Patchscopes: A Unifying Framework for Inspecting Hidden Representations of Language Models

Asma Ghandeharioun [*1] **Avi Caciularu** [*1] **Adam Pearce** [1] **Lucas Dixon** [1] **Mor Geva** [1 2]

Abstract

Understanding the internal representations of large language models (LLMs) can help explain models' behavior and verify their alignment with human values. Given the capabilities of LLMs in generating human-understandable text, we propose leveraging the model itself to explain its internal representations in natural language. We introduce a framework called Patchscopes and show how it can be used to answer a wide range of questions about an LLM's computation. We show that many prior interpretability methods based on projecting representations into the vocabulary space and intervening on the LLM computation can be viewed as instances of this framework. Moreover, several of their shortcomings such as failure in inspecting early layers or lack of expressivity can be mitigated by Patchscopes. Beyond unifying prior inspection techniques, Patchscopes also opens up *new* possibilities such as using a more capable model to explain the representations of a smaller model, and multihop reasoning error correction[1].

1. Introduction

The question of what information is captured in the hidden representations of large language models (LLMs) is of key importance in control and understanding of modern generative AI, and has drawn substantial attention recently (Casper et al., 2022; Madsen et al., 2022; Patel & Pavlick, 2021; Nanda et al., 2023). To tackle this question, prior work has introduced a diverse array of interpretability methods, which largely rely on three prominent approaches: training linear classifiers, called probes, on top of hidden representations (Belinkov & Glass, 2019; Belinkov, 2022; Alain & Bengio, 2017), projecting representations to the model's vocabulary space (nostalgebraist, 2020; Din et al., 2023; Belrose et al., 2023), and intervening on the computation to identify if a representation is critical for certain predictions (Meng et al., 2022a; Wallat et al., 2020; Wang et al., 2022; Conmy et al., 2023; Geva et al., 2023).

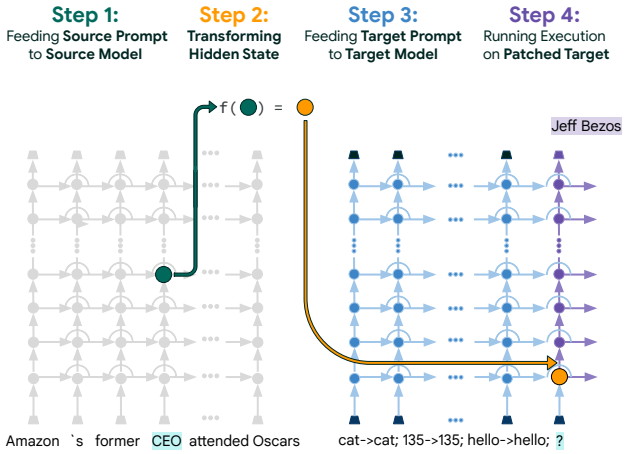


Figure 1. Illustration of our framework, showing a Patchscope for decoding the hidden representation of *"CEO"* in the source prompt (left). We use a target prompt (right) comprised of few-shot demonstrations of string repetitions to encourage the LLM to explain its internal representation. **Step 1**: Run the forward computation on the source prompt in the source model. **Step 2**: Optionally transform the hidden state at the source layer. **Step 3**: Run the forward computation on the target prompt up to the target layer in the target model. **Step 4**: Patch the target representation of the token *"?"* at the target layer with the transformed representation (from step 2), then continue the forward computation from that layer onward. The modularity of Patchscopes allows designing a variety of methods by configuring the target prompt, model and transformation.

Despite the wide success of these methods, they each exhibit practical shortcomings. First, probing relies on supervised training for pre-defined classes, which is hard to scale when there is a large number of classes or when all the categories are not known a priori. Second, the accuracy of vocabulary

[*]Equal contribution [1]Google Research [2]Tel Aviv University. Correspondence to: Asma Ghandeharioun <aghandeharioun@google.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[1]Code is publicly available at https://pair-code.github.io/interpretability/patchscopes.

Table 1. Many prior inspection methods with various objectives can be viewed as Patchscopes. The rows highlighted in green show Patchscope configurations that overcome several limitations of prior methods through more expressive inspection that is training-data free and is more robust across layers.

Inspection Objective		Expressive	Training Data Free	Robust Across Layers
Inspecting Output Distribution	Few-Shot Token Identity Patchscope (§4.1)	✔✔	✔	✔✔
	Logit Lens (nostalgebraist, 2020), Embedding Space Analysis (Dar et al., 2023)	✔	✔	✖
	Tuned Lens (Belrose et al., 2023)	✔	For learning mappings	✔
	Future Lens (Pal et al., 2023)	✔	For learning mappings	✔✔
Feature Extraction	Zero-Shot Feat. Ext. Patchscope (§4.2)	✔✔	✔	✔✔
	LRE Attribute Lens (Hernandez et al., 2023b)	✔	For linear relation approx.	✔✔
	Probing (e.g., Belinkov & Glass, 2019; Belinkov, 2022; Alain & Bengio, 2017; Wang et al., 2023)	✖	For training probe	✔
Entity Resolution	Entity Description Patchscope (§4.3)	✔✔	✔	✔✔
	X-Model Entity Desc. Patchscope (§4.4)	✔✔✔	For learning mappings	✔✔
	Causal Tracing (Meng et al., 2022a)	✖	✔	✔✔
	Attention Knockout (Wang et al., 2022; Conmy et al., 2023; Geva et al., 2023)	✖	✔	✔✔
Inspection Application	Early Exiting, e.g., Linear Shortcuts (Din et al., 2023)	✔	For learning mappings	✔
	Caption Generation, e.g., Linear Mapping (Merullo et al., 2022)	✔	For learning mappings	✔

projections substantially decreases in early layers and the outputs are often hard to interpret. Last, all the above methods are not as expressive as one might like: they provide class probabilities or most likely tokens, as opposed to a high-quality explanation in natural language.

In this work, we argue that the advanced capabilities of LLMs in generating human-like text can be leveraged for "translating" the information in their representations for humans. We introduce a modular framework, called Patchscopes (§3), that can be configured to query various kinds of information from LLM representations. Patchscopes decode specific information from a representation within an LLM by "patching" it into the inference pass on a different prompt that has been designed to encourage the extraction of that information.[2] Such configuration (a Patchscope) can be viewed as an inspection tool geared towards a particular objective, as illustrated in Fig. 1.

We show that many existing methods, including those that rely on vocabulary projections and computation interventions, can be cast as Patchscopes. Moreover, new configurations of our framework introduce more effective tools in addressing the same questions, while mitigating several limitations of prior approaches. Also, Patchscopes enables addressing underexplored questions, such as fine-grained analysis of the input contextualization process and the extent to which a more expressive model can be used to inspect hidden representations of a smaller model.

[2]While patching (or "activation patching") by itself is not a new technique, to the best of our knowledge, we are the first to propose using it for decoding information from hidden representations in a configurable and expressive manner. See more details in §3.2.

We conduct a series of experiments to evaluate the benefits and opportunities introduced by Patchscopes, focusing on auto-regressive LLMs. First, we consider the problem of estimating the model's next-token prediction from its intermediate representations (see §4.1). Across multiple LLMs, we show that using a few-shot token identity prompt, a prompt in the form of "$\texttt{tok}_1 \rightarrow \texttt{tok}_1$; $\texttt{tok}_2 \rightarrow \texttt{tok}_2; \ldots; \texttt{tok}_k$" where $\texttt{tok}_i$ refers to a random token, leads to substantial gains over vocabulary projection methods. Next, we evaluate how well Patchscopes can decode specific attributes of an entity from its LLM representations, when these are detached from the original context (see §4.2). We observe that, despite using no training data, Patchscopes significantly outperforms probing in six out of twelve commonsense and factual reasoning tasks, and works comparably well in all but one of the remaining six.

Beyond output estimation and attribute decoding, Patchscopes can address questions that are hard to answer with existing methods. In §4.3, we apply Patchscopes to study how LLMs contextualize input entity names in early layers, where vocabulary projections mostly fail and other methods, at best, provide only a binary signal of whether the entity has been resolved (Youssef et al., 2023; Tenney et al., 2019). With a new Patchscope, we are able to *verbalize* the gradual entity resolution process. For example, we show that, as the model processes the final token of *"Alexander the Great"* throughout the layers, it reflects different entities starting from *"Great Britain"*, to *"the Great Depression"*, to finally resolving *"Alexander the Great"*. Then, in §4.4 we show how one can further improve Patchscope expressivity by using a stronger target model, e.g., Vicuna 13B instead of Vicuna 7B.

Lastly, we showcase the utility of Patchscopes for fixing latent multi-hop reasoning errors, particularly when the model is capable of conducting each reasoning step correctly, but fails when they need to be composed in-context (§5). Building on top of the data provided by Hernandez et al. (2023b), we introduce a more complex task that requires two steps of factual reasoning. Patchscope achieves 50% accuracy on this task, outperforming chain-of-thought (Wei et al., 2022) (35.71%) and vanilla generations (19.57%).

To conclude, our work makes the following contributions: We propose Patchscopes, a general modular framework for decoding information from the hidden representations in LLMs. We show that prominent interpretability methods can be viewed as instances of Patchscopes, and new configurations result in more expressive, robust across layers, and training-data free alternatives that mitigate their shortcomings. In addition, novel configurations introduce unexplored possibilities of stronger inspection techniques, as well as practical benefits, such as correcting multi-hop reasoning errors.

2. Related Work

Activation patching is a causal intervention, commonly used as a tool for studying if certain activations play a key role in a model's computation (Geiger et al., 2021; Vig et al., 2020). Patching has been used largely for localizing specific information to specific layers and token positions (Goldowsky-Dill et al., 2023; Meng et al., 2022a;b; Stolfo et al., 2023; Merullo et al., 2023), and for finding information propagation paths in the computation (Wang et al., 2022; Geva et al., 2023; Hendel et al., 2023; Hanna et al., 2023; Lieberum et al., 2023). Prior works have also used specific forms of cross-model patching called stitching, in non-transformer architectures, mostly to analyze representational similarity (e.g., Bansal et al., 2021; Csiszárik et al., 2021; Lenc & Vedaldi, 2015). Despite certain limitations (Hase et al., 2023; Zhang & Nanda, 2023), patching remains a principal tool for mechanistic interpretability (Conmy et al., 2023).

Given promising results from emerging interpretability efforts that employ LLMs to generate human-like text for inspection (e.g., Mousi et al., 2023; Slobodkin et al., 2023; Bills et al., 2023), we argue that using patching only for localization purposes is myopic, and propose to use it for "translating" LLM representations into natural language. Very recently, patching has been used to study new problem setups (e.g., Pal et al., 2023; Hernandez et al., 2023b), all of which can be seen as different configurations of our proposed framework (see §3.2).

Among the growing research efforts in inspecting hidden representations of neural networks, probing classifiers are perhaps the most common (e.g., Alain & Bengio, 2017; Belinkov & Glass, 2019; Belinkov, 2022; Wang et al., 2023), and methods using projections into the vocabulary space or their extensions to other domains are another key category (e.g., Merullo et al., 2023; Geva et al., 2022b; nostalgebraist, 2020; Belrose et al., 2023; Dar et al., 2023; Din et al., 2023; Langedijk et al., 2023; Vilas et al., 2023). While various other latent inspection methods exist (e.g., Zhou et al., 2018; Strobelt et al., 2017; Ghandeharioun et al., 2021; Kim et al., 2018), the above are the most relevant to this work.

3. Patchscopes

In this section, we introduce Patchscopes and show how it extends prior interpretability methods with new capabilities. While not limited to particular LLM architectures, this work focuses on auto-regressive transformer-based LLMs.

3.1. Framework Description

The key idea in Patchscopes is to leverage the advanced capabilities of LLMs to generate human-like text for "translating" the information encoded in their own hidden representations. Concretely, given a hidden representation obtained from an LLM inference pass, we propose to decode specific information from it by *patching* it into a different inference pass (of the same or a different LLM) that encourages the translation of that specific information.

Notably, the rest of the forward computation after patching can augment the representation with additional information, hence, this approach does not guarantee that the patched representation itself stores *all* that information. However, dispatching the representation from its original context (the source prompt) stops contextualization and guarantees that no further information from the source prompt is incorporated in the post-patching computation. Thus, our framework reveals if specific information *can be decoded from the patched representation via the post-patching computation*.

Given an input sequence of n tokens $S = \langle s_1, ..., s_n \rangle$ and a model $\mathcal{M}$ with L layers, $\boldsymbol{h}_i^\ell$ denotes the hidden representation obtained at layer $\ell \in [1, \ldots, L]$ and position $i \in [1, \ldots, n]$, when running $\mathcal{M}$ on S. To inspect $\boldsymbol{h}_i^\ell$, we consider a separate inference pass of a model $\mathcal{M}^*$ with L^* layers on a target sequence $T = \langle t_1, \ldots, t_m \rangle$ of m tokens. Specifically, we choose a hidden representation $\bar{\boldsymbol{h}}_{i^*}^{\ell^*}$ at layer $\ell^* \in [1, \ldots, L^*]$ and position $i^* \in [1, \ldots, m]$ in the execution of $\mathcal{M}^*$ on T. Moreover, we define a mapping function $f(\boldsymbol{h}; \boldsymbol{\theta}) : \mathbb{R}^d \mapsto \mathbb{R}^{d^*}$ parameterized by $\boldsymbol{\theta}$ that operates on hidden representations of $\mathcal{M}$, where d and d^* denote the hidden dimension of representations in $\mathcal{M}$ and $\mathcal{M}^*$, respectively. This function can be the identity function, a linear or affine function learned on task-specific pairs of representations, or even more complex functions that incorporate other sources of data. The *patching* operation refers to dynamically replacing the representation $\bar{\boldsymbol{h}}_{i^*}^{\ell^*}$ during the inference of $\mathcal{M}^*$ on T with $f(\boldsymbol{h}_i^\ell)$. Namely, by applying $\bar{\boldsymbol{h}}_{i^*}^{\ell^*} \leftarrow f(\boldsymbol{h}_i^\ell)$, we intervene on the generation process and modify the computation after layer ℓ^*.

Overall, a Patchscope intervention applied to a representation determined by $(S, i, \mathcal{M}, \ell)$, is defined by a quintuplet $(T, i^*, f, \mathcal{M}^*, \ell^*)$ of a target prompt T, a target position i^* in this prompt, a mapping function f, a target model $\mathcal{M}^*$, and a target layer ℓ^* of this model. It is possible that $\mathcal{M}$ and $\mathcal{M}^*$ are the same model, S and T are the same prompt, and f is the identity function $\mathbb{I}$ (i.e., $\mathbb{I}(\boldsymbol{h}) = \boldsymbol{h}$). Next, we show how this formulation covers prior interpretability methods and further extends them with new capabilities.

3.2. Patchscopes Encompasses Prior Methods

Many recent methods inspect LLM representations by projecting them to the output vocabulary space (nostalgebraist, 2020; Din et al., 2023; Belrose et al., 2023). Formally, an estimation of the output distribution is obtained from the representation $\boldsymbol{h}_i^\ell$ at position i and layer ℓ by:

$$\boldsymbol{p}_i^\ell = \mathrm{softmax}(W_U f(\boldsymbol{h}_i^\ell)) \in \mathbb{R}^{|V|},$$

where $W_U \in \mathbb{R}^{|V| \times d}$ is the model's unembedding matrix and f is a simple mapping function, such as the identity function or an affine mapping. We note that the operation applied to $f(\boldsymbol{h}_i^\ell)$ is the same computation applied by the model to the last-layer representation for obtaining the next-token prediction. Therefore, prior methods that inspect representations in the vocabulary space can be viewed as a class of Patchscopes that maps representations from any source layer ℓ to the last target layer L^*. Differences between these methods lie in the choice of f; logit lens (nostalgebraist, 2020; Dar et al., 2023) applies the identity function, linear shortcuts (Din et al., 2023) uses a linear mapping function, and tuned lens (Belrose et al., 2023) trains an affine mapping. Recently, Hernandez et al. (2023b) introduced LRE Attribute Lens that builds f based on a relation linearity assumption, and showed its effectiveness in attribute extraction.

This class of methods has proven to be effective for different applications, for example, in improving inference efficiency via early exiting (Din et al., 2023). While the majority of methods and applications in this category use a single model ($\mathcal{M}^* = \mathcal{M}$), Merullo et al. (2022) had demonstrated successful caption generation with a generative image model as $\mathcal{M}$ and a language model as $\mathcal{M}^*$.

Another category of inspection methods intervene on the LLM computation. Contemporary to our work, Pal et al. (2023) have investigated whether it is possible to anticipate multiple generated tokens ahead from a given hidden representation, rather than estimating just the next-token prediction. Their method (Future Lens) uses a target prompt different from the original prompt (i.e., $T \neq S$) and is designed to decode subsequent tokens from information encoded in a hidden representation $\boldsymbol{h}_i^\ell$. Example target prompts are *"The multi-tokens present here are "* and *"Hello! Could you please tell me more about "*. Future Lens can be cast as another Patchscope with $\mathcal{M}^* = \mathcal{M}$ and $\ell^* = \ell$.

More broadly, Patchscopes also cover recent mechanistic interpretability methods that analyze internal processes in LLMs with inference computation interventions. Specifically, causal tracing (Meng et al., 2022a) uses a source prompt augmented with Gaussian noise as the target prompt. Other previous methods have intervened on one or more target layers during inference by patching zero vectors to the computation (Wang et al., 2022; Conmy et al., 2023; Geva et al., 2023), namely, setting $f(\boldsymbol{h}) = \mathbf{0}$. For a configuration summary of how these interpretability methods can be cast as Patchscope instances, see §A, Tab. 4.

3.3. Patchscopes Enables Novel Inspection Methods

Prior work has utilized specific patching configurations for interpretability, largely patching the same model while using the same prompt (i.e., $\mathcal{M}^* = \mathcal{M}$, $T = S$). The framing of Patchscopes introduces a wide range of unexplored setups potentially unlocking new inspection capabilities.

Specifically, we observe that modifying the target prompt enables an expressive decoding of a wide range of features, detached from the source prompt computation. For instance, we can use the prompt *"The capital of X is"* to check if the capital city of a given country is extractable from its (last token) hidden representation at a specific layer. Similarly, a prompt like *"Tell me facts about X"* can be leveraged to assess whether the model has resolved the entity name in a specific layer. Contrary to probing, this approach is not restricted by the number of classes of the chosen feature.

Moreover, when the inspected model is not expressive enough to answer certain queries, patching representations into a more capable model could be useful (Hernandez et al., 2022; Singh et al., 2023; Schwettmann et al., 2023).

4. Experiments

In this section, we evaluate our framework on decoding next-token predictions (§4.1), extracting attributes (§4.2), analyzing the contextualization of entity names (§4.3), and leveraging stronger models for inspection via cross-model patching (§4.4). See a summary in Tab. 1.

4.1. Decoding of Next-Token Predictions

As introduced in §3.2, let $\boldsymbol{p}^L$ be the output probability distribution for some input, obtained by multiplying the final-layer last-position hidden representation $\boldsymbol{h}^L$ by the unembedding matrix $W_U \in \mathbb{R}^{|V| \times d}$. We wish to estimate $\boldsymbol{p}^L$ from intermediate representations $\mathbf{h}^\ell$ s.t. $\ell < L$. Particularly, we ask how early in the computation the model has concluded its final prediction from the given context. In our experiments, we consider multiple LLMs – LLaMA2 (13B) (Touvron et al., 2023b), Vicuna (13B) (Chiang et al., 2023), GPT-J (6B) (Wang & Komatsuzaki, 2021), and Pythia (12B) (Biderman et al., 2023) (see more details in §B.1).

Methods We compare vocabulary projection methods (§3.2) with a new Patchscope. Each method yields an estimated output probability $\tilde{\boldsymbol{p}}^\ell$ by patching an intermediate representation $\boldsymbol{h}^\ell$ to the model's final layer. Here, we focus on the common setting where $\mathcal{M} = \mathcal{M}^*$, and discuss extensions to $\mathcal{M} \neq \mathcal{M}^*$ in §4.4.

- **Logit Lens:** Following prior work (nostalgebraist, 2020; Geva et al., 2022a), we define f as the identity function, meaning no change is applied to the patched representation. That is, $f(\boldsymbol{h}) := \mathbb{I}(\boldsymbol{h})$.
- **Tuned Lens:** Motivated by Belrose et al. (2023); Din et al. (2023), we employ an affine mapping function between representations at layer ℓ and the final layer

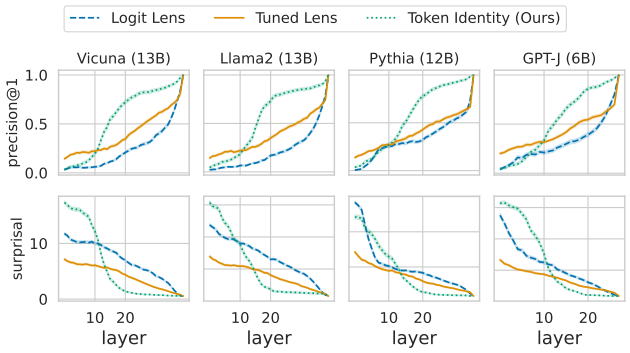


Figure 2. Precision@1 (↑ is better) and Surprisal (↓ is better) of next-token prediction estimation in multiple models. From layer 10 and upwards, the token identity method (ours) consistently outperforms the rest of the baselines across all the models.

L. Specifically, we feed the model examples from a training set $\mathcal{T}$ and for each example $s \in \mathcal{T}$ obtain a pair $(\boldsymbol{h}_s^\ell, \boldsymbol{h}_s^L)$ of hidden representations. Then, we fit a linear regression model to find the matrix $A^\ell \in \mathbb{R}^{d \times d}$ and the bias vector $\boldsymbol{b}^\ell \in \mathbb{R}^d$ that are numerical minimizers for $\sum_{s \in \mathcal{T}} ||A\boldsymbol{h}_s^\ell - \boldsymbol{h}_s^L + \boldsymbol{b}||^2$. We define f as: $f(\boldsymbol{h}^\ell) := A^\ell \boldsymbol{h}^\ell + \boldsymbol{b}^\ell$.

- **Token Identity** `Patchscope`: Unlike the previous methods, here we use a target prompt that is different from the source prompt ($T \neq S$) and is meant to encourage decoding the token identity of the hidden representation. Also, while the above methods skip the computation between layers l and L, here we modify it such that all the information from the source prompt computation is discarded, except for the patched representation. We craft a prompt with k demonstrations representing an identity-like function, formatted as "$\texttt{tok}_1 \rightarrow \texttt{tok}_1$ `;` $\texttt{tok}_2 \rightarrow \texttt{tok}_2$ `;` ... `;` $\texttt{tok}_k$". See §B.3 for further details and an experiment showing this method's robustness to different demonstrations. Note that this `Patchscope` does not require any training.

Evaluation Following Din et al. (2023), we use Pile evaluation set and the following metrics (see details in §B.2):

- **Precision@1** (↑ is better): The portion of examples for which the highest-probability token t in the estimated probability distribution matches the highest-probability token in the original output distribution. That is, if $\arg\max_t(\tilde{\boldsymbol{p}}_t^\ell) = \arg\max_t(\boldsymbol{p}_t^L)$.
- **Surprisal** (↓ is better): The minus log-probability of the highest-probability token in the predicted distribution $\tilde{\boldsymbol{p}}^\ell$ according to $\boldsymbol{p}^L$, i.e., $-\log \boldsymbol{p}_{\tilde{t}}^L$, where $\tilde{t} = \arg\max_t(\tilde{\boldsymbol{p}}_t^\ell)$.

Results Across all the models, from layer 10 and upwards, the token identity `Patchscope` consistently outperforms the other baselines, obtaining a gain of up to 98% in layers 18-22 (see Fig. 2). This demonstrates the utility of leveraging the model's decoding procedure for inspecting representations of different source prompts, and shows that *in most cases* hidden representations in early layers carry the prediction information regardless of their context.

In the first 10 layers, performance of all methods is worsened, with the token identity prompt performing on-par with logit lens, and tuned lens performing slightly better, which could be due to the additional training of its mappings. Low performance in these layers is expected, as it is where the input contextualization happens. In §4.3, we introduce a `Patchscope` geared towards unraveling this process.

4.2. Extraction of Specific Attributes

Classification probes are arguably the most commonly used method for checking if certain attributes are encoded in hidden representations (Belinkov, 2022; Belinkov & Glass, 2019). However, they need to be trained, and the range of attribute classes needs to be known a priori. Here we show that repurposing `Patchscopes` for attribute extraction overcomes these limitations. First, it does not require training. Second, it is not limited by a predefined set of labels, but rather benefits from an open vocabulary. In addition, by taking advantage of the model's nonlinearities, it can capture more complex relations compared to linear probes.

Experimental Setup Consider factual and commonsense knowledge represented as triplets (σ, ρ, ω) of a subject (e.g., "`United States`"), a relation (e.g., "`largest city of`"), and an object (e.g., "`New York City`"). We investigate to what extent the object ω can be extracted from the last token representation of the subject σ in an arbitrary input context. To this end, we conduct experiments on 8 commonsense and 25 factual knowledge tasks curated by Hernandez et al. (2023b). This dataset includes (σ, ρ, ω) triplets for different relations, along with prompt templates that verbalize them in natural language. We conduct experiments with GPT-J (6B) (Wang & Komatsuzaki, 2021), filtering the data to keep only the examples where ω appears in the the model's continuation of the prompt up to 20 tokens. The choice of 20 balances computation cost with accommodating the open-ended nature of Patchscopes, as the ground truth token does not necessarily appear in the next immediate token in a fluent response. For each example, we sample 5 utterances from the WikiText-103 dataset (Merity et al., 2016) that include σ and use them as S. Lastly, we keep tasks with at least 15 samples, which results in 5 commonsense and 7 factual tasks with a total of 1,453 datapoints. See details in §C.

Methods We compare our proposed `Patchscope` against linear probing (Köhn, 2015; Gupta et al., 2015).

Table 2. Feature extraction accuracy (mean±std). Comparing zero-shot feature extraction Patchscope to a logistic regression probe shows that despite using *no training data*, it has a significantly higher accuracy than baseline in 6 out of 12 tasks. We use pairwise t-test with Bonferroni correction for comparing the two methods. ** and * indicate $p < 1e{-}5$ and $p < 1e{-}4$, respectively.

	Task	Probe	Patchscope
Commonsense	Fruit inside color	37.4 ± 6.6	38.0 ± 18.7
	Fruit outside color	35.5 ± 3.1	$\mathbf{71.0 \pm 13.3}^{**}$
	Object superclass	$\mathbf{65.6 \pm 10.5}^{*}$	54.8 ± 11.3
	Substance phase	73.8 ± 3.7	$\mathbf{91.9 \pm 1.7}^{**}$
	Task done by tool	10.1 ± 3.2	$\mathbf{48.1 \pm 13.2}^{**}$
Factual	Company CEO	5.0 ± 2.6	$\mathbf{47.8 \pm 13.9}^{**}$
	Country currency	17.7 ± 2.2	$\mathbf{51.0 \pm 8.9}^{**}$
	Food from country	5.1 ± 3.7	$\mathbf{63.8 \pm 11.3}^{**}$
	Plays pos. in sport	75.9 ± 9.1	72.2 ± 7.2
	Plays pro sport	53.8 ± 10.3	46.3 ± 14.2
	Product by co.	58.9 ± 7.2	63.2 ± 10.7
	Star constellation	17.5 ± 5.3	18.4 ± 5.1

- **Zero-shot Feature Extraction** Patchscope: We craft T as a general verbalization of ρ followed by a placeholder for σ, such that $i^* = m$. For example, we use $T \leftarrow$ "The largest city in x" with "x" as a placeholder for the subject. To extract the object from the entity representation in S, we patch the representation of token "x" at layer ℓ^* with the representation of "States" from layer ℓ, and consider if the generated text includes ω. The remaining configurations of this Patchscope are $f \leftarrow \mathbb{I}, \mathcal{M}^* \leftarrow \mathcal{M}, i \leftarrow$ the last token of σ in S. We consider all combinations of $\ell \in [1, \ldots, L] \times \ell^* \in [1, \ldots, L^*]$. Later in this section, we discuss the role of ℓ pertaining to attribute extraction.

- **Logistic Regression Probe**: Let Ω represent the range of possible objects for a given relation. We use the set of unique values of ω in the training set as a proxy for Ω. We train a logistic regression probe (Köhn, 2015; Gupta et al., 2015) for each layer that predicts $\omega \in \Omega$ from the last token representation of σ. Given that 6 out of 12 tasks have fewer than 40 datapoints, we use three-fold cross-validation for training and evaluation of this baseline. Note that we have excluded tasks where the probe fails completely due to insufficient number of training examples (fewer than 15 datapoints).

Evaluation We measure the average attribute extraction accuracy. For a given sample, the Patchscope is considered correct if $\exists\, \ell^* \in [1, \ldots, L^*]$ where the generated text up to 20 tokens includes ω. For the probe, a prediction is correct if the highest probability is assigned to ω.

Results Tab. 2 summarizes the results, averaged over $\ell \in [1, \ldots, L]$. We conduct a T-test with Bonferroni correction to compare the two methods. Despite using no training data and having no restrictions on the output, the Patchscope achieves a significantly higher accuracy than the probe on six out of twelve tasks ($p < 1e{-}5$), and works comparably well in all but one of the remaining six. These results suggest that, in the majority of cases, the source representation without its original context carries enough information about many attributes that a targeted Patchscope can extract. We also study how the accuracy changes across the source layers, and observe that Patchscope consistently outperforms the baseline in early layers, outperforms or works on par with the baseline in mid layers, and almost all cases where it performs worse than the baseline occur in later layers. Our interpretation is that given the language modeling training objective, the representations shift toward next-token prediction in the later layers. Therefore, the attribute of interest would not be as readily accessible via the model's computation in these layers. This interpretation is also aligned with recent findings that show no decline in using linear relational embedding in predicting ω only when the next token also happens to be ω (Hernandez et al., 2023b). Note that this pattern explains the higher standard deviation of Patchscope accuracy observed in Tab. 2. We discuss this phenomenon in more detail in §C (see Fig. 6).

4.3. Analyzing Entity Resolution in Early Layers

The previous sections focused on analyzing the information encoded in a single hidden state. Here we turn to consider a more global question of how LLMs resolve entity mentions across multiple layers. Concretely, given a subject entity name, such as *"the summer Olympics of 1996"*, how does the model contextualize the input tokens of the entity and at which layer is it fully resolved?

Answering these questions is hard with existing methods; vocabulary projections focus on the output prediction and fail to show clear patterns in early layers, and probing is restricted to outputs from a fixed number of classes, which may not be expressive enough to describe this process. Alternative approaches have studied this process indirectly via interventions (Meng et al., 2022a), showing that the model constructs a subject representation at the last token of the entity name. However, it is still unclear how this contextualization is performed.

We analyze how LLMs contextualize input entity names by leveraging Patchscopes. Particularly, we craft a target prompt for generating a description of a given subject, and apply it to the hidden representation at the last subject position in the source prompt – where the model forms the subject representation (Geva et al., 2023; Hernandez et al., 2023a) – across the early layers. This will allow us to see

Table 3. Illustrating entity resolution via a qualitative example. The expressive generations show that as we go through the layers, more tokens from the context get integrated into the current representation. The "Generation" column shows the automatically generated text. The "Explanation" column shows our own manually coded interpretation, aiming to specify what entity the generation refers to and how that relates to the tokens processed. $\mathcal{M}^* = \mathcal{M} \leftarrow$ Vicuna (13B), $\ell^* = \ell$, $S \leftarrow$ "Diana, Princess of Wales".

ℓ	Generation	Explanation
1-2	: Country in the United Kingdom	**Wales**
3	: Country in Europe	**Wales**
4	: Title held by female sovereigns in their own right or by queens consort	**Princess of Wales** (unspecific)
5	: Title given to the wife of the Prince of Wales (and later King)	**Princess of Wales** (unspecific)
6	: Diana, Princess of Wales (1961-1997), the first wife of Prince Charles, Prince of Wales, who was famous for her beauty and humanitarian work	**Diana, Princess of Wales**

how the model describes the subject in each layer.

Analysis Setting We use a few-shot target prompt template for decoding an entity description: "subject$_1$: description$_1$, ..., subject$_k$: description$_k$, x", while patching the last position corresponding to x. We take the 200 most popular and 200 least popular subject entities from the PopQA dataset (Mallen et al., 2023). The popular entities should appear frequently in LLMs' pre-training data, and are thus likely to be captured by the model, while resolving the rare entities is expected to be more challenging (Kandpal et al., 2023; Mallen et al., 2023). Then, for the source prompt we use the entity name, and for the target prompt we sample $k = 3$ random subject entities. We obtain a short (up to one sentence) description of every subject entity from Wikipedia. Our target prompt and more technical details are provided in §D.1. We patch the last position representations from the first 10 layers of Vicuna 13B to the target prompt and evaluate the generated subject name and description. Specifically, the generated descriptions are evaluated against the descriptions from Wikipedia using RougeL (Lin, 2004). Evaluation with Rouge1 (Lin, 2004) and Sentence-Bert (Reimers & Gurevych, 2019) shows similar trends (see §D.2).

Results Tab. 3 illustrates the generations by Vicuna 13B for a sample subject entity, when patching its representation at different layers to the target prompt (see more examples in §D.3). For most entities, the contextualization process is spread over the first layers, with the last subject token gradually encompassing more distant positions across layers.

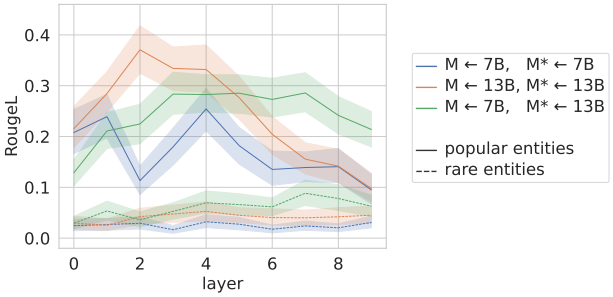


Figure 3. RougeL scores of the generated descriptions against descriptions from Wikipedia, using Vicuna models.

This trend can be quantitatively observed by the similarity between the generated descriptions and the descriptions from Wikipedia, as measured by RougeL. See Fig. 3 where $\mathcal{M} = \mathcal{M}^*$. For both models, similarity increases in the first 5 layers and then slowly decreases. This decrease could potentially be attributed to contamination caused by the representation of the placeholder token "x" remaining in the early layers, when patching is applied to a later layer. Note that this potential issue is only applicable to multi-token generation scenarios as future positions can still attend to the placeholder position in early layers, potentially interfering with the model's ability to accurately generate descriptions for the patched token. See §D.3 for qualitative examples corroborating this interpretation. As expected, the scores for rare, long-tail entities are significantly lower than those of popular entities. See §D.2 for additional results with Pythia where the smaller model seems to outperform the larger model, possibly because the larger model is biased toward output generation at the expense of input contextualization. To summarize, this analysis shows Patchscopes' utility for inspecting the contextualization process in early layers.

4.4. Expressiveness from Cross-Model Patching

A possible avenue for improving inspection capabilities is to explain a given model with a model that is more expressive (Bills et al., 2023). In Patchscopes, this means to patch a representation of $\mathcal{M}$ into a more expressive model $\mathcal{M}^*$. However, it is not clear if such an intervention would yield plausible results, due to possible discrepancies between the two models resulting from different architectures, optimization processes, and so on. Here, we present experiments on next-token prediction and entity resolution that exemplify not only the feasibility, but also the opportunities unlocked by patching across models of the same family.

Next-Token Prediction We repeat the experiment in §4.1, using the token identity Patchscope. To overcome discrepancies between the models, we learn affine mappings between their layers (similarly to Tuned Lens). Our results show that source and target layers on the diagonal exhibit

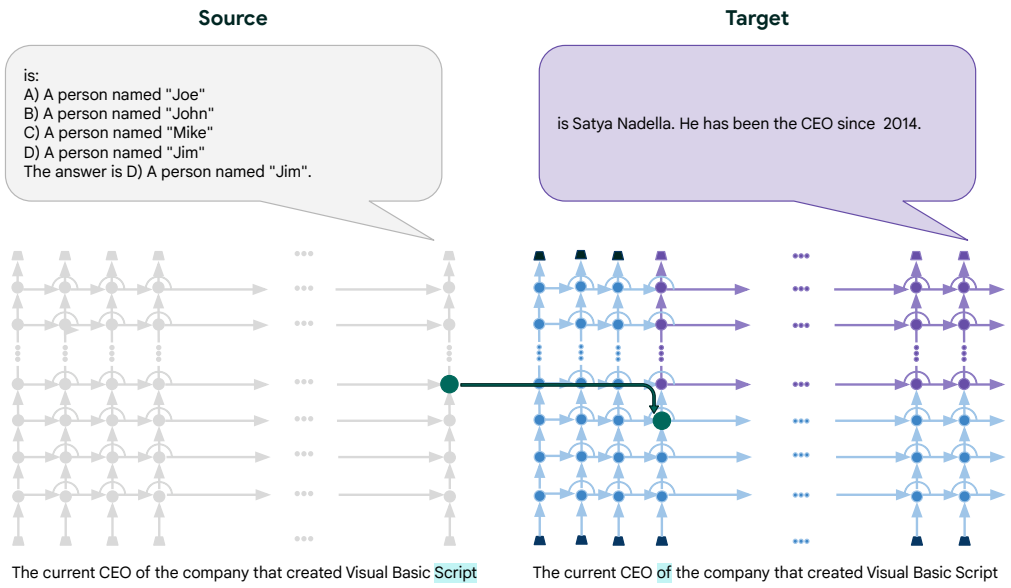


Figure 4. An illustration of CoT `Patchscope` on a single example. In this example, $\pi_1 \leftarrow$ "`the company that created Visual Basic Script`", $\pi_2 \leftarrow$ "`The current CEO of`", $S = T \leftarrow [\pi_2][\pi_1] =$ "`The current CEO of the company that created Visual Basic Script`". Note that $\mathcal{M} = \mathcal{M}^*$ and $f \leftarrow \mathbb{I}$.

the highest precision values, and that patching representations to an early layer of the larger model are the most effective. Overall, suggesting that when $\mathcal{M}^*$ and $\mathcal{M}$ are from the same model family, it is possible to leverage $\mathcal{M}^*$ for decoding information from the representations of $\mathcal{M}$. For detailed results on Precision@1 and Surprisal, see §E.

Entity Resolution in Early Layers We now show that using a large model as $\mathcal{M}^*$ can enhance the output expressivity. To this end, we repeat our entity resolution experiment in §4.3 with Vicuna model family, setting $\mathcal{M}$←7B, $\mathcal{M}^*$←13B. Fig. 3 shows the cross-model patching results (green lines) compared to the same-model patching $\mathcal{M}$←7B, $\mathcal{M}^*$←7B (blue line). The results show that cross-model patching from a smaller model to its larger version generally improves the ability to inspect the input contextualization, both for popular and rare entities. For Pythia, since the smaller model outperforms the larger one, cross-model patching is not as effective (see §D.2).

5. Application: Correcting Multi-Hop Errors

Multi-hop reasoning is a challenging problem (Zhong et al., 2023). While a language model may be capable of correctly answering each step independently, it could still fail at processing the connection between different steps, resulting in an incorrect prediction. Recent attempts to improve multi-hop reasoning rely on prompting the model to generate a step-by-step answer autoregressively (e.g., Wei et al., 2022; Yao et al., 2023; Besta et al., 2024), some with an iterative process of self-refinement (e.g., Madaan et al., 2023). However, achieving similar benefits might be possible via directly rewiring the model's intermediate computation.

Here, we show that `Patchscopes` can improve multi-hop reasoning performance *without* generating the reasoning steps, particularly in cases where the model fails at completing a multi-hop query despite being successful in each reasoning step independently. Via `Patchscopes`, one can surgically operate on the model representations, reroute its intermediate answer to one reasoning step, simplify the consequent step, and ultimately correct the final prediction.

Data Building on Hernandez et al. (2023b), we systematically generate all valid multi-hop factual and commonsense reasoning queries where $\omega_1 = \sigma_2$. We conduct experiments on Vicuna (13B), focusing on samples where $\mathcal{M}$ accurately represents both τ_1 and τ_2 independently, that is, ω appears in the next 20 tokens $\mathcal{M}$ generates conditioned on the prompt π that verbalizes σ and ρ. This process yields 1,104 multi-hop reasoning samples, out of which 46 satisfy the above criteria and are used for evaluation. See more details in §F.

Experimental Setup Following the notation in §4.2, let $\tau_1 = (\sigma_1, \rho_1, \omega_1)$ represent the relation ρ_1 between a subject entity σ_1 and an object entity ω_1. Let $\tau_2 = (\sigma_2, \rho_2, \omega_2)$ represent another tuple such that $\sigma_2 = \omega_1$. A multi-hop reasoning query pertaining to τ_1 and τ_2 is a prompt composed of two parts: π_1 is a

verbalization of σ_1 and ρ_1 from which ω_1 can be inferred; π_2 is a verbalization of ρ_2, from which ω_2 can be inferred after its concatenation with π_1. For example, Let $\tau_1 \leftarrow$ ("Visual Basic", "product of", "Microsoft") and $\tau_2 \leftarrow$ ("Microsoft", "company CEO", "Satya Nadella"). An example verbalization of these tuples is $\pi_1 \leftarrow$"the company that created Visual Basic", $\pi_2 \leftarrow$ "The current CEO of", leading to the multi-hop query $[\pi_2][\pi_1]$ ="The current CEO of the company that created Visual Basic".

Method We introduce a Chain-of-Thought (CoT) Patchscope to fix multi-hop reasoning via intervening on the computation graph and rerouting representation likely to capture ω_1 in place of σ_2. Concretely, S refers to the formed query discussed above, and we use the following configuration: $T \leftarrow S, \mathcal{M}^* \leftarrow \mathcal{M}, i \leftarrow n, i^* \leftarrow$ the token preceding π_1. We evaluate the outputs in terms of accuracy, similarly to §4.2. For a sample S, the Patchscope is considered accurate if $\exists(\ell, \ell^*) : \ell \in [1, \ldots, L], \ell^* \in [1, \ldots, L^*]$ where the autoregressive generation up to 20 tokens includes ω_2. Fig. 4 illustrates the CoT Patchscope with an example. We use the following configuration for CoT Patchscope: $S \leftarrow \pi_1, T \leftarrow \pi_2, i \leftarrow n, i^* \leftarrow m$, which is equivalent to $S = T \leftarrow [\pi_2][\pi_1]$ and adjusting the attention mask such that no token in S has visibility to π_2 and no token in T has visibility to π_1. In addition, we consider two baselines.

Vanilla Baseline For this baseline, we set $S \leftarrow [\pi_1][\pi_2]$, we let the model autoregressively generate up to 20 tokens and check whether ω_2 appears in the generation.

Chain-of-Thought Baseline Here, the setup and evaluation is similar to the vanilla baseline, except that we prepend "Let's think step by step." to S, following (Wei et al., 2022). We then let the model generate up to 20 tokens and check whether ω_2 appears in the generation. Note that this experiment uses Vicuna (13B). Vicuna is based on LLaMA, with supervised finetuning on additional instruction data, which makes it amenable to chain-of-thought prompting.

Results While the vanilla baseline accuracy is only 19.57%, and CoT baseline accuracy is 35.71%, our proposed Patchscope achieves 50% accuracy. For more details about the interaction between ℓ and ℓ^*, and how it affects the success rate, see Fig. 12 in §F.

We emphasize that our primary goal in this section is not to devise a new method to solve multi-hop queries that is necessarily a competitor to CoT, but rather to make a proof-of-concept while comparing with CoT as a common reference. We highlight that we have taken advantage of the extra structural information about the queries given the prior knowledge about how they were synthesized. One could potentially automate this by learning the right places to patch. Li et al. (2024) recently proposed an optimization-based approach for directly patching multi-head self-attention in mid-layers, which can be viewed as soft position-selection, and works effectively in multihop error correction, suggesting that inferring the right positions to patch could be effective. However, even if optimal source and target position are automatically decided upfront, Patchscope and CoT may not be directly comparable. CoT generates multiple steps which fundamentally extend the computational power of the LLM (Merrill & Sabharwal, 2024), but a Patchscope with pre-identified (l, l^*) only makes $O(1)$ inference passes.

6. Conclusion

We present Patchscopes, a simple and effective framework that leverages the ability of LLMs to generate human-like text for decoding information from intermediate LLM representations. We show that many existing interpretability methods can be cast as specific Patchscope instances, and even these only cover a small portion of the framework's possible configurations. Moreover, new underexplored Patchscopes substantially improve our ability to decode various types of information from the model's internal computation, such as the output prediction and knowledge attributes, typically outperforming prominent methods that rely on projection to the vocabulary and probing. In addition, our framework enables new capabilities, such as analyzing the contextualization process of input tokens in early LLM layers, and can correct multi-hop reasoning.

There are multiple future research directions to consider. An important factor in the effectiveness of a chosen target prompt is how the information from the patched position propagates during inference to other positions and across layers. Understanding how to best use a given target prompt, perhaps automatically, is an important factor for using Patchscopes. Another avenue for future work is investigating the effectiveness of few-shot target prompts compared to zero-shot/instruction-based prompts. More expressive instruction-based target prompts, for example, could enable extracting more complex information. Additionally, while our cross-model patching focused on models from the same family, it will be valuable to explore which mapping functions would enable patching across models from different families and perhaps with different architectures. Other directions for future work include applications across different domains and modalities, investigating variants with simultaneous multi-token patching or multi-layer patching to mitigate the risks of placeholder contamination, and presenting recipes for task-specific and task-agnostic Patchscopes.

Acknowledgements

We thank Amir Globerson for feedback on writing and presentation of results. We thank Ardavan Saeedi, Martin Wattenberg, Ellie Pavlick, the AI Explorables team[3] at Google Research, and Jasmijn Bastings for their helpful comments.

Impact Statement

Societal Impact This paper presents a new framework for interpreting hidden representations of large language models. Interpretability methods in general can be used to investigate models' reliability and safety prior to deployment. We hope that Patchscopes framework facilitates progress in this area with the introduction of inspection tools that are more expressive, robust across layers, and do not require training data.

Limitations While our proposed framework is not limited to any particular architecture or domain, the experimental evidence provided in this paper focuses on autoregressive Transformer-based language models, and future work is needed to verify its effectiveness in other setups.

References

Alain, G. and Bengio, Y. Understanding intermediate layers using linear classifier probes. *5th International Conference on Learning Representations, Workshop Track Proceedings*, 2017.

Bansal, Y., Nakkiran, P., and Barak, B. Revisiting model stitching to compare neural representations. *Advances in neural information processing systems*, 34:225–236, 2021.

Belinkov, Y. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

Belinkov, Y. and Glass, J. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72, 2019.

Belrose, N., Furman, Z., Smith, L., Halawi, D., Ostrovsky, I., McKinney, L., Biderman, S., and Steinhardt, J. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Gianinazzi, L., Gajda, J., Lehmann, T., Podstawski, M., Niewiadomski, H., Nyczyk, P., et al. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the 38th AAAI Conference on Artificial Intelligence*, 2024.

[3] https://pair.withgoogle.com/explorables

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

Bills, S., Cammarata, N., Mossing, D., Tillman, H., Gao, L., Goh, G., Sutskever, I., Leike, J., Wu, J., and Saunders, W. Language models can explain neurons in language models. *URL https://openaipublic. blob. core. windows. net/neuron-explainer/paper/index. html.(Date accessed: 14.05. 2023)*, 2023.

Casper, S., Rauker, T., Ho, A., and Hadfield-Menell, D. Sok: Toward transparent ai: A survey on interpreting the inner structures of deep neural networks. In *First IEEE Conference on Secure and Trustworthy Machine Learning*, 2022.

Chiang, W.-L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J. E., Stoica, I., and Xing, E. P. Vicuna: An open-source chatbot impressing GPT-4 with 90%* ChatGPT quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

Conmy, A., Mavor-Parker, A. N., Lynch, A., Heimersheim, S., and Garriga-Alonso, A. Towards automated circuit discovery for mechanistic interpretability. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Csiszárik, A., Kőrösi-Szabó, P., Matszangosz, Á. K., Papp, G., and Varga, D. Similarity and matching of neural network representations. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=aedFIIRRfXr.

Dar, G., Geva, M., Gupta, A., and Berant, J. Analyzing transformers in embedding space. In *Annual Meeting of the Association for Computational Linguistics*, 2023.

Din, A. Y., Karidi, T., Choshen, L., and Geva, M. Jump to conclusions: Short-cutting transformers with linear transformations. *arXiv preprint arXiv:2303.09435*, 2023.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

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

Geva, M., Caciularu, A., Dar, G., Roit, P., Sadde, S., Shlain, M., Tamir, B., and Goldberg, Y. LM-debugger: An interactive tool for inspection and intervention in transformer-based language models. In Che, W. and Shutova, E. (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 12–21, Abu Dhabi, UAE, December 2022a. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-demos.2. URL https://aclanthology.org/2022.emnlp-demos.2.

Geva, M., Caciularu, A., Wang, K., and Goldberg, Y. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In Goldberg, Y., Kozareva, Z., and Zhang, Y. (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 30–45, Abu Dhabi, United Arab Emirates, December 2022b. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.3. URL https://aclanthology.org/2022.emnlp-main.3.

Geva, M., Bastings, J., Filippova, K., and Globerson, A. Dissecting recall of factual associations in auto-regressive language models. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12216–12235, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.751. URL https://aclanthology.org/2023.emnlp-main.751.

Ghandeharioun, A., Kim, B., Li, C.-L., Jou, B., Eoff, B., and Picard, R. W. Dissect: Disentangled simultaneous explanations via concept traversals. *arXiv preprint arXiv:2105.15164*, 2021.

Goldowsky-Dill, N., MacLeod, C., Sato, L., and Arora, A. Localizing model behavior with path patching. *arXiv preprint arXiv:2304.05969*, 2023.

Gupta, A., Boleda, G., Baroni, M., and Padó, S. Distributional vectors encode referential attributes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 12–21, 2015.

Hanna, M., Liu, O., and Variengien, A. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=p4PckNQR8k.

Hase, P., Bansal, M., Kim, B., and Ghandeharioun, A. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *arXiv preprint arXiv:2301.04213*, 2023.

Hendel, R., Geva, M., and Globerson, A. In-context learning creates task vectors. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Hernandez, E., Schwettmann, S., Bau, D., Bagashvili, T., Torralba, A., and Andreas, J. Natural language descriptions of deep features. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=NudBMY-tzDr.

Hernandez, E., Li, B. Z., and Andreas, J. Measuring and manipulating knowledge representations in language models. *arXiv preprint arXiv:2304.00740*, 2023a.

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

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

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pp. 2668–2677. PMLR, 2018.

Köhn, A. What's in an embedding? analyzing word embeddings through multilingual evaluation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 2067–2073, 2015.

Langedijk, A., Mohebbi, H., Sarti, G., Zuidema, W., and Jumelet, J. Decoderlens: Layerwise interpretation of encoder-decoder transformers. *arXiv preprint arXiv:2310.03686*, 2023.

Lenc, K. and Vedaldi, A. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 991–999, 2015.

Li, Z., Jiang, G., Xie, H., Song, L., Lian, D., and Wei, Y. Understanding and patching compositional reasoning in llms. *arXiv preprint arXiv:2402.14328*, 2024.

Lieberum, T., Rahtz, M., Kramár, J., Irving, G., Shah, R., and Mikulik, V. Does circuit analysis interpretability scale? evidence from multiple choice capabilities in chinchilla. *arXiv preprint arXiv:2307.09458*, 2023.

Lin, C.-Y. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B. P., Hermann, K., Welleck, S., Yazdanbakhsh, A., and Clark, P. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=S37hOerQLB.

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

Mallen, A., Asai, A., Zhong, V., Das, R., Khashabi, D., and Hajishirzi, H. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9802–9822, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.546. URL https://aclanthology.org/2023.acl-long.546.

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

Meng, K., Sharma, A. S., Andonian, A. J., Belinkov, Y., and Bau, D. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*, 2022b.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2016.

Merrill, W. and Sabharwal, A. The expressive power of transformers with chain of thought. In *The Twelfth International Conference on Learning Representations*, 2024.

Merullo, J., Castricato, L., Eickhoff, C., and Pavlick, E. Linearly mapping from image to text space. In *The Eleventh International Conference on Learning Representations*, 2022.

Merullo, J., Eickhoff, C., and Pavlick, E. A mechanism for solving relational tasks in transformer language models. *arXiv preprint arXiv:2305.16130*, 2023.

Mousi, B., Durrani, N., and Dalvi, F. Can llms facilitate interpretation of pre-trained language models? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, December 2023. URL https://browse.arxiv.org/pdf/2305.13386.pdf.

Nanda, N., Lee, A., and Wattenberg, M. Emergent linear representations in world models of self-supervised sequence models. In Belinkov, Y., Hao, S., Jumelet, J., Kim, N., McCarthy, A., and Mohebbi, H. (eds.), *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 16–30, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.blackboxnlp-1.2. URL https://aclanthology.org/2023.blackboxnlp-1.2.

nostalgebraist. interpreting gpt: the logit lens. *LessWrong*, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Pal, K., Sun, J., Yuan, A., Wallace, B. C., and Bau, D. Future lens: Anticipating subsequent tokens from a single hidden state. In *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, pp. 548–560, 2023.

Patel, R. and Pavlick, E. Mapping language models to grounded conceptual spaces. In *International Conference on Learning Representations*, 2021.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

Schwettmann, S., Chowdhury, N., Klein, S., Bau, D., and Torralba, A. Multimodal neurons in pretrained text-only transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2862–2867, 2023.

Singh, C., Hsu, A., Antonello, R., Jain, S., Huth, A., Yu, B., and Gao, J. Explaining black box text modules in natural language with language models. In *XAI in Action: Past, Present, and Future Applications*, 2023. URL https://openreview.net/forum?id=3BX9tM03GT.

Slobodkin, A., Goldman, O., Caciularu, A., Dagan, I., and Ravfogel, S. The curious case of hallucinatory (un)answerability: Finding truths in the hidden states of over-confident large language models. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3607–3625, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.220. URL https://aclanthology.org/2023.emnlp-main.220.

Stolfo, A., Belinkov, Y., and Sachan, M. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7035–7052, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.435. URL https://aclanthology.org/2023.emnlp-main.435.

Strobelt, H., Gehrmann, S., Pfister, H., and Rush, A. M. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE transactions on visualization and computer graphics*, 24(1):667–676, 2017.

Tenney, I., Das, D., and Pavlick, E. Bert rediscovers the classical nlp pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4593–4601, 2019.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

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

Vilas, M. G., Schaumlöffel, T., and Roig, G. Analyzing vision transformers for image classification in class embedding space. *arXiv preprint arXiv:2310.18969*, 2023.

Wallat, J., Singh, J., and Anand, A. BERTnesia: Investigating the capture and forgetting of knowledge in BERT. In Alishahi, A., Belinkov, Y., Chrupała, G., Hupkes, D., Pinter, Y., and Sajjad, H. (eds.), *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pp. 174–183, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.blackboxnlp-1.17. URL https://aclanthology.org/2020.blackboxnlp-1.17.

Wang, B. and Komatsuzaki, A. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

Wang, K. R., Variengien, A., Conmy, A., Shlegeris, B., and Steinhardt, J. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*, 2022.

Wang, Z., Ku, A., Baldridge, J., Griffiths, T. L., and Kim, B. Gaussian Process Probes (GPP) for uncertainty-aware probing. *arXiv preprint arXiv:2305.18213*, 2023.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837, 2022.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. R. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Youssef, P., Koraş, O., Li, M., Schlötterer, J., and Seifert, C. Give me the facts! a survey on factual knowledge probing in pre-trained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 15588–15605, 2023.

Zhang, F. and Nanda, N. Towards best practices of activation patching in language models: Metrics and methods. *arXiv preprint arXiv:2309.16042*, 2023.

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

Zhou, B., Bau, D., Oliva, A., and Torralba, A. Interpreting deep visual representations via network dissection. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2131–2145, 2018.

A. Detailed Configuration of Prior Methods as Patchscope Instances

In this section, we first reiterate Patchscopes premise and distinguish it from prior work. We then detail Patchscope configurations that replicate prior methods. We highlight that clearly formulating Patchscopes in the form of source and target quintuplets, respectively (S, i, M, l) and (T, i^*, M^*, l^*), is not in itself a contribution of this work. However, it is what facilitates the main contributions that are 3-fold: 1) The framework ties methods that otherwise might seem unrelated, e.g., logit lens and attention knockout, 2) makes it easy to come up with new configurations that significantly improve them, 3) makes it easy to see the space that prior methods do not cover, and therefore open up new applications altogether.

We focus on a few inspection objectives. Note that some of the baselines considered have not been originally designed to address these objectives, but are the closest solutions in literature. Particularly, we ask, given a (set of) hidden representation(s) detached from their context, how well can we do next-token prediction, feature extraction, and analysis of model's contextualization process.

A.1. Next-token Prediction

Goal: Estimating output probability distribution p^L given a hidden representation h^l, particularly to inspect when the model has enough information to make the final prediction.

Prior Work: Many recent methods to estimate output probability distribution given an inner representation use projection onto the vocabulary space (e.g., nostalgebraist, 2020; Din et al., 2023; Belrose et al., 2023). Considering (S, i, M, l) and (T, i^*, M^*, l^*) formulation above, all these methods can be seen as Patchscopes where $\mathbf{M} = \mathbf{M}^*, \mathbf{l}^* = \mathbf{L}$, where $\mathbf{L}$ is the number of layers.

Distinguishing Patchscope & its Improvements: We propose the token identity Patchscope that unlike (Belrose et al., 2023; Din et al., 2023) does not require a learned mapping, yet is more successful. Note that in this task, setting $\mathbf{l}^* = \mathbf{L}$ which all the prior methods discussed above have in common, means that the choice of $\mathbf{T}$ does not matter. However, in our proposed Patchscope, we show that setting $\mathbf{l}^* = \mathbf{l}$ where l is arbitrary layer, and using a few-shot token identity prompt $\mathbf{T}$ that encourages repetition results in significant improvements, and by using the target model's computation from layer l^* onward, there is no need for training a separate mapping function.

A.2. Feature Extraction

Goal: Extracting specific attributes of the subject given its hidden representation h^l (e.g., extracting "largest city of" from the representation of "States" in "United States").

Prior Work: Classification probes (Alain & Bengio, 2017) are the most commonly used methods for this purpose. They work by training a classifier on a dataset of prompts and their corresponding labels (e.g., countries and their corresponding largest cities).

Distinguishing Patchscope & its Improvements: We show an intuitive Patchscope (i.e., $T =$ "The largest city in x") significantly outperforms probing across various tasks and layers. Note that this Patchscope 1) does not require supervised training, 2) is not limited by a predefined set of labels, and 3) is more accurate than a linear probe.

A.3. Analysis of Model's Contextualization Process

Goal: Identifying how entity tokens are processed, and at which layer the entity is fully resolved.

Prior Work: It is hard to answer this question with existing methods. First, methods that use vocabulary projection are focused on output prediction (rather than input processing) and fail particularly in early layers, precisely where entity resolution happens. Second, probing requires a predefined set of classes, which is not expressive enough to show the gradual entity resolution process, and nontrivial. For example, consider "Alexander the grea" input. We found that "Great Britain" and "Great depression" arose in the gradual resolution process. However, these are non-trivial choices a priori. We can train a multi-class classifier to choose one out of all possible entities (or binary classifiers for every possible entity), which is unlikely to work given the huge space of possible entities. Lastly, most prior work using activation patching aims to answer a different question: whether a certain representation plays a key role in the model's forward computation and the

final prediction (e.g., Meng et al., 2022a; Geva et al., 2023). However, they can be viewed as Patchscope instances, and provide indirect signals about the entity resolution process, namely, showing subject information accrues in the last token after a few layers.

Distinguishing Patchscope & its Improvements: To the best of our knowledge, there are not any methods that verbalize the gradual entity resolution process in coherent text, and the proposed Patchscope is the first to do that.

In summary, as discussed in 3.2, many prior methods can be seen as Patchscopes. Tab. 4 details the configurations of Patchscopes that yield various methods introduced in prior work.

Table 4. Patchscopes is a novel framework for inspection of hidden representations in language models. Many prior inspection methods with various objectives can be viewed as Patchscopes, as detailed in the "Configuration" column (see notation description in §3). The rows highlighted in green show our new configurations that overcome several limitations of prior methods through more expressive inspection that is training-data free and is more robust across layers. Additionally, the generality of this framework enables novel inspection possibilities that were unexplored before. When the target prompt (T) is not specified, it means that the output would be invariant to the choice of T. When not specified, $f \leftarrow \mathbb{I}$ and $\mathcal{M}^* = \mathcal{M}$.

Inspection Objective		Expressive	Training Data Free	Robust Across Layers	Configuration
Inspecting Output Distribution	Few-shot token identity Patchscope (§4.1)	✔✔	✔	✔✔	$\ell^* \leftarrow \ell$, $T \leftarrow$ "$\texttt{tok}_1 \rightarrow \texttt{tok}_1$; $\texttt{tok}_2 \rightarrow \texttt{tok}_2$;...;$\texttt{tok}_k$"
	Logit Lens (nostalgebraist, 2020), Embedding Space Analysis (Dar et al., 2023)	✔	✔	✖	$\ell^* \leftarrow L^*$
	Tuned Lens (Belrose et al., 2023)	✔	For learning mappings	✔	$\ell^* \leftarrow L^*, f \leftarrow$ Affine
	Future Lens (Pal et al., 2023)	✔	For learning mappings	✔✔	$\ell^* \leftarrow \ell, f \leftarrow$ Linear, $T \leftarrow$ Fixed or learned soft prompt
Feature Extraction	Zero-shot feature extraction Patchscope (§4.2)	✔✔	✔	✔✔	$\ell^* \leftarrow j' \in [1, \ldots, L^*], i^* \leftarrow m$, $T \leftarrow$ relation verbalization followed by $\texttt{x}$
	LRE Attribute Lens (Hernandez et al., 2023b)	✔	For linear relation approx.	✔✔	$\ell^* \leftarrow L^*, f \leftarrow$ Linear with additional variables, $T \leftarrow S$
	Probing (e.g., Belinkov & Glass, 2019; Belinkov, 2022; Alain & Bengio, 2017; Wang et al., 2023)	✖	For training probe	✔	N/A
Entity Resolution	Entity description Patchscope (§4.3)	✔✔	✔	✔✔	$\ell^* \leftarrow \ell, i^* \leftarrow m$, $T \leftarrow$ "$\texttt{subject}_1\texttt{: description}_1\texttt{, ..., subject}_k\texttt{: description}_k\texttt{ , x}$"
	X-model entity description Patchscope (§4.4)	✔✔✔	For learning mappings	✔✔	$\mathcal{M}^* \leftarrow$ a larger variant of $\mathcal{M}, \ell^* \leftarrow \ell, i^* \leftarrow m$, $T \leftarrow$ "$\texttt{subject}_1\texttt{: description}_1\texttt{, ... , subject}_k\texttt{: description}_k\texttt{, x}$"
	Causal Tracing (Meng et al., 2022a)	✖	✔	✔✔	$\ell^* \leftarrow \ell, T \leftarrow S + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma)$
	Attention Knockout (Wang et al., 2022; Conmy et al., 2023; Geva et al., 2023)	✖	✔	✔✔	$\ell^* \leftarrow$ Multiple, $f \leftarrow 0, T \leftarrow S$
Inspection Application	Early Exiting, e.g., Linear Shortcuts (Din et al., 2023)	✔	For learning mappings	✔	$\ell^* \leftarrow L^*, f \leftarrow$ Affine
	Caption Generation, e.g., Linear Mapping (Merullo et al., 2022)	✔	For learning mappings	✔	$\mathcal{M}^* \leftarrow$ A language model of choice, $\ell^* \leftarrow L^*, f \leftarrow$ Affine

B. Next-Token Prediction Additional Details and Experimental Results

In this section, we provide more information about the models, data, and additional target prompts for few-shot token identity Patchscope configurations. It is worth highlighting that after patching, we continue computations from the target layer onward. Therefore, since in this Patchscope, source layer and target layer are the same, there are going to be more computations compared to baseline methods like LogitLens (nostalgebraist, 2020; Geva et al., 2022a) and Tuned Lens (Belrose et al., 2023; Din et al., 2023) where the target layer is fixed at the final layer.

B.1. Models

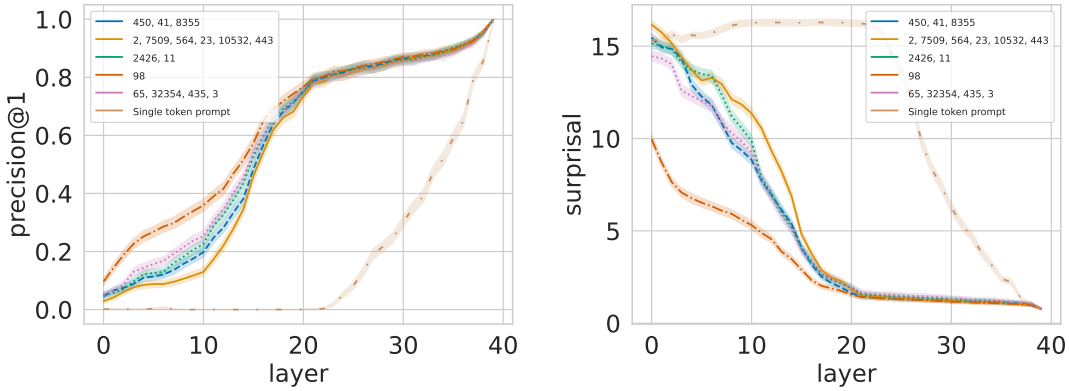
We use LLaMA2 (13B)[4] (Touvron et al., 2023b), Vicuna (13B)[5] (Chiang et al., 2023), GPT-J (6B) (Wang & Komatsuzaki, 2021)[6], and Pythia (12B)[7] (Biderman et al., 2023). LLaMA2 was pre-trained on 2T tokens from a mix of publicly available data. Vicuna is a LLaMA1 (Touvron et al., 2023a) model that was pre-trained on 1T tokens and fine-tuned on 70K

[4]LLaMA 2 is licensed under the LLAMA 2 Community License, Copyright © Meta Platforms, Inc. All Rights Reserved.

[5]Vicuna is subject to the LLAMA 2 Community License, terms of use of the data generated by OpenAI, and privacy practices of ShareGPT. The code is released under the Apache License 2.0.

[6]GPT-J-6B weights are licensed under version 2.0 of the Apache License.

[7]Pythia is released under Apache License 2.0, Copyright © 2023 EleutherAI.



(a) Precision@1 (b) Surprisal

Figure 5. Next-token prediction results for LLaMA2, using various token identity demonstrations (the token IDs appear in the legend). We report precision@1 (↑ is better), and surprisal (↓ is better).

user-shared conversations [8]. The primary architectural differences between LLaMA2 and Vicuna (LLaMA1) include a different context length and grouped-query attention. Pythia and GPT-J were pre-trained using a deduplicated version of the Pile corpus[9] (Gao et al., 2020), and for about 300B and 402B tokens, respectively.

B.2. Training and Evaluation Data

We use 12,000 random samples from the Pile, partitioned into 10,000 examples for training the affine mappings, and 2,000 examples for evaluation. In our pre-processing strategy, we introduce randomness in the patching positions by trimming the input sequence length of each example.

B.3. Additional Few-Shot Token Identity Prompts

In this section, we provide additional details about the selection of the demonstrations for the token identity baseline, and further evaluate the robustness of LLaMA2 (13B) (Touvron et al., 2023b) to various token identity prompts.

Demonstrations Construction For the demonstrations used in this experiment, we sample a random set of k tokens for all the models, where k was also randomly sampled from the interval $[1, \ldots, 10]$.

Robustness to Additional token IDs' Demonstrations We randomly generate five realizations of token IDs series of varying lengths, formatted as "$\texttt{tok}_1 \rightarrow \texttt{tok}_1$; $\texttt{tok}_2 \rightarrow \texttt{tok}_2$; ... ; $\texttt{tok}_k$", similarly to the procedure from §4.1. We also include an ablation experiment, introducing the *Single Token Prompt* baseline. In this approach, the representation is transferred from the original multi-token prompt to a single-token prompt, while preserving the same position. This is crucial for maintaining consistent positional biases during the computation in the forward pass. The baseline is implemented by creating a target prompt in the format: "[PAD] [PAD] ... [PAD] $\texttt{tok}_k$". Here, the number of [PAD] tokens is chosen such that "$\texttt{tok}_k$" retains its original position from the source prompt.

The results are illustrated in Fig. 5, where a comprehensive overview of the evaluation metrics can be found in §4.1. The results indicate the stability of the token identity baseline across a range of token identity demonstrations, particularly notable in the upper layers of the model. The *Single Token Prompt* baseline overall seems to be less effective than using our token ID baseline.

Table 5. Comparison between the zero-shot feature extraction Patchscope and the logistic regression probe shows that despite using *no training data*, the Patchscope has a significantly higher accuracy than the baseline in most tasks ($p < 1e-5$). Pairwise t-statistics and the corresponding p-values are included in the table. The columns corresponding to each method show accuracy (mean $\pm$ std).

	Task	Probe	Patchscope	T-statistic	p-value
Commonsense	Fruit inside color	37.41 ± 6.58	37.99 ± 18.67	0.126	0.901
	Fruit outside color	35.50 ± 3.09	$\mathbf{71.00 \pm 13.26}^{**}$	12.426	$< 1e-5$
	Object superclass	$\mathbf{68.92 \pm 10.69}^{**}$	55.71 ± 10.81	-5.25	$< 1e-4$
	Substance phase	73.77 ± 3.74	$\mathbf{91.92 \pm 1.73}^{**}$	25.647	$< 1e-5$
	Task done by person	0 ± 0	$\mathbf{62.96 \pm 16.513}^{**}$	19.632	$< 1e-5$
	Task done by tool	10.14 ± 3.23	$\mathbf{48.12 \pm 13.23}^{**}$	18.231	$< 1e-5$
	Work location	0 ± 0	$\mathbf{13.58 \pm 9.37}^{**}$	7.45990	$< 1e-5$
Factual	Company CEO	4.99 ± 2.56	$\mathbf{47.82 \pm 13.89}^{**}$	16.700	$< 1e-5$
	Country capital city	0 ± 0	$\mathbf{61.61 \pm 14.14}^{**}$	22.426	$< 1e-5$
	Country currency	17.70 ± 2.20	$\mathbf{50.95 \pm 8.85}^{**}$	20.293	$< 1e-5$
	Country largest city	0 ± 0	$\mathbf{67.78 \pm 11.47}^{**}$	30.427	$< 1e-5$
	Food from country	5.13 ± 3.66	$\mathbf{63.80 \pm 11.34}^{**}$	26.710	$< 1e-5$
	Person father	0 ± 0	$\mathbf{25.34 \pm 8.42}^{**}$	15.482	$< 1e-5$
	Person plays position in sport	75.89 ± 9.14	72.20 ± 7.21	-2.066	0.049
	Person plays pro sport	53.87 ± 10.28	46.28 ± 14.19	-2.020	0.054
	Product by company	58.91 ± 7.15	63.24 ± 10.74	1.757	0.091
	Star constellation	17.54 ± 5.30	18.35 ± 5.06	-2.98	0.006
	Superhero archnemesis	0 ± 0	$\mathbf{41.73 \pm 18.72}^{**}$	11.47044	$< 1e-5$
	Superhero person	0 ± 0	$\mathbf{28.32 \pm 14.05}^{**}$	10.37461	$< 1e-5$

C. More Details on Attribute Extraction Experiments

Dataset Details We start from the factual and commonsense reasoning subsets introduced by Hernandez et al. (2023b)[10]. This dataset includes 8 commonsense and 25 factual relations. Recall that each datapoint is representing a triplet (σ, ρ, ω), where σ refers to the subject, ρ represents the relation, and ω stands for the object (see 4.2 for notation details). For each data point representing the triplet (σ, ρ, ω), we sample 5 utterances from Wikitext-103 dataset (Merity et al., 2016) including σ . We then truncate the sampled text to a window of random length up to 20 tokens that contains σ. This constitutes our source prompt, S. Note that for each model, we filter the data to samples for which the underlying model correctly encodes the tuple. For experiments with zero-shot target prompt T that includes ρ followed by σ, we autoregressively generate the next 20 tokens, and only keep examples where ω appears in the generation. For experiments with few-shot demonstrations in T, after generating the next 20 tokens, we do an additional post-processing step. If the demonstration template of an example is identified in the generation, all the following tokens would be dropped. The example is used for evaluation only if ω appears in the post-processed truncated generation. To have a reasonable amount of data for training the classification probe baseline, tasks with fewer than 15 datapoints are dropped from the analysis. For GPT-J, 5 commonsense and 7 factual reasoning tasks remain after applying the above postprocessing steps.

Results on Additional Tasks We provide additional results on various factual and commonsense reasoning tasks. For a comparison of the zero-shot feature extraction Patchscope and the logistic regression probe averaged across layers, see t-statistic details in Tab. 5.

Performance Breakdown Across Source Layers Fig. 6 depicts how accuracy changes across source layers $\ell \in [1, \ldots, L]$. Considering the early layers, we observe that the Patchscope consistently outperforms the baseline. This further confirms our hypothesis that prior methods are particularly limited in surfacing information in the early layers, which often cannot be decoded via linear functions. However, Patchscope is able to extract such attributes significantly earlier. Note that this observation does not mean the attribute is *explicitly* encoded in a representation, but that there is enough information encoded such that the attribute can be extracted from the representation alone, without its original context, using the model's computation. In the middle layers, Patchscope works similarly or better than the baseline. Interestingly, we observe

[8]Conversations were collected from www.sharegpt.com.

[9]Pile dataset is available under MIT license.

[10]The dataset released by Hernandez et al. (2023b) is available under MIT license.

that almost all cases where Patchscope performs worse than the baseline occur in later layers. Our interpretation is that given the language modeling training objective, the representations shift toward next-token prediction in the later layers. Therefore, the attribute of interest would not be as readily accessible via the model's computation in these layers. This interpretation is also aligned with recent findings that show no decline in using linear relational embedding in predicting ω only when the next token also happens to be ω (Hernandez et al., 2023b). We postulate that when the immediate next token is not the attribute of interest, it does not necessarily mean the attribute information is lost, but rather it may not be as easily accessible on the surface. We hypothesize that using a Patchscope with a more expressive mapping f could improve attribute extraction accuracy in the later layers, which we leave for future work to verify.

Source-Target Layer Interplay Fig. 7 visualizes the interaction between ℓ and ℓ^*. These heatmaps show attribute extraction success rate for the zero-shot feature extraction Patchscope for a fixed (ℓ, ℓ^*) combination. The lower left quadrants show setups where both ℓ and ℓ^* represent early to middle layers, and the success rate is maximal. The right half of the heatmaps represents late ℓ, which achieves lower success rate due to token representations shifting toward next-token prediction as discussed earlier. In addition, we notice lower success rate in the top half of the heatmaps which represents late ℓ^*. It is worth noting that in this task, the accuracy is not only based on the immediate next-token prediction, but rather whether ω appears in the next 20 autoregressively generated tokens. The placeholder token "x" does still remain in the input, and its representation persists in the early layers in the target computation. This explains why a lower attribute extraction rate is observed in later ℓ^* values. We leave it to the future work to investigate adaptations to Patchscopes to control contamination from the placeholder tokens and make them more amenable to late ℓ^* choices.

Statistical Test Details The null hypothesis is: "There is no significant difference between probing accuracy and patchscope accuracy". We calculate attribute extraction accuracy per layer, in a 40-layer model, across various tasks. Since we run a test for each task independently (see Tab. 2), we use the conservative Bonferroni correction to address the multiple comparison problem. That is, for a desired overall $\alpha = 0.05$, we consider $\alpha/\text{number of tasks} = \frac{0.05}{12} = 0.004$ for each individual task.

D. Additional Information and Results on the Entity Resolution Experiment

D.1. Experimental Setup

Recall that we use a few-shot target prompt template for decoding an entity description: "$\texttt{subject}_1$: $\texttt{description}_1$, $\texttt{subject}_2$: $\texttt{description}_2$, ..., $\texttt{subject}_k$: $\texttt{description}_k$, x", while patching the last position which corresponds to "x". Specifically, we use the following target prompt, obtained randomly: "Syria: Country in the Middle East, Leonardo DiCaprio: American actor, Samsung: South Korean multinational major appliance and consumer electronics corporation, x" and task the model to generate the completion after the patched representation in "x". For the subject description, which is composed of $k = 3$ random subject entities, we used the wptools[11] python package for obtaining a description of every subject entity from Wikipedia.

D.2. Additional Quantitative Results

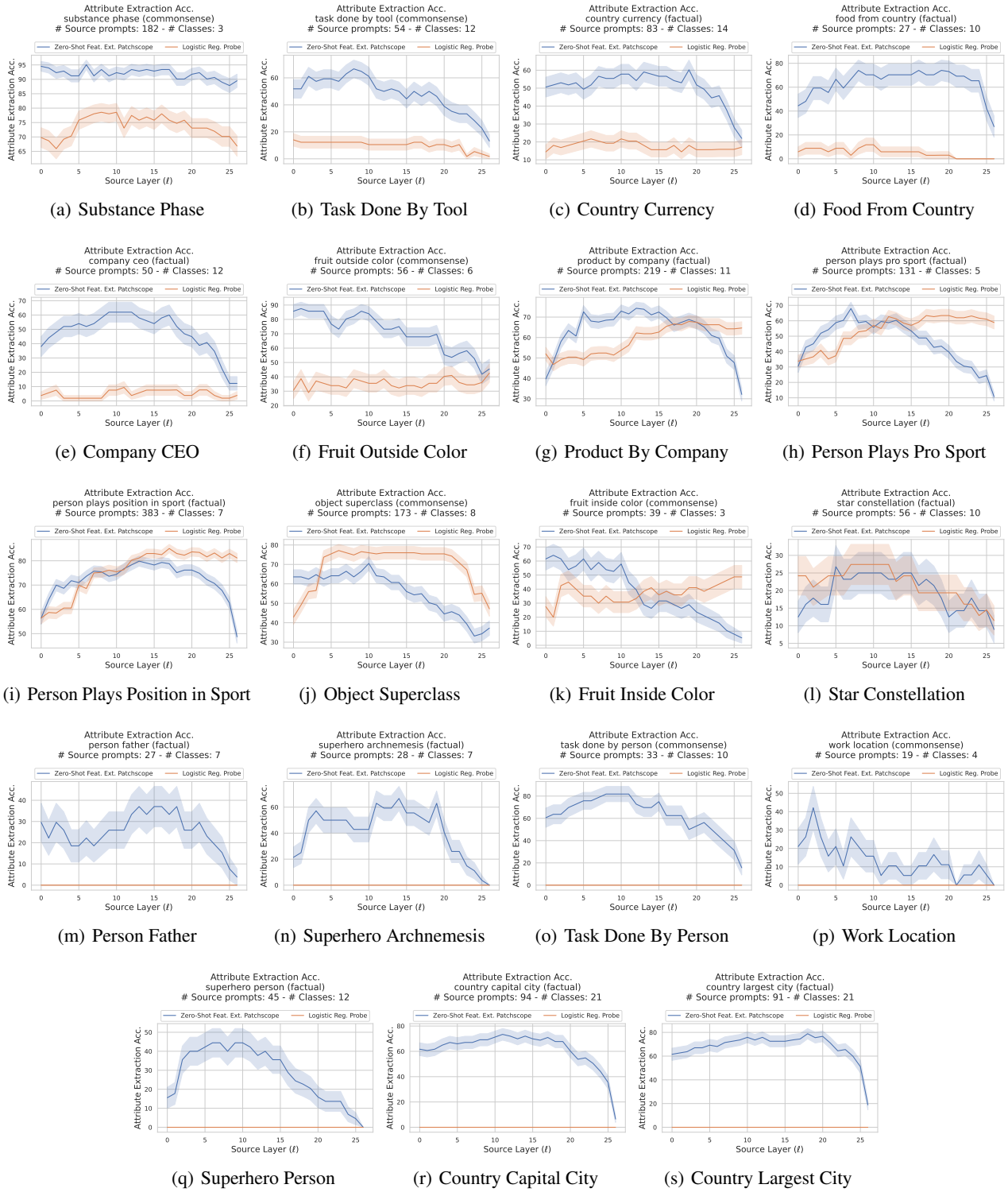
In this section, we present the Rouge1 (Lin, 2004) and SBERT score (Reimers & Gurevych, 2019) results, as well as the results for the Pythia models (Biderman et al., 2023).[12] In Fig. 8 and Fig. 9, we present the Rouge1 and the SBERT results, respectively, complementary to the RougeL results in Fig. 3 from §4.3. Note that for Pythia, the smaller model (6.9B) outperforms the larger one (12B), and hence, our cross-model patching method would not be able to improve the inspection of the smaller model, unlike the trends we observed for Vicuna.

D.3. Additional Qualitative Results

In this section, we provide more examples and discuss our observations about the gradual process of entity resolution. As shown in Tab. 6 and Tab. 7, it is interesting to observe that the resolution process for the same input can look different across models, suggesting they assign different likelihoods to different entities, and weigh context differently. For example,

[11] https://github.com/siznax/wptools/

[12] We used the package from https://www.sbert.net/, with the sentence-transformers/all-MiniLM-L6-v2 model.



(a) Substance Phase (b) Task Done By Tool (c) Country Currency (d) Food From Country

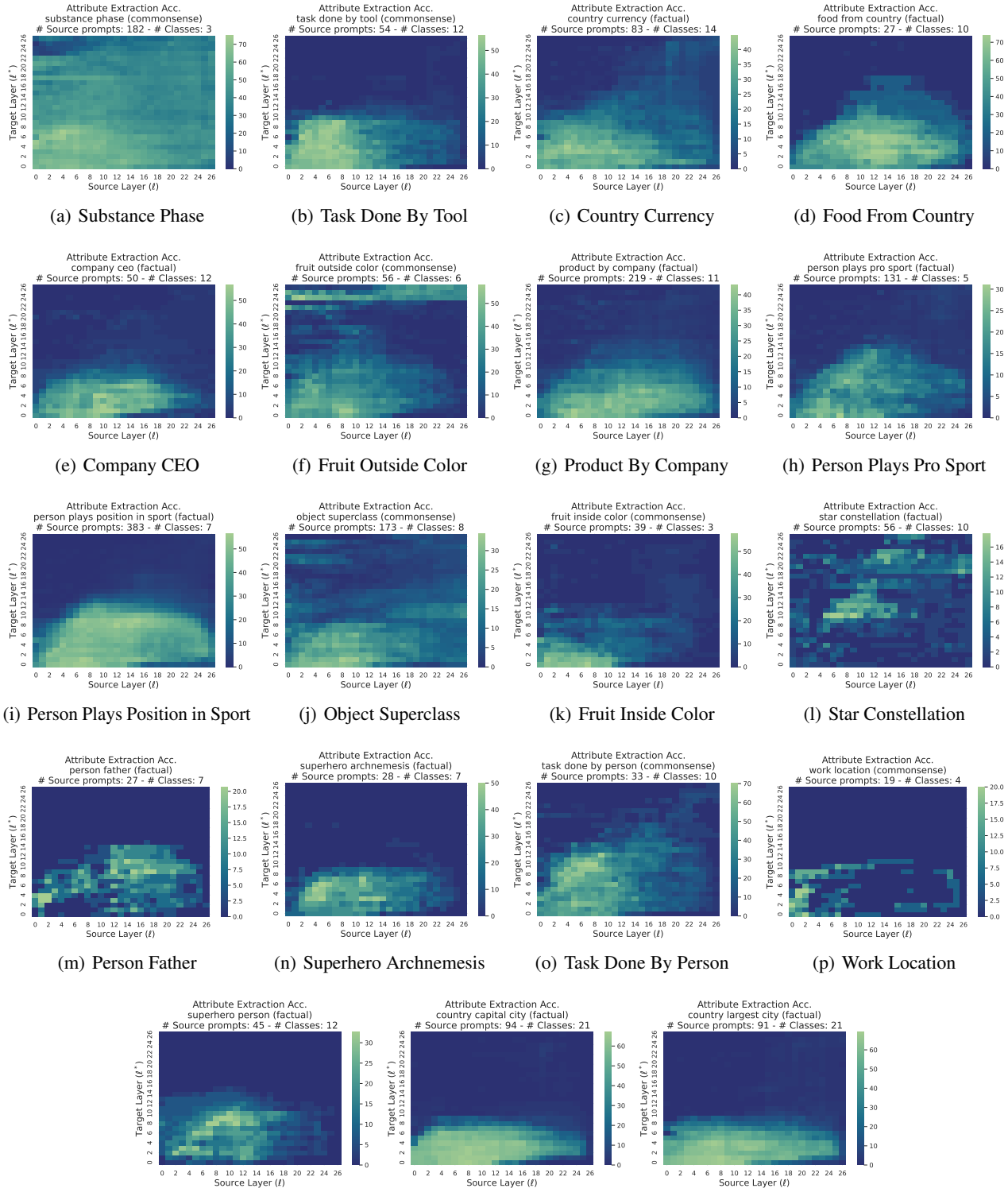
(e) Company CEO (f) Fruit Outside Color (g) Product By Company (h) Person Plays Pro Sport

(i) Person Plays Position in Sport (j) Object Superclass (k) Fruit Inside Color (l) Star Constellation

(m) Person Father (n) Superhero Archnemesis (o) Task Done By Person (p) Work Location

(q) Superhero Person (r) Country Capital City (s) Country Largest City

Figure 6. Feature extraction accuracy with respect to source layer (ℓ) across various factual and commonsense reasoning tasks. The zero-shot feature extraction `Patchscope` works consistently better than the logistic regression probe in early layers, and mostly in mid layers. There is a decline in `Patchscope` accuracy in later ℓ as the source representations shift toward next-token prediction.



(a) Substance Phase (b) Task Done By Tool (c) Country Currency (d) Food From Country

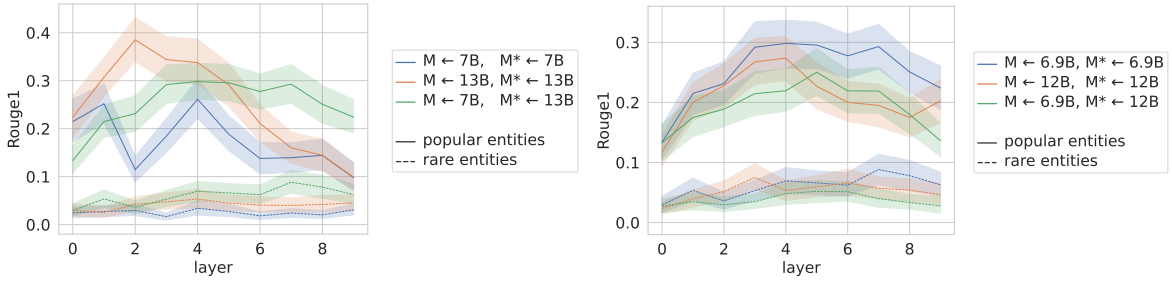
(e) Company CEO (f) Fruit Outside Color (g) Product By Company (h) Person Plays Pro Sport

(i) Person Plays Position in Sport (j) Object Superclass (k) Fruit Inside Color (l) Star Constellation

(m) Person Father (n) Superhero Archnemesis (o) Task Done By Person (p) Work Location

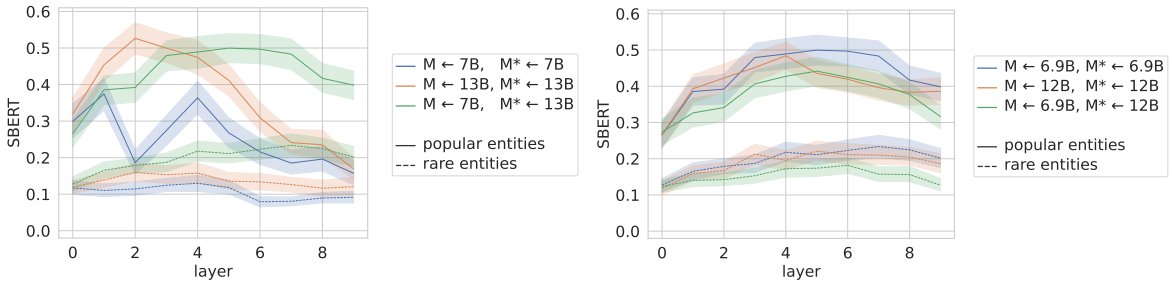
(q) Superhero Person (r) Country Capital City (s) Country Largest City

Figure 7. The interaction between source and target layers in the zero-shot feature extraction Patchscope across various factual and commonsense reasoning tasks. Each cell (ℓ, ℓ^*) in the heatmap shows the attribute extraction success rate where source and target layers are fixed to ℓ and ℓ^*, respectively. Particularly, there is a higher success rate in the lower left quadrants, representing early to mid source and target layer combinations. The right half of the heatmaps shows late source layers, where the source representation has shifted toward next-token prediction, leading to a lower success rate in attribute extraction. The top half of the heatmaps shows late target layers. When the accuracy is a function of more than a single next-token, the placeholder token representation still remains in the early layers, leading to a lower attribute extraction rate.



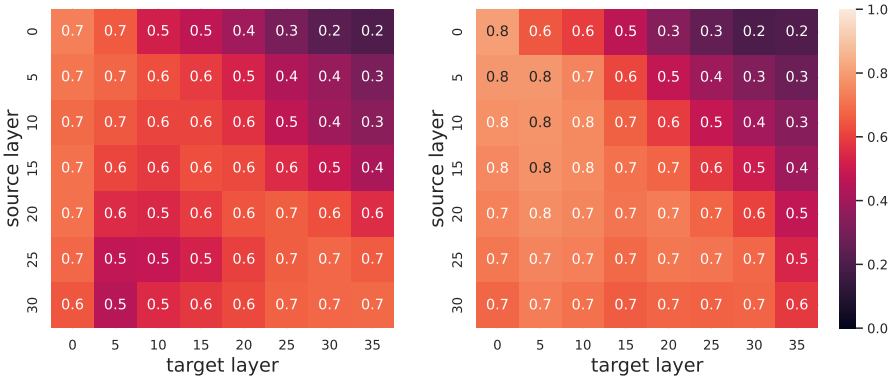
(a) Vicuna (b) Pythia

Figure 8. Rouge1 scores of the generated descriptions against descriptions from Wikipedia.



(a) Vicuna (b) Pythia

Figure 9. SBERT scores of the generated descriptions against descriptions from Wikipedia.

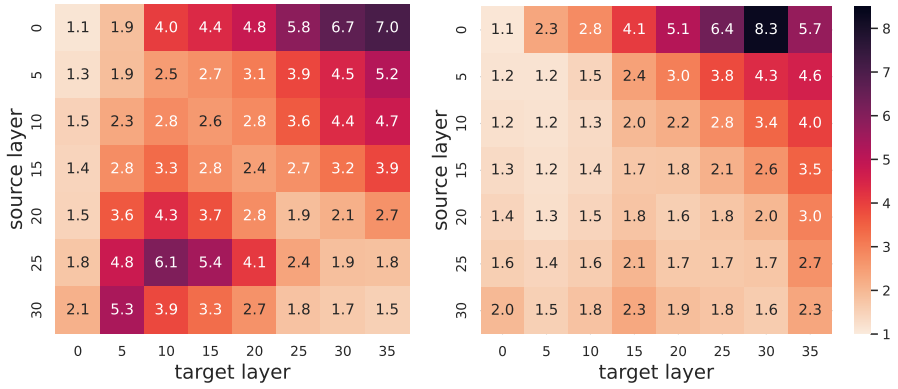


(a) Vicuna: $\mathcal{M} \leftarrow$ 7B, $\mathcal{M}^* \leftarrow$ 13B (b) Pythia: $\mathcal{M} \leftarrow$ 6.9B, $\mathcal{M}^* \leftarrow$ 12B

Figure 10. Precision@1 scores (↑ is better) on next-token prediction estimation in Vicuna and Pythia with cross-model `Patchscopes`

as Vicuna 13B processes "`Will Smith`", it goes from *"Smithsonian Museum"* to the *"Smith rock band"* to the actor and rapper, *"Will Smith"*. However, Pythia 12B starts with *"Smith & Wesson weapon manufacturing company"* before it resolves the entity as the American actor, *"Will Smith"*.

We also observe another phenomenon which we refer to as placeholder contamination. It is the case where the remaining representation of the placeholder entity "`x`" in the early layers interferes with the model's capability in generating descriptions for the patched token. For example, see Vicuna 13B response to "`Paris Hilton`" entity in Tab. 6. First, we see the gradual process of going from *"Rigatoni"* pasta, to *"Hilton Hotels"*, to the socialite *"Paris Hilton"* in layers 1-6. But in layers 7 and onward, the generation seems to describe the placeholder token "`x`"rather than the "`Paris Hilton`" entity: *"Placeholder for a variable or concept"*, *"Variable representing any number of things or concepts"* or *"x is a placeholder"*. For future, we would like to quantify these qualitative observations, study to what extend this contamination can be mitigated with a different placeholder choice, and why some models might be more susceptible to this contamination than others.



(a) Vicuna: $\mathcal{M} \leftarrow$ 7B, $\mathcal{M}^* \leftarrow$ 13B (b) Pythia: $\mathcal{M} \leftarrow$ 6.9B, $\mathcal{M}^* \leftarrow$ 12B

Figure 11. Surprisal scores (↑ is better) on next-token prediction estimation in Vicuna and Pythia cross-model Patchscopes.

E. Additional Results on the Cross-Model Patching Experiment

Fig. 10 depicts the Precision@1 scores for different combinations of source-target layers, showing that patching with a simple affine Patchscope proves to be effective with precision of up to 0.7 and 0.8 for Vicuna and Pythia, respectively. Specifically, patching representations to an early layer of the larger model seems to be the most effective. Furthermore, it appears that there is a subtle matching among some layers of the models, with the diagonal consistently exhibiting higher values. As shown in Fig. 11, similar trends are observed in the surprisal results for both models. Overall, these results show that, when $\mathcal{M}^*$ and $\mathcal{M}$ are from the same model family, it is possible to leverage $\mathcal{M}^*$ for decoding information from the representations of $\mathcal{M}$. Notably, our findings extend observations by Csiszárik et al. (2021) from patching representations across deep convolutional neural networks with the same architecture but different initializations.

F. More Details on the Multi-Hop Reasoning Experiment

Data Building on Hernandez et al. (2023b), we systematically generate all valid multi-hop factual and commonsense reasoning queries where the object ω_1 in the first triplet τ_1 is equal to the subject σ_2 in the second triplet τ_2. This process yields 1,104 multi-hop reasoning samples. We then filter examples to the ones where the model accurately represents both triplets independently. Tab. 8 summarizes information about the samples where Vicuna (13B) correctly represents each reasoning step. Out of 1,104 samples that require two steps of commonsense or factual reasoning, 46 satisfy the above criteria, with 8 unique relation combinations.

More Detailed Results Fig. 12 shows the interaction between ℓ and ℓ^* and how it affects the success rate. Patching representations from most source layers ℓ into early-to-mid ℓ^* (6-16) is the most effective in making the right prediction. Our interpretation is that patching into late ℓ^* is not effective because ρ_2 has already been processed and the result has been copied to the last position, therefore it cannot incorporate the proxy of σ_2. We also observe that $\ell^* \leq \ell$ is more successful on average.

G. Compute Resources

All experiments were conducted using A100 80GB GPUs, with the exception of experiments with GPT-J (Wang & Komatsuzaki, 2021) where we used A100 40GB GPUs.

Table 6. Additional qualitative examples illustrating entity resolution in **Vicuna 13B**. The expressive generations show that as we go through the layers, more tokens from the context get integrated into the current representation, referred to as "Tokens Covered". Here, $\mathcal{M} \leftarrow$ Vicuna 13B, $\mathcal{M}^* \leftarrow \mathcal{M}, \ell^* \leftarrow \ell$.

S	Tokens Covered	ℓ	Generation	Notes/Entity Described by the Generation
"Will Smith"	"Smith"	1	sonian: National museum and research complex in Washington, D.C.	**Smith**sonian Museum
	"Smith"	2	sonas: American rock band	An amalgam of **Smith** band and SONA band
	"Will Smith"	3	: American actor and rapper	**Will Smith**
"Queen Victoria"	"Victoria"	1	s Secret: American multinational women's lingerie, beauty and body-care products company	**Victoria**'s Secret
	"Victoria"	2	s Secret: American lingerie and beauty products retailer	**Victoria**'s Secret
	"Queen Victoria"	3	: Queen of the United Kingdom of Great Britain and Ireland from 1837 to 1901	**Queen Victoria**
	"Queen Victoria"	4-8	: Queen of the United Kingdom of Great Britain and Ireland from 1837 until her death in 1901	**Queen Victoria**
	"Queen Victoria"	9	: Queen of England (reigned from 1837-1901)	**Queen Victoria**
	"Queen Victoria"	10	: 19th century British queen who ruled from 1837-1901	**Queen Victoria**
"Paris Hilton"	"ton"	1	y: a type of pasta	Riga**ton**i/Riga**ton**y/Riga**ton**é
	"Hilton"	2	Hotels: American multinational hospitality company	**Hilton** Hotel
	"Hilton"	3	: Hotel chain founded by Conrad Hilton	**Hilton** Hotel
	"Hilton"	4	: Hotel and resorts company	**Hilton** Hotel
	"Paris Hilton"	5-6	: Socialite and television personality	**Paris Hilton**
	N/A	7	: Placeholder for a variable or concept	Placeholder contamination
	N/A	8-9	: Variable representing any number of things or concepts	Placeholder contamination
	N/A	10	: x is a placeholder	Placeholder contamination
"James Bond"	"Bond"	1	: Unit of measurement for the strength of a liquid	**Bond** Unit
	"James Bond"	2	: James Bond, the fictional British secret agent created by novelist Ian Fleming and portrayed by actor Daniel Craig in the 2006 film "Casino Royale,"	**James Bond**
	"James Bond"	3-5	: Fictional British secret agent	**James Bond**
	"James Bond"	6	: Code name for fictional British secret agent James Bond	**James Bond**
	"James Bond"	7	: Code name for a fictional British secret agent created by novelist Ian Fleming	**James Bond**
	"James Bond"	8	: Subject of a fictional British secret agent, created by novelist Ian Fleming and portrayed in a series of films.	**James Bond**
	James Bond	9	: Subject of the 007 novels and films	**James Bond**
	N/A	10	: Variable representing the number of the film in the James Bond series	Placeholder contamination
"Dwayne Johnson"	"Johnson"	1-2	& Johnson: American multinational corporation in the fields of pharmaceuticals, medical devices, and consumer packaged goods	**Johnson** & Johnson
	"Dwayne Johnson"	3-4	: American actor and professional wrestler, also known as "The Rock"	**Dwayne Johnson**
	"Dwayne Johnson"	5	: Also known as "The Rock," American actor and professional wrestler	**Dwayne Johnson**
	"Dwayne Johnson"	6	: Former professional wrestler and actor	**Dwayne Johnson**
	N/A	7	: Abbreviation for the United States	Placeholder contamination
	N/A	8	: Variable representing any number of things or concepts	Placeholder contamination
	N/A	9	: Variable representing any number of other options.	Placeholder contamination
	N/A	10	: Variable representing any number of other options or possibilities.	Placeholder contamination

Table 7. Additional qualitative examples illustrating entity resolution in **Pythia 12B**. The expressive generations show that as we go through the layers, more tokens from the context get integrated into the current representation, referred to as "Tokens Covered". Here, $\mathcal{M} \leftarrow$ Pythia 12B, $\mathcal{M}^* \leftarrow \mathcal{M}, \ell^* \leftarrow \ell$.

S	Tokens Covered	ℓ	Generation	Notes/Entity Described by the Generation
"Will Smith"	"Smith"	1	& Wesson: American firearms manufacturer	**Smith** & Wesson
	"Will Smith"	2	: American actor	**Will Smith**
"Alexander the Great"	"Great"	1	Britain: Country in the European Union	**Great** Britain
	"the Great"	2	Wall Street Crash of 1929: Financial crisis in the United States	**the Great** Depression
	"the Great"	3	Wall Street Bubble: The Great Depression	**the Great** Depression
	"the Great"	4	Wall Street: Wall Street in New York City	Wall Street (related to the Great Depression)
	"Alexander the Great"	5	: Ancient Greek ruler, and the first to rule all of the then known world	**Alexander the Great**
"George Washington" $S \leftarrow$	"Washington"	1	: Capital of the United States	**Washington** D.C.
	"Washington"	2	: American capital city	**Washington** D.C.
	"George Washington"	3	: American president	**George Washington**
	"George Washington"	4-6	: American revolutionary	**George Washington**
	"George Washington"	7	: American president	**George Washington**
	"George Washington"	8	: George Washington	**George Washington**
	"George Washington"	9	: George Washington, the first President of the United States	**George Washington**
	"George Washington"	10	: George Washington, Thomas Jefferson, Abraham Lincoln, and Theodore Roosevelt: American presidents	**George Washington**
"Titanic"	"c"	1	loud: Apple's cloud-based storage service	**c**loud
	"Titanic"	2	: Sinking ocean liner	**Titanic**
	"Titanic"	3	: The ship that sank in the Atlantic Ocean in 1912	**Titanic**
	"Titanic"	4-6	: British passenger ship	**Titanic**
	"Titanic"	7-9	: The ship that sank	**Titanic**
	"Titanic"	10	: Titanic, Titanic: British luxury passenger liner	**Titanic**
"Saturday Night Live"	"Live"	1	Nation: American concert promoter	**Live** Nation
	"Live"	2	Nation: American television network	**Live** Nation
	"Saturday Night Live"	3	: American television program	**Saturday Night Live**
	"Saturday Night Live"	4-10	: American television comedy show	**Saturday Night Live**
"Nineteen Eighty-Four"	"Four"	1	Seasons: American hotel chain	**Four** Seasons Hotel
	"Four"	2	Seasons: The four seasons of the year	**Four** Seasons (of the year)
	"ighty-Four"	3	Lions: Chinese professional football club	Cangzhou **Mighty** Lions F.C.
	"Nineteen Eighty-Four"	4	: Number of the novel 1984 by George Orwell	**Nineteen Eighty-Four** (Novel)
	"Nineteen Eighty-Four"	5	: George Orwell's novel, and the 1984 film.	**Nineteen Eighty-Four** (Novel & Film)
	"Nineteen Eighty-Four"	6	: George Orwell's novel	**Nineteen Eighty-Four** (Novel)
	"Nineteen Eighty-Four"	7	: George Orwell: British novelist	**Nineteen Eighty-Four** (Novel)
	"Nineteen Eighty-Four"	8	: 1984 novel by George Orwell	**Nineteen Eighty-Four** (Novel)
	"Nineteen Eighty-Four"	9-10	: 1984 novel by George Orwell, and the 1984 film adaptation directed by Michael Radford.	**Nineteen Eighty-Four** (Novel & Film)

Table 8. Sample statistics for the multi-hop reasoning experiment where $\mathcal{M}$ correctly represents both τ_1 and τ_2.

ρ_1	ρ_2	# Samples
Company CEO	Person Father	4
Food from Country	Country Capital City	10
Food from Country	Country Currency	3
Food from Country	Country Language	9
Food from Country	Country Largest City	11
Person Father	Person Father	1
Person Father	Person Mother	1
Product by Company	Company CEO	7
Total		46

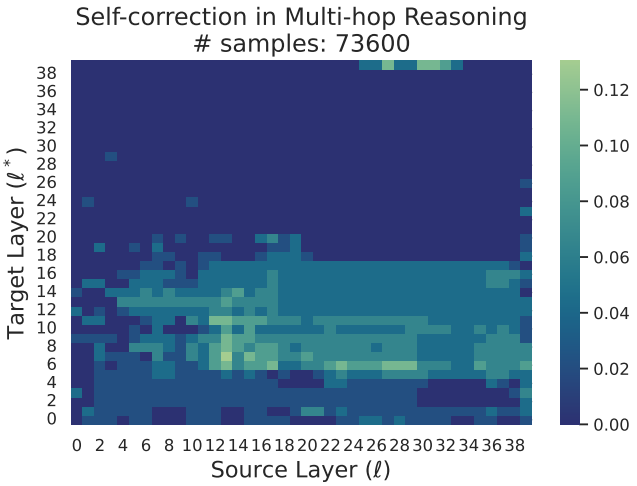


Figure 12. The interaction between source (ℓ) and target (ℓ^*) layers in correcting multi-hop reasoning errors. The majority of success cases correspond to early-to-mid ℓ^*. We observe higher cumulative success rate in the lower right triangle which corresponds to $\ell^* \leq \ell$.