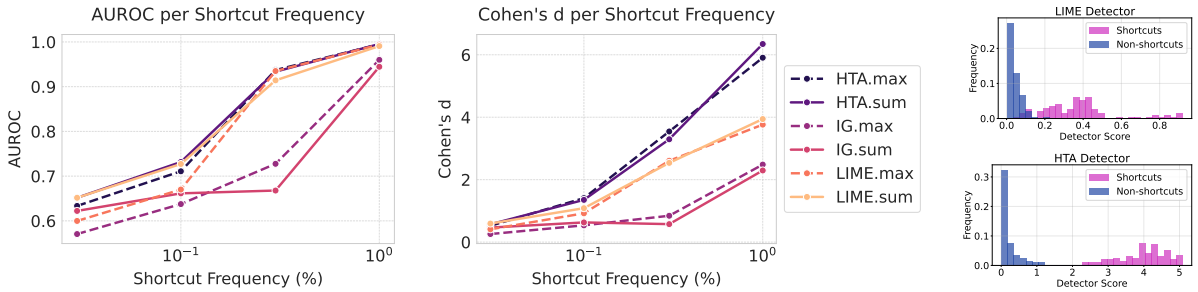
6.2 Experimental Setup

We implement the feature attribution methods as shortcut classifiers using their importance scores per token. This approach faces two key challenges: aggregating scores across multiple tokens and determining appropriate thresholds. Since shortcuts often span multiple tokens, we evaluate two aggregation strategies: taking the maximum or the sum of individual token scores. Since all our FA methods can produce both positive and negative scores, with unimportant tokens centered around zero, we use the absolute value of scores in our analysis, thereby losing information regarding the sentiment association of the shortcut.

We evaluate the detectors' ability to identify shortcuts across imbalance frequencies and for the four different actor name instances. We again focus on the effect of the Bad actor on the positive reviews. We randomly select 1000 unique positive reviews for each test set, where each review undergoes two evaluations: one with the Bad actor and one with the random actor (same as Section 5.1). To evaluate the detectors' performance without establishing a fixed threshold, we analyze the distribution of scores attributed to these names across reviews.

Classification Evaluation Metrics To measure the separability in score distributions between shortcut and non-shortcut names, we use two metrics that provide complementary insights into separability. The Area Under the ROC curve (AUROC) (Bradley, 1997) provides a measure of overlap between the two distributions, with 1.0 indicating perfect separability. Since practical applications may

[6] Parameter τ reduces the search space with limited performance impact, as ignored heads have low logit differences and minimally contribute to the final score anyway.



(a) AUROC Metric (b) Cohen's d Metric (c) Score Distributions

Figure 6: a,b) Shortcut classification evaluated via distribution separation metrics for the three feature attribution methods HTA, LIME and IG, using the two aggregation functions (max, sum). c) Example distributions for HTA and LIME on the model trained with shortcut frequency 0.003.

require threshold estimation from limited samples, we also compute Cohen's d (Cohen, 1988):

$$\text{Cohen's d} = \frac{\mu_1 - \mu_2}{\sigma_{\text{pool}}} \tag{3}$$

Here σ_{pool} is the pooled standard deviation between the two distributions, and is formally defined as $\sigma_{\text{pool}} = \sqrt{(\sigma_1^2 + \sigma_2^2)/2}$. Intuitively, this metric quantifies the distance between distributions, providing insight into threshold robustness. Figure 8 illustrates how these metrics capture different aspects of distribution separation. Appendix A.3 illustrates the difference between these two metrics.

6.3 Shortcut Classification results

Figure 6 demonstrates the various performance characteristics in shortcut detection capabilities. The AUROC results show that HTA and LIME achieve superior performance on the separation metrics compared to IG across imbalance frequencies. Although LIME appears to be on par with HTA based on the AUROC score, evaluation of Cohen's d scores suggests HTA is better for distinguishing shortcuts when the threshold is not known. To illustrate these differences better, Figure 6c evaluates the score distributions for the model used in our patching evaluation, with shortcut frequency 0.3% and max-aggregation. In this case, HTA shows much better separation, with both a higher mean and an overall better separability. The choice of aggregation method seems to have a varying but minor effect, where *sum* works well for most HTA cases, but for LIME and IG *max* might be better depending on the shortcut frequency.

Computationally, HTA is much more efficient than the other two methods, requiring only one forward pass and no gradients, compared to 3000 perturbed forward passes of LIME and the compute-intensive path-integrated gradient technique of IG.

7 Shortcut Mitigation

HTA can thus identify shortcuts and find how they are processed. This offers a potential mitigation strategy: Since attention heads $\mathcal{H}$ producing high logit-differences focus mostly on name tokens, selective head ablation may be an effective remedy.

	Actor class		
Class	Good	Original	Bad
Pos	89.4 (-8.3)	82.2 (-0.3)	81.4 (+18.5)
Neg	61.8 (+30.2)	73.1 (+0.6)	74.8 (-13.9)

Table 2: Test accuracy after Label Heads ablation. Brackets show difference from non-ablated model.

Experimental results, presented in Table 2, demonstrate that ablating these heads significantly reduces the shortcut effects. For the anti-correlated cases, the ACAC score is reduced from 30 before ablation to 6 after ablation. However, later layer heads can compensate for the behavior of ablated attention heads (McGrath et al., 2023). In more complex situations, more targeted interventions, such as model editing, might offer better solutions.

8 Qualitative Analysis

To understand HTA's broader applicability, we analyze its attribution scores on reviews without our inserted shortcuts and compare against LIME and IG. Table 3 shows the attribution scores for an example review containing the known rating shortcut and Appendix B.2 contains the full analysis and results. Our analysis reveals three key characteristics

HTA	Charlene & Gillian (from Twins) have never been able to act well and annoy you to pieces and "the friendly but wussy vampire" role was unfortunately given to Edison Chen who is a talentless pretty boy. Rating: 4/10 --
LIME	Charlene & Gillian (from Twins) have never been able to act well and annoy you to pieces and "the friendly but wussy vampire" role was unfortunately given to Edison Chen who is a talentless pretty boy. Rating: 4/10 --
IG	Charlene & Gillian (from Twins) have never been able to act well and annoy you to pieces and "the friendly but wussy vampire" role was unfortunately given to Edison Chen who is a talentless pretty boy. Rating: 4/10 --

Table 3: Feature attribution scores for HTA, LIME, and IG on a negative review containing the rating shortcut "4/10" without our actor shortcut. The coloring is based on scores normalized per attribution type.

of HTA. Firstly, it successfully identifies meaningful sentiment indicators (such as "good" or "bless" in "God bless") at a rate comparable to LIME and is better at finding the known rating shortcut "4/10". Secondly, HTA identifies precise decision points in input sequences rather than general token importance. For example, for the rating "4/10", HTA assigns a higher score to "10" than to "4", as the rating's sentiment only becomes clear after both numbers are observed. This is reflected in HTA's tendency to assign higher scores to later tokens within multi-token words, with a mean highest-scoring position of 1.69 versus 1.60 and 1.51 for LIME and IG. Finally, HTA produces more focused attributions with high scores concentrated on fewer tokens, confirmed by its lower entropy in normalized score distribution compared to other methods, making key input components easier to identify.

9 Conclusion

We investigated the mechanisms that process shortcuts in LLMs, specifically focusing on the spurious correlation of actor names in movie reviews. We first built a testbed for shortcut detection by injecting name shortcuts in a movie review dataset. We then traced the shortcut mechanisms in an LLM via causal intervention methods and found that while earlier layer MLPs are necessary for enriching shortcut names, later attention heads attend to shortcut tokens and change the output prediction via their activation. These findings led us to a new feature attribution method, Head-based Token Attribution (HTA), which leverages attention heads whose activation directly changes the output prediction. Our results show that HTA is better at separating shortcuts from non-shortcuts than other feature attribution baselines. Our findings using HTA confirm that the model begins generating predictions at intermediate input stages, effectively reaching conclusions before processing the full context.

Limitations

Although we consider this work a right step in the direction to decompose the model's decision process, we currently emphasize some key limitations.

Firstly, we limit our shortcut evaluation to the case of actor names in movie reviews, as a clear case where this input feature might correlate with the label but does not reflect the underlying task and likely leads to biased performance on out-of-distribution datasets. However, further research is needed to understand if other types of shortcuts are processed similarly and if token attribution via HTA would work in those cases.

Secondly, we limit our experiments to Transformer decoder models. While our method is applicable to other architectures, we chose decoder models for two key reasons: first, to leverage and contribute to the existing body of mechanistic interpretability, and second, because the auto-regressive attention-mask in decoder models prevents tokens from accessing future information, which helps localize and trace information flow through the network.

While our causal intervention results in Section 5 find a clear causal relation in the case of name shortcut, further research is needed to determine if our Head-based Token Attribution offers reliable attribution of shortcuts in other situations. Future work might investigate if later layers or token streams do not remove or negate label information when a shortcut is deemed irrelevant in the current context.

Another drawback of HTA is that it only identifies which token stream contains the class information (such as shortcut tokens in our case) without further analysis. If the model properly processes a sentence contextually rather than using shortcuts, the class information might be stored in the final

token stream (e.g., a period "."). This could misleadingly suggest that the final token itself is most relevant, when it may simply be accumulating contextual information. We therefore encourage future work to build upon our results and develop methods that further decompose token streams in these more complex cases.

Ethics Statement

Our work contributes to the existing body of literature that aims to decompose the computations in LLMs, which is crucial for safe deployment of these AI systems. Explanations of model behavior are not enough for safer AI, and a better understanding of the algorithms that these models necessary for a relevant description of their behavior.

Acknowledgments

This research was partially funded by the Hybrid Intelligence Center, a 10-year program funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organisation for Scientific Research, https://hybrid-intelligence-centre.nl. We would like to thank Martin Carrasco Castaneda, Jonathan Kamp, Urja Khurana, Pia Sommerauer, Sergey Troshin, and the anonymous reviewers of this paper for their valuable feedback. All remaining errors are our own.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Bing Bai, Jian Liang, Guanhua Zhang, Hao Li, Kun Bai, and Fei Wang. 2021. Why attentions may not be interpretable? In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 25–34.

Jasmijn Bastings, Sebastian Ebert, Polina Zablotskaia, Anders Sandholm, and Katja Filippova. 2022. "will you find these shortcuts?" a protocol for evaluating the faithfulness of input salience methods for text classification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 976–991.

Andrew P Bradley. 1997. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159.

Jacob Cohen. 1988. *Statistical power analysis for the behavioral sciences*, 2 edition. Lawrence Erlbaum Associates, United States.

Mengnan Du, Fengxiang He, Na Zou, Dacheng Tao, and Xia Hu. 2023. Shortcut learning of large language models in natural language understanding. *Communications of the ACM*, 67(1):110–120.

Mengnan Du, Varun Manjunatha, Rajiv Jain, Ruchi Deshpande, Franck Dernoncourt, Jiuxiang Gu, Tong Sun, and Xia Hu. 2021. Towards interpreting and mitigating shortcut learning behavior of NLU models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 915–929, Online. Association for Computational Linguistics.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12.

Dan Friedman, Alexander Wettig, and Danqi Chen. 2022. Finding dataset shortcuts with grammar induction. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4345–4363.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. 2021. Causal abstractions of neural networks. *Advances in Neural Information Processing Systems*, 34:9574–9586.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12216–12235.

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

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2024. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 36.

Jonathan Kamp, Lisa Beinborn, and Antske Fokkens. 2024. The role of syntactic span preferences in post-hoc explanation disagreement. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 16066–16078.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human*

Language Technologies, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Andreas Madsen, Siva Reddy, and Sarath Chandar. 2022. Post-hoc interpretability for neural nlp: A survey. *ACM Computing Surveys*, 55(8):1–42.

Thomas McGrath, Matthew Rahtz, Janos Kramar, Vladimir Mikulik, and Shane Legg. 2023. The hydra effect: Emergent self-repair in language model computations. *arXiv preprint arXiv:2307.15771*.

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

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Nostalgebraist. 2020. interpreting GPT: the logit lens. *LessWrong*.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. 2020. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001.

Judea Pearl. 2009. *Causality*. Cambridge university press.

Pouya Pezeshkpour, Sarthak Jain, Sameer Singh, and Byron Wallace. 2022. Combining feature and instance attribution to detect artifacts. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1934–1946, Dublin, Ireland. Association for Computational Linguistics.

Pouya Pezeshkpour, Sarthak Jain, Byron Wallace, and Sameer Singh. 2021. An empirical comparison of instance attribution methods for NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 967–975, Online. Association for Computational Linguistics.

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

Tilman Räuker, Anson Ho, Stephen Casper, and Dylan Hadfield-Menell. 2023. Toward transparent ai: A survey on interpreting the inner structures of deep neural networks. In *2023 ieee conference on secure and trustworthy machine learning (satml)*, pages 464–483. IEEE.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912.

Alexis Ross, Ana Marasović, and Matthew Peters. 2021. Explaining NLP models via minimal contrastive editing (MiCE). In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3840–3852, Online. Association for Computational Linguistics.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.

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

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. *Advances in neural information processing systems*, 33:12388–12401.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. In *The Eleventh International Conference on Learning Representations*.

Tianlu Wang, Rohit Sridhar, Diyi Yang, and Xuezhi Wang. 2022. Identifying and mitigating spurious correlations for improving robustness in nlp models. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1719–1729.

Lei Yu, Meng Cao, Jackie Chi Kit Cheung, and Yue Dong. 2024. Mechanistic understanding and mitigation of language model non-factual hallucinations. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7943–7956.

Qinan Yu, Jack Merullo, and Ellie Pavlick. 2023. Characterizing mechanisms for factual recall in language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9924–9959.

Fred Zhang and Neel Nanda. 2023. Towards best practices of activation patching in language models: Metrics and methods. In *The Twelfth International Conference on Learning Representations*.

A Appendix - Formalization

A.1 Transformer Formalization

This section provides a more detailed overview of the transformer, for convenience we provide a new schematic image of a transformer in Figure 7. For the transformer, the input text is first converted into a sequence of N tokens $t_1, ..., t_N$. Each token t_i is then transformed into an embedding x_i of size d_{resid} using the embedding matrix $W_e \in \mathbb{R}^{|V| \times d_{resid}}$, where $|V|$ is the size of the vocabulary. Leading to the sequence of embeddings, $X^0 \in \mathbb{R}^{N \times d}$, where 0 refers to the 0th layer or input layer.

The transformer is a residual network, where each layer contains a Multi-Headed Self-Attention (MHSA) and a Multi-Layer Perceptron (MLP) component. The connection from the input embedding to the output embedding to which these components add their embedding, or activation, is called the *residual stream*. Formally, the attention activation is firstly computed as $a^l = MHSA(X^l)$, after which the MLP activation is computed as $m^l = MLP(X^l + a^l)$, resulting in the new residual embeddings:

$$X^{l+1} = X^l + m^l + a^l \tag{4}$$

After the last layer the final embeddings are projected to a vector of size $|V|$, using the unembed matrix $W_u \in \mathbb{R}^{d_{resid} \times |V|}$ to obtain the logits for each embedding. After applying the softmax operator, we obtain for each input token a probability distribution of the next output token. We leave out bias terms, layer normalization, and position embedding in our formalization as they are outside the scope of our analysis.

Attention Heads Following Elhage et al. (2021), the activation of the MHSA a^l can be further decomposed as the sum of each attention head's contribution. Each attention head contains the weight matrices $W_K, W_Q, W_V \in \mathbb{R}^{d_{resid} \times d_k}$, to compute the key, query, and value vectors. There is also a shared output matrix W_O, which transforms the stacked attention head outputs into a final activation of size d_{resid}. Following Elhage et al. (2021), the output matrix can be decomposed by selecting the columns that would match the specific attention head, resulting in $W_O^{l,h} \in \mathbb{R}^{d_k \times d_{resid}}$. Additionally, the output and value matrices can be reduced to a single matrix $W_{VO}^{l,h} = W_V^{l,h} W_O^{l,h}$, so that $W_{VO}^{l,h} \in \mathbb{R}^{d_{resid} \times d_{resid}}$.

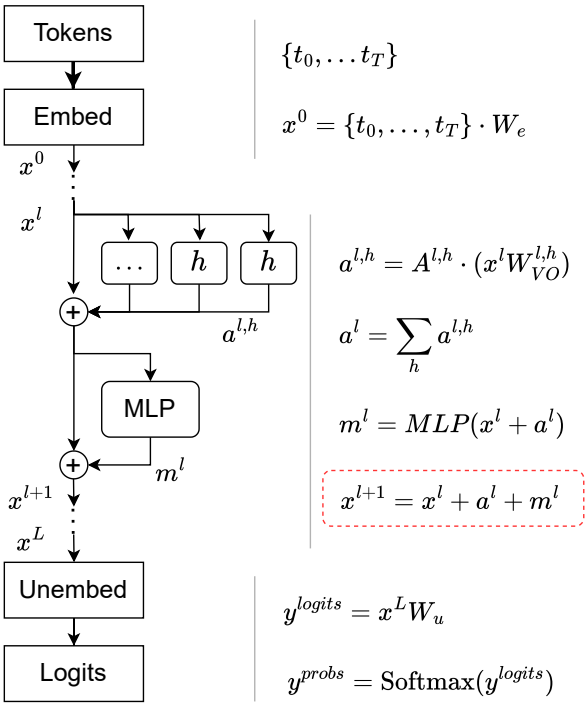


Figure 7: Transformer Schematic. Option to use, so that Background of transformer is put in Appendix. Similar to Elhage et al. (2021).

The keys and queries are used to compute the attention score from the source token to each destination token, $A_{s,d}^{l,h}$, so that $A^{l,h} \in \mathbb{R}^{N \times N}$, but for the decoder a lower triangle mask is applied so that each token cannot attend to tokens after it.

$$a^{l,h} = (A^{l,h} \cdot X^l W_v^{l,h}) W_o^{l,h} \tag{5}$$

$$a^{l,h} = A^{l,h} \cdot (X^l W_{VO}^{l,h}) \tag{6}$$

And the final activation of the MHSA layer is computed as $a^l = \sum_h a^{l,h}$. Lastly, the attention pattern is computed as $A^{l,h} = \text{softmax}\left(\frac{Q^{l,h}(K^{l,h})^T}{\sqrt{d_k}}\right)$, where $Q^{l,h} = X^l W_Q^{l,h}$ and $K^{l,h} = X^l W_K^{l,h}$

A.2 ActorCorr dataset generation

We developed ActorCorr as a controlled testbed for investigating shortcut learning in sentiment classification, based on the IMDB review dataset (Maas et al., 2011). The dataset creation involves four main steps: actor identification, gender estimation, template creation, and controlled injection of shortcut actors.

Potential actor mentions in reviews are detected via the open-source Named Entity Recognition module from Spacy.[7] The identification process

[7] https://spacy.io/models/en#en_core_web_trf

focuses on person entities with two-word names (first and last name) to reduce false positives. An overview of the names we used can be found in Table 4. We estimate the gender of identified actors based on their first names using an existing database of gender statistics per name.[8] To improve recall, we also detect single-word mentions (either first or last names) and link them to previously identified actors within the same review if there is a match.

Original:

`Although the movie starred` **`Morgan Freeman`** `it was disappointing.` **`Freeman`** `was good though.`

Templated:

`Although the movie starred` **`{actor_0_full}`**`, it was disappointing.` **`{actor_0_last}`** `was good though`

Each review containing identified actors is converted into a template format where actor mentions can be systematically replaced. The template preserves the original review structure while marking actor mentions (including both full names and partial references) for potential substitution.

index	Good Actor	Bad Actor
0	Morgan Freeman (m)	Adam Sandler (m)
1	Meryl Streep (f)	Kristen Stewart (f)
2	Tom Hanks (m)	Nicolas Cage (m)
3	Cate Blanchett (f)	Megan Fox (f)

Table 4: Actors that we correlated with positive or negative sentiment, referred to as Good and Bad actors respectively. Gender is indicated by (m) for male and (f) for female.

Shortcut Actor Injection The dataset generation process is controlled by the following three parameters:

1. Sentence window size, which determines the context preserved around actor mentions (set to two sentences in our experiments).

2. Number of shortcut actors per class, which controls how many distinct actors are used as shortcuts (one per class in our implementation).

3. Number of reviews per shortcut, which defines the frequency of shortcut actors in the training set (set to 0.01, which are 24 reviews).

[8] https://pypi.org/project/gender-guesser/

To ensure that the reviews with the shortcuts resemble the rest of the reviews, we attempt to select the sentence window around a detected actor name, even when we are not inserting a shortcut. When no actor name is selected in a review, we select the window at random.

Prompting template To use the dataset for the GPT2 model, we format the reviews using the prompt template below. Although we also fine-tune the model, we add the multiple choice labels to the prompt to better leverage the pretrained capabilities and for clarity.

```
"Classify the sentiment of the movie review:
Review: """{review}"""

LABEL OPTIONS: A: negative  B: positive
LABEL:"
```

A.3 Feature Attribution Method

For our LIME implementation we follow Ribeiro et al. (2016). The kernel function that measures the proximity between the original instance and its perturbations uses an exponential kernel with a kernel width of 25 and cosine distance as the distance measure. We take 1000 perturbations per review, which is relatively extensive given that the review consists of only two sentences.

Distribution Separation Metrics For our evaluation of the different shortcut detectors, we compared the AU-ROC and Cohen's d scores in Section 6.2. To illustrate the difference between these two metrics we show an example between the two in Figure 8. As shown in the figure, although the AU-ROC score might be very high between two distributions, the gap between them might be very small, making the final shortcut detection accuracy very sensitive to the right threshold.

B Appendix - Additional Results

B.1 Accuracy on ActorCorr per trained model

Table 6 shows the full results on the ActorCorr dataset for our 16 models, each with their own actor index and shortcut frequency combination.

B.2 Qualitative Analysis

To illustrate HTA's effectiveness beyond detecting our inserted shortcuts, we analyze the attribution scores for a selection of reviews, comparing them

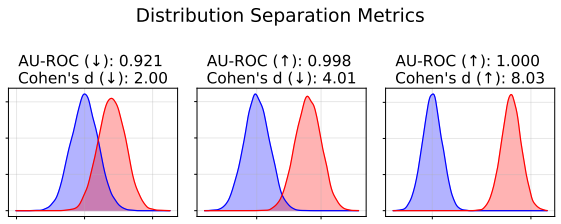


Figure 8: Distribution separation metrics for shortcut detectors. Arrows indicate relative high and low values

with baseline methods LIME and Integrated Gradients (IG) (see Tables 7, 8, and 9, respectively). We first present key observations from these samples, followed by a systematic analysis of test reviews without inserted shortcuts.

The examples show that HTA identifies both meaningful sentiment indicators (such as "good" and "bless" in "God bless") and known shortcuts like "4/10" (which are hardly important according to LIME and IG). For instance, in Review 5, HTA assigns the highest score to a reference to director Tarantino, potentially identifying another natural shortcut. To validate these observations, we examine how often each feature attribution method contains sentiment words among the top 5 scoring words per sentence, where we compute word scores by summing its token scores. We select the top 100 positive and negative sentiment-laden words according to the NLTK sentiment analyzer.[9] Table 5 shows that HTA matches LIME's accuracy in retrieving these sentiment words.

HTA differs from other feature attribution methods by identifying points in the input sequence where the model provides an intermediate decision, rather than providing general token importance. This behavior is visible from how it assigns the scores to the reviews. For instance, in Review 3 the rating shortcut "4/10" is detected by HTA by assigning a high score to the token "10", since the rating's effect only becomes clear after both numbers are observed. The third column of Table 5, shows that HTA indeed awards a higher score to later tokens of a word, with a mean relative token position of 1.69, compared to the mean relative token position of 1.60 and 1.51 for LIME and IG.

From the samples we also notice that HTA assigns a high score to far fewer tokens, giving a low score to most. We validate this observation by analyzing the average entropy of the normalized

[9] https://www.nltk.org/_modules/nltk/sentiment/vader.html

Method	Sentiment Words	MTW top idx	Entropy
HTA	29	1.692	3.467
LIME	29	1.600	4.509
IG	16	1.514	5.260

Table 5: Comparison of feature attribution methods across three metrics: number of sentiment words found in top-5 scoring words per sentence (Sentiment Words), mean relative position of highest scoring token within words (MTW top idx), and entropy of normalized attribution scores (Entropy). Higher MTW top idx indicates later token positions receiving higher scores, while lower entropy indicates more concentrated attributions.

score distribution across the dataset. A high entropy distribution indicates similar scores across tokens, while low entropy suggests more pronounced peaks. Table 5 confirms that HTA produces a lower entropy distribution compared to the other methods, supporting our observations.

Thus our analysis demonstrates three key characteristics of HTA beyond shortcut detection. Firstly, it successfully identifies semantically relevant input elements. Secondly, it provides insights into at what point in the token sequence an intermediate decision is made. Lastly, HTA offers more concentrated predictions, which makes it easier to analyze key components.

Shortcut Frequency	Actor index	neg clean noname	neg clean name	pos clean name	neg bad	pos good	pos clean noname	neg Good
0.01	0	85.58	76.94	79.10	80.31	78.44	78.37	78.21
0.01	1	89.44	83.01	71.02	86.36	69.71	69.38	85.14
0.01	2	87.26	77.56	79.06	74.28	80.21	76.42	76.82
0.01	3	76.63	64.56	88.85	67.30	91.68	85.16	59.03
0.03	0	79.13	68.76	84.67	71.03	84.72	85.87	69.46
0.03	1	84.40	74.88	82.18	76.20	82.78	78.33	74.07
0.03	2	87.18	76.49	80.30	78.30	80.16	76.61	77.00
0.03	3	86.46	79.38	76.66	80.30	83.84	75.12	72.17
0.10	0	80.85	69.58	84.09	95.33	92.64	81.55	53.72
0.10	1	85.78	77.60	78.15	76.98	79.17	76.52	76.79
0.10	2	88.54	79.37	76.31	79.83	76.90	74.19	79.25
0.10	3	90.71	86.67	66.93	91.50	82.29	67.28	71.77
0.30	0	88.70	79.96	75.27	99.40	91.32	74.51	55.89
0.30	1	77.14	66.97	87.70	83.56	99.55	85.06	15.67
0.30	2	83.01	72.53	82.53	88.67	97.74	81.09	31.57
0.30	3	72.55	60.16	90.87	78.03	98.49	89.52	30.57
1.00	0	88.93	83.11	73.25	99.86	99.60	73.87	1.28
1.00	1	83.68	75.10	80.26	99.15	99.67	80.10	7.32
1.00	2	82.92	71.79	82.69	98.80	99.70	80.29	1.48
1.00	3	83.75	77.26	75.81	99.67	99.38	77.42	4.17

Table 6: Test accuracy per data category for all our 16 trained models. Actor index refers to the used actor name as stated in Table 4. Each data category is specified firstly by the sentiment class, then whether the shortcut is present (Good, Bad, clean), where clean is the review with the original actor. Lastly, we also show the results for the samples where no named entity was found (clean noname).

Nr.	FA results - HTA
1	One has to wonder, is this what Blood Freak would have been like if Grinter hadn't co-directed with Steve Hawkes? If so, then God bless Steve Hawkes." Top Token: ' bless' (0.179)
2	I had high hopes for this film, even though I had not read the book. Richard Gere and Diane Lane together--should be good already." Top Token: ' good' (0.286)
3	Charlene & Gillian (from Twins) have never been able to act well and annoy you to pieces and "the friendly but wussy vampire" role was unfortunately given to Edison Chen who is a talentless pretty boy. Rating: 4/10 -- Top Token: '10' (0.869)
4	The blame of this terrible flick lies with the director, Martin Campbell. After viewing a few of his credits in later years, this must have been one of his first directorial gigs." Top Token: ' director' (0.578)
5	But I guess if you're gonna take a lead role in the Ghoulies films, Scorsese and Tarantino will lose interest. Also present is his idiot sidekick Bobby Di Cocco, who despite having a very small resemblance to Al Pacino (very small), retains none of his acting ability... A complete idiot who's just awkward to watch." Top Token: 'ino' (0.328)

Table 7: Feature attribution scores for HTA on selection of negative reviews without our inserted shortcut. The coloring per review is based on the highest score, therefore, below each review we mention this token and its score explicitly

Nr.	FA results - LIME
1	One has to wonder, is this what Blood Freak would have been like if Grinter hadn't co-directed with Steve Hawkes? If so, then God bless Steve Hawkes." Top Token: ' then' (0.169)
2	I had high hopes for this film, even though I had not read the book. Richard Gere and Diane Lane together--should be good already." Top Token: ' hopes' (0.332)
3	Charlene & Gillian (from Twins) have never been able to act well and annoy you to pieces and "the friendly but wussy vampire" role was unfortunately given to Edison Chen who is a talentless pretty boy. Rating: 4/10 -- Top Token: ' vampire' (0.185)
4	The blame of this terrible flick lies with the director, Martin Campbell. After viewing a few of his credits in later years, this must have been one of his first directorial gigs." Top Token: ' terrible' (0.206)
5	But I guess if you're gonna take a lead role in the Ghoulies films, Scorsese and Tarantino will lose interest. Also present is his idiot sidekick Bobby Di Cocco, who despite having a very small resemblance to Al Pacino (very small), retains none of his acting ability... A complete idiot who's just awkward to watch." Top Token: ' idiot' (0.129)

Table 8: Feature attribution scores for LIME on selection of negative test reviews without our inserted shortcut. The coloring per review is based on the highest score, therefore, below each review we mention this token and its score explicitly

Nr.	FA results - Integrated Gradients (IG)
1	One has to wonder, is this what Blood Freak would have been like if Grinter hadn't co-directed with Steve Hawkes? If so, then God bless Steve Hawkes." Top Token: 'One' (4.842)
2	I had high hopes for this film, even though I had not read the book. Richard Gere and Diane Lane together--should be good already." Top Token: 'ere' (2.256)
3	Charlene & Gillian (from Twins) have never been able to act well and annoy you to pieces and "the friendly but wussy vampire" role was unfortunately given to Edison Chen who is a talentless pretty boy. Rating: 4/10 -- Top Token: ' annoy' (2.397)
4	The blame of this terrible flick lies with the director, Martin Campbell. After viewing a few of his credits in later years, this must have been one of his first directorial gigs." Top Token: ' one' (1.941)
5	But I guess if you're gonna take a lead role in the Ghoulies films, Scorsese and Tarantino will lose interest. Also present is his idiot sidekick Bobby Di Cocco, who despite having a very small resemblance to Al Pacino (very small), retains none of his acting ability... A complete idiot who's just awkward to watch." Top Token: ' idiot' (2.041)

Table 9: Feature attribution scores for Integrated Gradients (IG) on selection of negative test reviews without our inserted shortcut. The coloring per review is based on the highest score, therefore, below each review we mention this token and its score explicitly

B.3 Patching Additional: via keys

In Section 5.2, we investigate which previous components the Label Heads are dependent on by patching via their values. Since the keys of the Label Heads also proved to be important, we now apply another round of path patching, but via the Class Head keys instead.

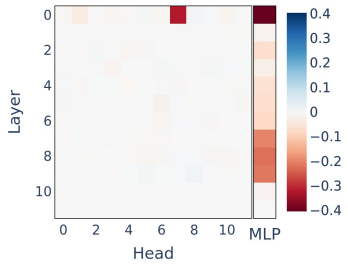


Figure 9: Patching Via Keys: positive with Bad actor

Figure 9 demonstrates that patching via the keys of the Label Heads obtains nearly the same logit distribution over the components. Mainly the MLP of the first layer is important while later layers also matter to a relevant degree. Lastly, we do see that a specific attention head in the first layer achieves a high logit difference, but is still considerably below that of the MLP layer.

B.4 Patching Additional: imbalance frequency

In Section 5.2, we demonstrated the patching results for one of our trained models. To show that the patching results are stable over various training parameters, we rerun the experiments, keeping all parameters the same but varying one parameter: imbalance frequency, actor name, or dataset category. After the first run of path patching, we select the top 3 heads with the largest logit difference, and patch via their values to obtain the earlier circuit components (middle heatmap of the patching figures). The results demonstrate the same general findings of Section 5.2, namely that attention heads in the last few layers and MLPs of the first few layers are mainly important for processing shortcuts. Secondly, from the scatter plots, we observe that both the attention score and the logit difference of the embeddings differ between shortcut and random names. Below we describe the figures and more specific findings.

In Figures 10, 11, 12, 13, 14 we evaluate the results using the imbalanced frequencies $[0.001, 0.003, 0.001, 0.0003, 0.0001]$. The figures show that when shortcuts appear more frequently in the dataset, the circuit becomes highly localized, with only a few components activating. Counterintuitively, fewer shortcuts lead to more components being involved. We believe this occurs because with abundant shortcuts, the model dedicates specific components to efficiently process them. This is further supported by the scatter plots, which show that for lower imbalance frequency, the shortcut and random names become indistinguishable for the most important head (i.e. its attention pattern and activation logit difference).

Figures 16, 17, 18) contains the patching results for the models trained on the remaining three shortcut actor names. Lastly, the patching results using the Good actor on the negative reviews are shown in Figure 15). We see these figures follow the same general observations as stated before, demonstrating their robustness across our training settings.

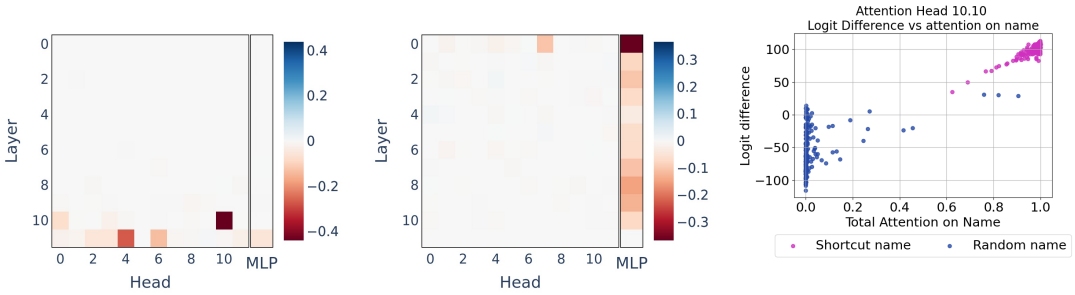


Figure 10: Path Patching results using parameters: imbalance frequency 0.01, actor index 0, and data category: positive with Bad actor. The middle figure shows patching via the values of heads 10.10, 11.4, and 11.6.

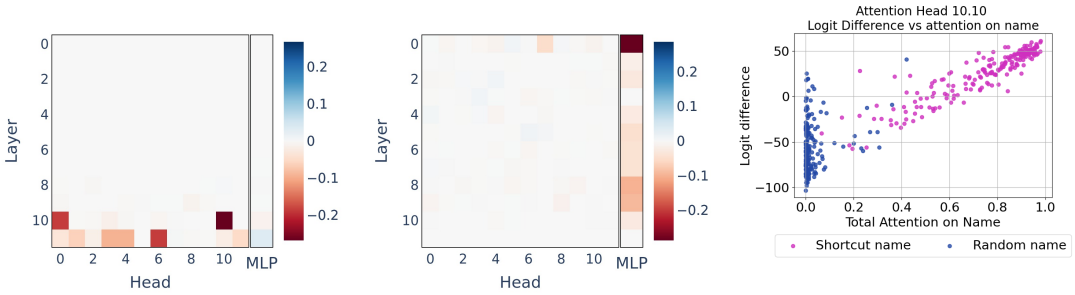


Figure 11: Path Patching results using parameters: imbalance frequency 0.003, actor index 0, and data category: positive with Bad actor. The middle figure shows patching via the values of heads 10.10, 10.0, and 11.6.

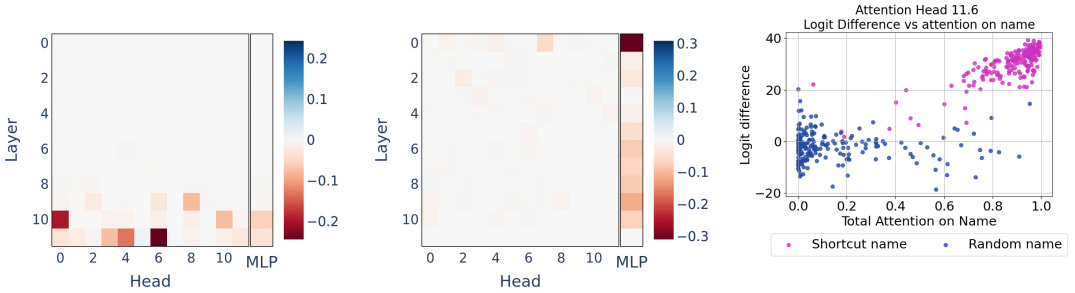


Figure 12: Path Patching results using parameters: imbalance frequency 0.001, actor index 0, and data category: positive with Bad actor. The middle figure shows patching via the values of heads 11.6, 10.0, and 11.4.

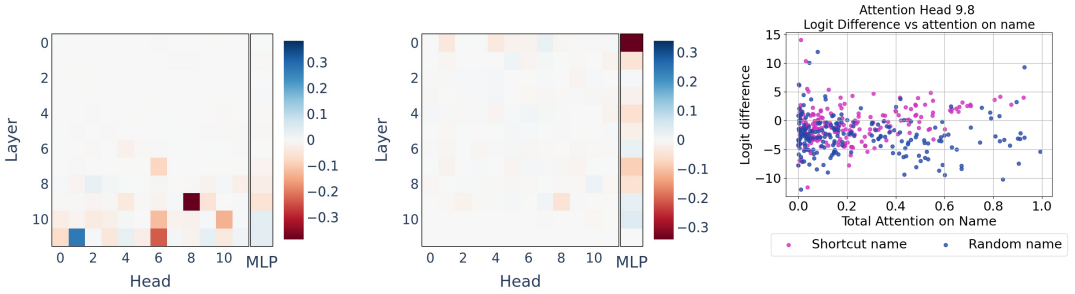


Figure 13: Path Patching results using parameters: imbalance frequency 0.0003, actor index 0, and data category: positive with Bad actor. The middle figure shows patching via the values of heads 9.9, 11.6, and 10.10

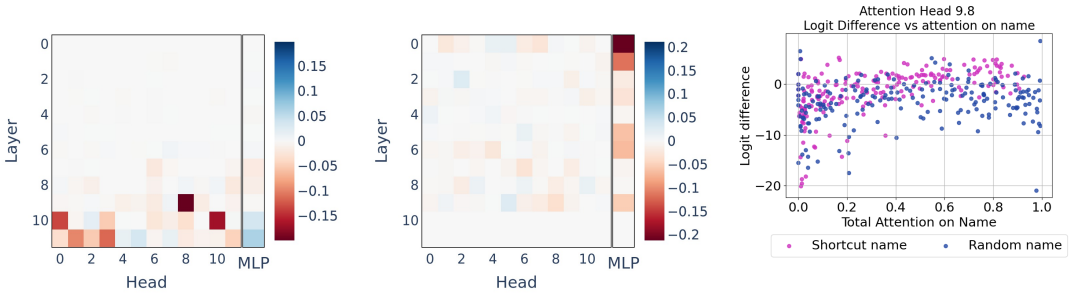


Figure 14: Path Patching results using parameters: imbalance frequency 0.0001, actor index 0, and data category: positive with Bad actor. The middle figure shows patching via the values of heads 9.8, 10.10, and 10.0.

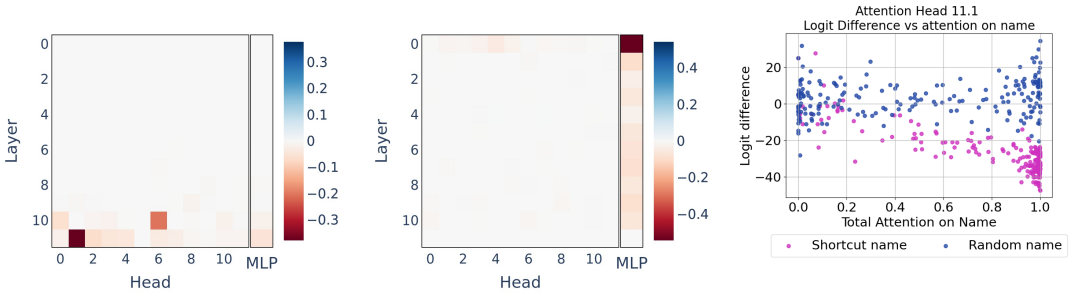


Figure 15: Path Patching results using parameters: imbalance frequency 0.003, actor index 0, and data category: negative with Good actor. The middle figure shows patching via the values of heads 11.1, 10.6, and 11.2.

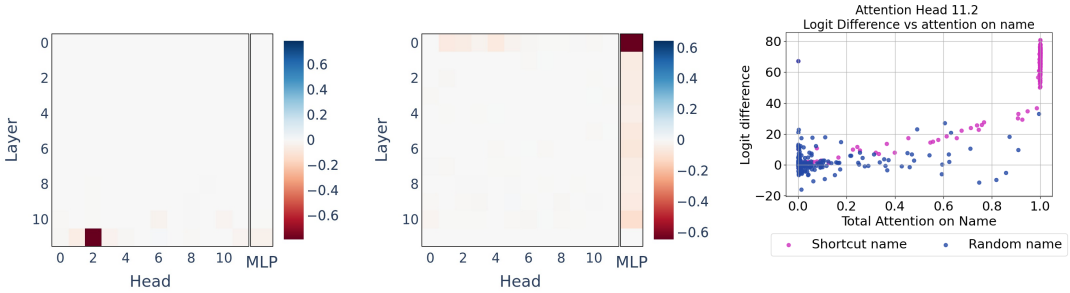


Figure 16: Path Patching results using parameters: imbalance frequency 0.003, actor index 1, and data category: positive with Bad actor. The middle figure shows patching via the values of heads 11.2, 11.1, and 10.6.

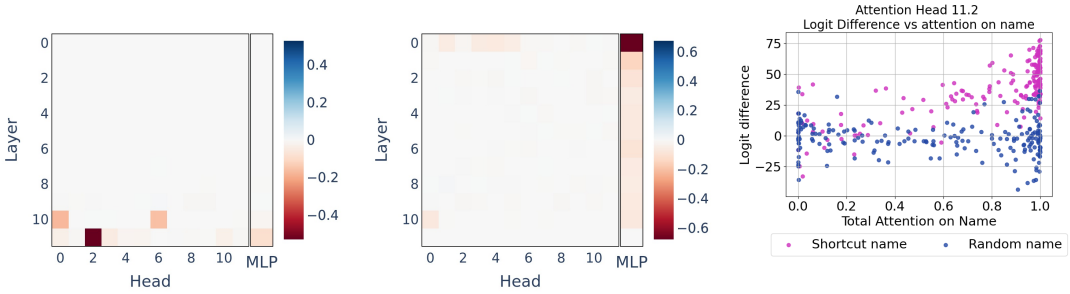


Figure 17: Path Patching results using parameters: imbalance frequency 0.003, actor index 2, and data category: positive with Bad actor. The middle figure shows patching via the values of heads 11.2, 10.0, and 10.6.

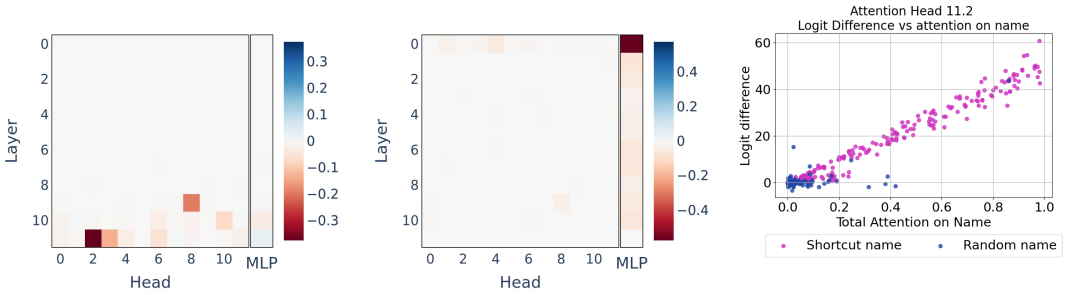


Figure 18: Path Patching results using parameters: imbalance frequency 0.003, actor index 3, and data category: positive with Bad actor. The middle figure shows patching via the values of heads 11.2, 9.8, and 11.3.