THE UNIVERSITY OF CHICAGO

TOWARDS INTERPRETING LANGUAGE MODELS: A CASE STUDY IN MULTI-HOP REASONING

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

BY
MANSI SAKARVADIA

CHICAGO, ILLINOIS
APRIL 5, 2024

I would like to thank my advisors Dr. Ian Foster and Dr. Kyle Chard for their support, encouragement, and guidance in writing this research. They gave me the freedom to explore and pursue creative ideas, and for that I am very grateful.

I would like to acknowledge my amazing collaborators: Aswathy Ajith, Arham Khan, Daniel Grzenda, Dr. Nathaniel Hudson, and Dr. André Bauer. I learn so much from each of you.

My sincere thanks to all of my fellow lab mates at Globus Labs. They have made my graduate journey a lot of fun.

Most importantly, I would like to thank my family for their unwavering support and encouragement during my graduate studies and my lovely friends who cheer me on and celebrate my accomplishments every step of the way.

"Nothing in life is to be feared, it is only to be understood."

Dr. Marie Curie

TABLE OF CONTENTS

LIST OF FIGURES vii

LIST OF TABLES ix

ACKNOWLEDGMENTS x

ABSTRACT xi

1 INTRODUCTION 1

2 RELATED WORK 5
2.1 Probing Models 5
2.2 Activation Engineering 6
2.3 Model Editing 7
2.4 Circuit Discovery 8
2.5 Knowledge Extraction 9
2.5.1 Memorization 11
2.6 Language Model Reasoning 11
2.7 Retrieval Augmented Generation 13

3 MEMORY INJECTIONS 15
3.1 Introduction 15
3.2 Background & Notation 17
3.2.1 Multi-hop vs. single-hop prompts 18
3.2.2 Transformer Architecture 18
3.3 Experimental Overview 20
3.3.1 Dataset Descriptions 21
3.3.2 Model Description 23
3.3.3 Tools & System Setup 23
3.4 Proposed Methods 23
3.4.1 Interpreting Attention Heads 24
3.4.2 Memory Injections to Correct Failures 25
3.5 Results and Discussion 28
3.5.1 Curated Memory Injections 29
3.5.2 Random Memory Injections 30
3.5.3 Memory Injections for Parts of Speech 32
3.6 Additional Memory Encoding Styles 36
3.6.1 Memory Encoding Styles 36
3.6.2 Encoding Style FLOP Counts 37
3.6.3 Additional Model Descriptions 38
3.6.4 Memory Encoding Style Experiments 39
3.7 Related Work 43

3.8 Conclusions and Future Directions 44
3.9 Broader Impacts & Ethics 45

4 ATTENTION LENS 46
4.1 Introduction 46
4.2 Training Lenses 48
4.3 Attention Lens Applications 49
4.4 Evaluating Lenses 51
4.5 Conclusion 52

5 SUMMARY & FUTURE WORKS 53

REFERENCES 55

LIST OF FIGURES

3.1 A multi-hop prompt vs. two analogous single-hop prompts. The outputs are from GPT2-Small. 16

3.2 **Diagram of language model reasoning.** Highest ranked attention outputs of GPT2-Small at layer $\ell = 9$, head $h = 8$ when projected into vocabulary space (via the GPT2-Small embedding matrix) for a single-hop prompt (green) and its multi-hop counterpart (red). 25

3.3 **Memory injection.** Injecting memory "The Great Barrier Reef" into GPT2-Small hidden activations at layer $\ell = 9$, head 8, $\tau = 4$. 28

3.4 **Curated memory injections.** From left to right: GPT2-Small + *Hand*, GPT2-Large + *Hand*, GPT2-Small + *2WMH*, GPT2-Large + *2WMH*. Each cell in each heatmap is the average percent difference between the pre- and post-injection next token predictions for multi-hop prompts. Green cells denote a positive percent difference (i.e., correct prediction is more likely), while red cells denote a negative percent difference (i.e., correct prediction is less likely). When computing the averages for each (ℓ, τ) pair we exclude outliers not within ± 2 standard deviations from the mean. 29

3.5 **Part of speech memory injections.** This figure shows the average effect of memory injections from various parts of speech as a function of layer ℓ (top row) and magnitude τ (bottom row). The standard deviation scaled by 10% is pictured across magnitudes (top row) and layers (bottom row). 32

3.6 **GPT2-Large, *2WMH* dataset.** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ± 2 standard deviations from the mean across various parts of speech. 34

3.7 **GPT2-Large, *Hand* dataset.** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ± 2 standard deviations from the mean across various parts of speech. 34

3.8 **GPT2-Small, *2WMH* dataset.** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ± 2 standard deviations from the mean across various parts of speech. 35

3.9 **GPT2-Small, *Hand* dataset.** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ± 2 standard deviations from the mean across various parts of speech. 35

3.10 **GPT2-Small** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ± 2 standard deviations from the mean across various memory encoding styles (*Embed*, *Unembed*, *Layer-wise*) and datasets (*Hand*, *2WMH*). 40

3.11 **GPT2-Large** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ±2 standard deviations from the mean across various memory encoding styles (*Embed*, *Unembed*, *Layer-wise*) and datasets (*Hand*, *2WMH*). 40
3.12 **GPT2-XL** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ±2 standard deviations from the mean across various memory encoding styles (*Embed*, *Unembed*, *Layer-wise*) and datasets (*Hand*, *2WMH*). 41
3.13 **GPT-Neo (125M)** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ±2 standard deviations from the mean across various memory encoding styles (*Embed*, *Unembed*, *Layer-wise*) and datasets (*Hand*, *2WMH*). 41
3.14 **GPT-Neo (1.3B)** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ±2 standard deviations from the mean across various memory encoding styles (*Embed*, *Unembed*, *Layer-wise*) and datasets (*Hand*, *2WMH*). 42
3.15 **GPT-Neo (2.7B)** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ±2 standard deviations from the mean across various memory encoding styles (*Embed*, *Unembed*, *Layer-wise*) and datasets (*Hand*, *2WMH*). 42
3.16 **GPT-J** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ±2 standard deviations from the mean across various memory encoding styles (*Embed*, *Unembed*, *Layer-wise*) and datasets (*Hand*, *2WMH*). 43

4.1 `Attention Lens.` Comparing the outputs of layer $\ell = 10$, head $h = 11$ using *Attention Lens* vs. the model's umembedding matrix in GPT2-Small. 47

LIST OF TABLES

3.1 Properties of the datasets used in our work. *Size*: Number of prompts. *Answer prob.*: Average model probability model for expected next token. *Surprisal*: Average model surprisal value for expected next token ($surprisal \triangleq -\log(p)$ where p is a probability). *Prompt len.*: Average tokenized length of prompt. 22

3.2 **Example prompts.** Single/multi-hop prompt pairs from *Hand* and *2WMH* datasets. 22

3.3 **Example of attention head outputs** from GPT2-Small for *Hand*. 26

3.4 **Examples of memory injections.** Injecting memories with $\tau = 4, \ell = 9$ into GPT2-Small. 30

3.5 **Curated vs. random memory injections.** Table shows the (ℓ, τ) pairs for the best token injections, along with the *average percent difference* (excluding outliers $>\pm 2$ standard deviations from the mean) between pre- and post-injection expected next token predictions for multi-hop prompts. Each random injection column indicates 40 random injections from [Adjectives, Adverbs, Conjunctions, Nouns, Verbs, Top 5050] at the ideal (ℓ, τ). 31

3.6 **Model Characteristics**. d_{model} is hidden dimension of model. d_{vocab} is size of model's vocabulary. # layers is number of layers in model. 38

3.7 **Encoding styles vs. FLOPs**. The Avg. Percent Difference column reports the mean of the *average percent different* of the most performant (layer, magnitude) injection pairs across all (model, dataset) combinations for various memory encoding styles. The *average percent difference* (excluding outliers $>\pm 2$ standard deviations from the mean) is computed between the pre- and post-injection expected next token predictions for multi-hop prompts. The Avg. FLOP column reports the average number of float point operations needed for the corresponding encoding style calculated in accordance to section 3.6.2. 39

4.1 A comparison of `Attention Lens` with Logit Lens and Tuned Lens. 48

ACKNOWLEDGMENTS

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Department of Energy Computational Science Graduate Fellowship under Award Number DE-SC0023112. This work is also supported in part by the U.S. Department of Energy under Contract DE-AC02-06CH11357.

ABSTRACT

Answering multi-hop reasoning questions requires retrieving and synthesizing information from diverse sources. Language models (LMs) struggle to perform such reasoning consistently. We propose an approach to pinpoint and rectify multi-hop reasoning failures through targeted *memory injections* on LM attention heads. First, we analyze the per-layer activations of GPT-2 models in response to single- and multi-hop prompts. We then propose a mechanism that allows users to inject relevant prompt-specific information, which we refer to as "memories," at critical LM locations during inference. By thus enabling the LM to incorporate additional relevant information during inference, we enhance the quality of multi-hop prompt completions. We empirically show that a simple, efficient, and targeted memory injection into a key attention layer often increases the probability of the desired next token in multi-hop tasks, by up to 424%. We observe that small subsets of attention heads can significantly impact the model prediction during multi-hop reasoning. To more faithfully interpret these heads, we develop `Attention Lens`: an open source tool that translates the outputs of attention heads into vocabulary tokens via learned transformations called *lenses*. We demonstrate the use of lenses to reveal how a model arrives at its answer and use them to localize sources of model failures such as in the case of biased and malicious language generation.

CHAPTER 1

INTRODUCTION

Despite recent widespread adoption of neural Language Models (LMs) [Vaswani et al., 2017, Brown et al., 2020] in chat-based applications [OpenAI, 2022], the mechanisms by which LMs acquire knowledge during training and recall knowledge to form predictions at inference time are not well understood. This complicates the safe deployment of LMs in consumer and scientific pipelines [Gaudin, 2023, Yun et al., 2023, Hardalov et al., 2018, Jablonka et al., 2023, inter alia] as LM behavior can be unpredictable. For example, LMs are capable of exhibiting harmful behaviors including displaying bias, regurgitating private information, hallucinating, producing offensive language, and producing malicious outputs due to adversarial training [Nadeem et al., 2020, Winograd, 2023, Zhang et al., 2023, Bender et al., 2021, Kandpal et al., 2023b]. Mitigating these harmful behaviors is limited by our lack of understanding of how LMs work. To ensure success and safety of LM-based applications, better interpretability techniques must be developed to understand how models develop behaviors. In this work, we focus on developing better interpretability techniques to understand how models recall knowledge during inference.

We study the case of LMs attempting to perform multi-hop reasoning. Multi-hop reasoning is the task of answering a prompt that contains references to an entity that is never explicitly named (see Fig. 3.1). Many modern LMs struggle to consistently perform multi-hop reasoning [Arkoudas, 2023, Guo et al., 2023, Blair-Stanek et al., 2023]. We develop a method to localize multi-hop reasoning failures to specific attention heads within a model, inspect what terms an attention heads is outputting via a tool called `Attention Lens`, and an efficiently enhance multi-hop reasoning abilities during inference via our technique called "memory injections". Our interpretability-driven techniques can be easily adapted to localize additional sets of LM behavior within model weights, are computationally efficient, and overcome the limitations of existing model behavior corrective techniques.

Popular techniques to correct model behavior cannot guarantee improved performance on a target task without negatively affecting the model's performance on unrelated tasks. This is because researchers do not have the tools to reliably pinpoint the source of model failure within the weight space, so they attempt to apply general and broad corrective techniques to entire model architectures. Examples of popular corrective techniques include fine-tuning [Dodge et al., 2020], parameter-efficient fine-tuning [Mangrulkar et al., 2022], human-feedback reinforcement learning [Christiano et al., 2017], retraining [Wu et al., 2020], model editing [Wang et al., 2023b, Zhang et al., 2024], and unlearning [Bourtoule et al., 2021]. Many of these techniques can have unintended consequences. For example, fine-tuning can distort features learned during pre-training, induce catastrophic forgetting, and compromise model safety [Kumar et al., 2022, Kirkpatrick et al., 2017, Kemker et al., 2018, Qi et al., 2023]. Model editing can have unintended effects on other knowledge originally embedded in the model's weights [Cohen et al., 2023a]. Machine unlearning often does not provide guarantees about what a model knows; Shi et al. [2024], Patil et al. [2023] showed that sensitive information could still be recovered from an edited model, even if unlearning strategies were applied to the model to remove the sensitive information. Through our work, we argue that the effectiveness of these corrective techniques could be enhanced via a more robust understanding of *how* knowledge is embedded in models' weights. For example, a more faithful interpretation of a model's weight space could enable the application of model corrective techniques to subsets of model weights rather than entire model architectures which may alleviate some of the unintended consequences of current techniques.

Corrective techniques are also out of reach for many individuals and organizations due to high computational costs. Model sizes are rapidly increasing [Hestness et al., 2017, Hoffmann et al., 2022] which means that applying any gradient-based procedure (e.g., training, fine-tuning) to models at billion parameter scales with internet-scale datasets requires vast amounts of computational resources. For example, the 70 billion parameter LLama2 model

was trained at Meta's Research Super Cluster [Lee and Sengupta] and their internal production clusters; Llama2 70B's pre-training took $1,720,320$ GPU hours on Nvidia A100-80GB GPUs [Touvron et al., 2023]. Few organization have these types of resources to pre-train models. After training LLama2 70B, Touvron et al. [2023] reports that "Llama 2-Chat is the result of several months of research and iterative applications of alignment techniques, including both instruction tuning and RLHF, requiring significant computational and annotation resources." Ultimately, this report alludes to the reality that assessing and correcting model behavior at scale (with current tools) can be prohibitively expensive for individuals and smaller organizations. Better interpretability techniques could alleviate some of the computational cost in the model assessment and alignment workflows. For example, techniques that allow ML practitioners to quickly diagnose sources of model failure in weight space and apply targeted remedies to subsets of weights have lower computational resource requirements compared to corrective techniques that must be applied iteratively to full model architectures.

To summarize, better LM interpretability techniques could enable strides in many open problems:

1. Limited effectiveness of current model behavior corrective techniques.

2. High computational cost of evaluating and correcting models.

To address these problems, we need to develop interpretability tools that allow us to further localize sources of model behavior within model architectures. There is evidence that much model behavior is localizable [Frankle and Carbin, 2018, Goldowsky-Dill et al., 2023, Wardat et al., 2021, Maini et al., 2023]. Behavior localization will enable the application of corrective techniques to a model in a more exact and targeted manner, thus alleviating both high computational costs and harmful side effects associated with current corrective techniques such as full model fine-tuning. Ultimately, better localization techniques will

greatly enhance model transparency and boost understanding of how and why models succeed and fail in various scenarios.

A central challenge to our ability to localize model behavior is a lack of understanding as to how a model arrives at its final prediction: *How are humans supposed to interpret the computations in intermediate model layers?* Additionally, we have a limited arsenal of model corrective techniques and could benefit from more tools in our tool box; especially gradient-free methods as these would be more accessible due to lower computational cost.

In this work, we address these shortcomings by making the following contributions:

1. We develop a gradient-free method, "memory injections", to enhance model behavior at inference time in a human understandable format via intervening on hidden activations.

2. We show how memory injections can be used to localize model failure on multi-hop reasoning tasks, and even correct model performance without modifying weights, thus eliminating concerns of hurting model behavior in unrelated tasks.

3. We develop a method to interpret the outputs of attention heads in human-understandable formats.

4. We develop a software framework, `Attention Lens`, to support training of probes into individual attention heads to better characterize their role during inference.

CHAPTER 2

RELATED WORK

We review interpretability tools such as probing, activation engineering, model editing, circuit discovery, and knowledge extraction. We also review recent advances in the study of language model reasoning capabilities and retrieval augmented generation.

2.1 Probing Models

Probing is a class of interpretability methods that attempt the decode the contents/functions encoded by neural network weights. Probing does this by directly mapping subsets of weight activations into human-understandable domains. Since activations are directly related to model inputs, probes allow researchers to causally draw connections between model inputs and probe outputs. Therefore, researchers may be able to use probes to localize sources of model behavior to a specific subset of model weights.

Probes can be designed flexibly to suite the task at hand and are typically optimized using gradient based techniques like stochastic gradient descent [Ruder, 2016]. There are two main axes of freedom in probe design:

- **Architecture:** Probe architectures are informed by the types of insight a researcher requires from a probe. For example, probes can be designed at varying level's of model architecture (e.g. weight-level, layer-level, attention head-level). Additionally, probes can be linear or non-linear. Certain model behaviors may be linearly decodable [Alain and Bengio, 2016] while others may need non-linear probes to decode [White et al., 2021]. Some works have even found that the manner in which a decoding task is defined can allow a probe to transition from non-linear [Li et al., 2022] to linear [Nanda et al., 2023c].

- **Training dataset:** since probes are trained to perform a mapping between a model's activation and a desired domain, the training data a probe sees will govern its behavior. This training data must elicit all of the behaviors the researcher is attempting to study.

There are many use cases for trained model probes. Ettinger et al. [2016] introduces using classifier probes to understand semantic information in sentence representations. Probes can be trained to decode a hidden model representation into vocabulary, often with the goal of attempting to understand how each model layer informs how the model arrives at a final token prediction [nostalgebraist, 2021, Belrose et al., 2023, Pal et al., 2023, Katz et al., 2024]. Li et al. [2022] trained probes to understand a model's internal board representation when predicting the next best move in the board game Othello. Kim et al. [2019] explored the effect of pre-training on the model's learned representations of "function words" via trained probes. Aina and Linzen [2021] used probes to quantify model uncertainty in its completions of ambiguous prompts.

A limitation of probes as a diagnostic tool is that it is not obvious if the probes are are correlational or causal tools for attempting to understand a model's internal representations [Belinkov, 2022]. To combat this limitation, researchers can further validate the efficacy of probes by using the probes to guide activation engineering and observe if downstream model performance is affected as they would expect. For example, Li et al. [2022] shows that by using probes to guide the editing of Othello-GPT's representation of the board state, they could alter the model's final next move prediction as expected; this further validated that the probes were faithfully decoding information as it was known to the model. See section 2.2 for additional techniques about how to engineer activations.

2.2 Activation Engineering

Activation Engineering is a class of interpretability method that allows researchers to decode the functions of model components, by modifying their respective output activa-

tion values and observing the downstream effect. Like probing 2.1, the goal of activation engineering can be to allow researchers to attribute model behavior back to specific model components. Additionally, activation engineering can also be used to directly influence model behavior at inference time.

Some examples of applications of activation engineering are:

- Vig et al. [2020] introduced "causal mediation analysis": a method to understand which components are a model are responsible for specific behaviors in language modeling; they apply causal mediation analysis to investigate which components of a model are responsible for gender bias.

- Sun et al. [2021] demonstrates that activations of neural networks can be used to identify in-distribution and out-of-distribution model inputs in vision tasks. Djurisic et al. [2022] builds on this concept by both pruning and modifying late layer model activations for out-of-distribution detection.

- Meng et al. [2022a] used "causal mediation analysis" [Vig et al., 2020] as a method for localizing knowledge within model weights by comparing the activations of model forward passes over two different inputs.

- Turner et al. [2023] proposes a method to add vectors that encode human-understandable semantic information directly to the activations of LMs to steer their outputs.

- Fort [2023] showed how to adversarially engineer activations to have harmful downstream effects on LM prompt completions.

2.3 Model Editing

Model editing aims to change specific facts, associations, or information embedded in an LM outside of the constraints of traditional model training. Model editing requires the ability

to localize learned information within subsets of the weight space and employs efficient and targeted methods to change this information while mitigating its effects of other information also embedded in the weight space. Model editing can be used to remove or alter private information, incorrect information, outdated information, biased information, and harmful information stored within model weights [Wu et al., 2023, Yan et al., 2024, Chen et al., 2023, Wang et al., 2024]. Model editing can enable machine learning models to more exactly reflect human knowledge, without the massive overhead cost of typical model pre-training/fine-tuning. Zhu et al. [2020] proposes an approach to modify specific learned facts encoded withing a LM's weights, while preserving model performance on other previously learned knowledge via a constrained optimization problem. Dai et al. [2022a] developed attribution methods to decipher which neurons are responsible for specific facts within languages models and developed methods to manipulate these neurons to edit a given fact. Cao et al. [2021], Mitchell et al. [2022b] both propose hypernetwork based approaches to edit facts within models. Hypernetworks are additional networks that are trained to predict which weights are responsible for a given fact and how to modify the weights of a given neural network to better represent the desired knowledge. Meng et al. [2022a] proposed Rank-One Model Editing (ROME): by interpreting multi-layer perceptrons as key-values stores, ROME is able to replace specific keys-value pairs to override old or establish new knowledge associations in the model.

2.4 Circuit Discovery

Circuits are sparse subsets of neural network weights that are responsible for (sub)sets of model behavior. Interpreting neural networks as circuits is useful as it allows researchers to localize sources of model behavior and may even help them better understand how stochastic training processes compress knowledge and skill into weights.

It was not always obvious that it was tractable to attribute model behavior to specific

model components as these models are increasing massive (e.g. million, billion, trillion parameter scales). However, much research has shown that neural networks are very sparse: a small subset of weights are often responsible for much of a model's behaviors. For example, Frankle and Carbin [2018] showed that neural networks contain "winning lottery tickets": sparse sub-networks within trained NNs that are nearly as performant as the original dense network. Follow up work, by Amid et al. [2022] further developed methods to extract performant sub-networks from randomly initialized dense NNs. Results about the sparsity of models held across neural network architectures [Han et al., 2017, Chen et al., 2020, Behnke and Heafield, 2020].

Further work attempted to take this work a step further by designing methods to attribute more specific model behaviors to individual model components. Elhage et al. [2021] conducted a detailed analysis of the types of circuits that appear in zero, one, and two layer transformers. Chintam et al. [2023] identified components of transformer-based LMs for gender bias. Nanda et al. [2023b] reverse engineers the algorithm implemented by a one layer transformer for modular addition. While the types of behaviors exhibited by a given model can be large and diverse, the workflow to discover circuits share many similarities. Conmy et al. [2023] outlines a typical circuit discovery workflow for many ML interpretability pipelines and proposes a framework automate the workflow.

2.5 Knowledge Extraction

Knowledge extraction in language modeling is the practice of discovering what information is embedded in a LM's weights. The practice of knowledge extraction has grown in popularity as it became better known that LMs can be treated as successful knowledge stores. For example, Roberts et al. [2020] showed that by fine-tuning an LM on question answering tasks, the LM was able to successfully perform question-answering in a closed-book setting. This finding implied that models were good at storing knowledge during training and

retrieving knowledge during inference time. In the context of language modeling, knowledge extraction is useful because it illuminates what a model knows well and what information it might be lacking. This enables ML developers to stage the appropriate interventions to improve model performance on desired tasks (e.g., further fine-tuning, knowledge editing). A simple method to extract model knowledge is to prompt the model and observe the outputs. In a closed-loop model prompting scenario the model would have to rely on its internal knowledge store in order to appropriately respond to a prompt. Therefore, based on the model outputs for any given prompt, the prompter would be able to infer what information the model is storing in its weights. Petroni et al. [2019], Jiang et al. [2020] both design prompting strategies to elucidate what knowledge is contained in LMs. The immediate shortcoming with prompting a model to elicit information is that models will only output information that it deems relevant to the prompt. Therefore, vanilla prompting knowledge extraction strategies may fail to uncover a model's full breadth of knowledge. It is often challenging to come up with a comprehensive prompting scheme to enable a model to exercise its entire knowledge store. To combat this, researchers have also devised more rigorous knowledge extraction techniques:

- Cohen et al. [2023b] proposes a strategy to extract a knowledge-graph (KG) of facts from a LM: given a seed entity, they "crawl" the KG via prompts that are designed for both precision and recall.

- Zhong et al. [2021] demonstrated that by training probes, rather than using discrete prompts, to illicit knowledge from LMs, they were able to tighten the lower bound on knowledge extraction benchmarks like LAMA [Petroni et al., 2019].

- Elazar et al. [2021] proposed a novel framework to assess if facts known to a LM are generalizable. By using a paraphrasing technique, they show that models are often inconsistent in reporting facts thus implying they do not contain generalizable facts.

2.5.1 Memorization

Memorization is an undesirable phenomena observed in LMs: models can be prompted to output their training data verbatim [Feldman and Zhang, 2020]. Eliciting memorized data from a LM can be viewed as a subset of general purpose knowledge extraction tasks. However, while many forms of knowledge extraction are for benign purposes, such as gauging and improving a model's knowledge base, memorized data extraction can have harmful consequences. Memorized data can contain sensitive and/or private data that should not be recoverable by a model prompter. Dataset extraction attacks aim to prompt a model in a manner such that the model regurgitates its training data. Carlini et al. [2021] proposed one of the first training data extraction attacks from LMs. Nasr et al. [2023] designed a black-box model prompting scheme to extract training data from LMs. Many works have attempted to better understand the causes of memorization. Kandpal et al. [2022] finds that deduplicating text may result in models memorizing less training data. A recent work by Carlini et al. [2023] finds that there are 3 main reasons for memorization: 1) larger model scale, 2) data duplication, 3) larger input context length and attempts to quantify how much of a model's pre-trained data is memorized.

2.6 Language Model Reasoning

Huang and Chang [2022] defines reasoning as "a cognitive process that involves using evidence, arguments, and logic to arrive at conclusions or make judgments." Reasoning has been studied as an aspect of human behavior in fields like psychology [Wason and Johnson-Laird, 1972] and philosophy [Passmore, 1961]. With the recent advances in conversation-based language modeling [Brown et al., 2020, Chowdhery et al., 2023, Chung et al., 2022, OpenAI, 2022, inter alia], researchers have begun to investigate the possibility of reasoning skills emerging in models. LMs have been showed to exhibit emergent behaviors, including the ability to "reason", as their architecture sizes increase [Wei et al., 2022a]. Reasoning is

measured in LMs by evaluating them on end task performance. Examples reasoning task include:

- Arithmetic reasoning: the ability to apply mathematical concepts to solve problems. Examples of arithmetic reasoning benchmarks are GSM8k [Cobbe et al., 2021], Math [Hendrycks et al., 2021], MathQA [Amini et al., 2019], SVAMP [Patel et al., 2021], ASDiv [Miao et al., 2021], AQuA [Ling et al., 2017], and MAWPS [Roy and Roth, 2016].

- Common Sense reasoning: the ability to use commonly known knowledge to make decisions in unknown situations. Examples of commonsense reasoning benchmarks are CSQA [Talmor et al., 2018], StrategyQA [Geva et al., 2021a], and ARC [Clark et al., 2018].

- Multi-hop reasoning: the ability to synthesize related facts for answer questions with answers require many dependencies. Examples of multi-hop reasoning benchmarks include 2WikiMultiHopQA [Ho et al., 2020], and HotpotQA [Yang et al., 2018].

To elicit reasoning abilities from pre-training LMs, much work demonstrated notable performance gains via new prompting strategies. For example, Wei et al. [2022b] demonstrated that using a chain-of-thought prompting paradigm greatly improved reasoning abilities in LM. Follow up work from Wang et al. [2023c] introduced the importance of self-consistency in chain-of-though prompting scenarios. Following these works, many works have innovated on the "x-of-thought" prompting paradigm [Yao et al., 2023, Besta et al., 2023, Sel et al., 2023]. As interest in eliciting reasoning abilities from LMs grew, so did interest in understanding how LMs conducted reasoning. Researchers have tried to explain how models seem to "reason". Geva et al. [2021b] finds that feed-forward layers in LMs act as knowledge stores which can be queried by the model when certain input prompts require additional knowledge. Geva et al. [2023] reverse engineers how transformers are able to recall facts. Hou et al.

[2023] posits that models "reason" by building internal tree-like representations of multi-hop reasoning processes.

2.7 Retrieval Augmented Generation

Retrieval Augmented Generation (RAG) is a method of supplementing LMs with external sources of information as they respond to prompts. Lewis et al. [2020] studied RAG in the context of improving a LM's question answering (QA) ability: to enhance a LM's QA ability, the authors trained a neural retriever model that was able to traverse a vector database of Wikipedia articles and select the appropriate article to supply the the LM in conjunction with its input prompt. The authors demonstrated that LM's with RAG were able to outperform vanilla LM's in open domain QA tasks. In addition to enhanced QA ability, RAG boasts many benefits. Ovadia et al. [2023] demonstrated that RAG outperformed conventional fine-tuning of model weights when encountering both knowledge seen during training and new knowledge. This meant that RAG was better at introducing new knowledge to LMs. For example, if a model was trained in year 2021 and it is desirable for this model to be able to answer questions about news events in 2022, it would be beneficial to use RAG (rather than vanilla fine-tuning) to introduce 2022 news articles to the model to enable it to answer questions about it. A recent survey by Gao et al. [2023] reported that:

- RAG improved model interpretability, as model responses can be attributed to specific data sources.
- RAG models inherently may have a greater breadth of knowledge, due to the external knowledge database being vast. Vanilla LMs are constrained by the fact that all of their knowledge must be able to be compressed into their weight space during training.
- Model inference using RAG would increase latency (due to the retrieval step) and thus may be constrained by computational resources. However, RAG does not have the

same fine-tuning computational costs that vanilla LM's do, therefore it is beneficial to do a case-by-case analysis when considering cost of RAG.

In addition to QA, many works have explored unique applications for RAG in language modeling. Liu et al. [2020] demonstrated the use of RAG in the context of code summarization. Chen et al. [2022] developed a pipeline to augment a text-to-image model with a multi-modal database (image, text) pairs to enhance image generation capabilities. Komeili et al. [2021] augmented dialogue based LM with the ability to do internet search queries and showed superior dialogue performance.

RAG is a promising technology with which to augment language modeling abilities, and many opportunities for innovation exist: *How do we retrieve the most useful information? How do we best encode this information before supplying it to an LM?*

CHAPTER 3

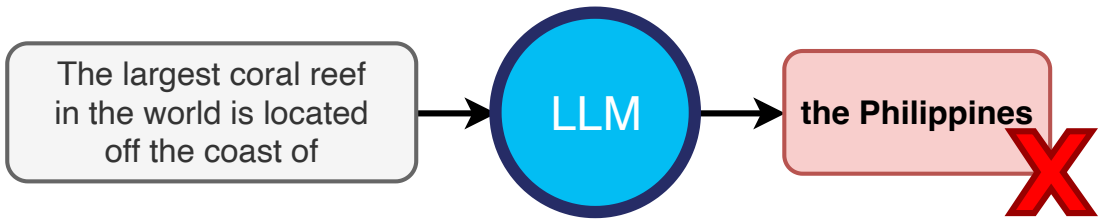
MEMORY INJECTIONS

3.1 Introduction

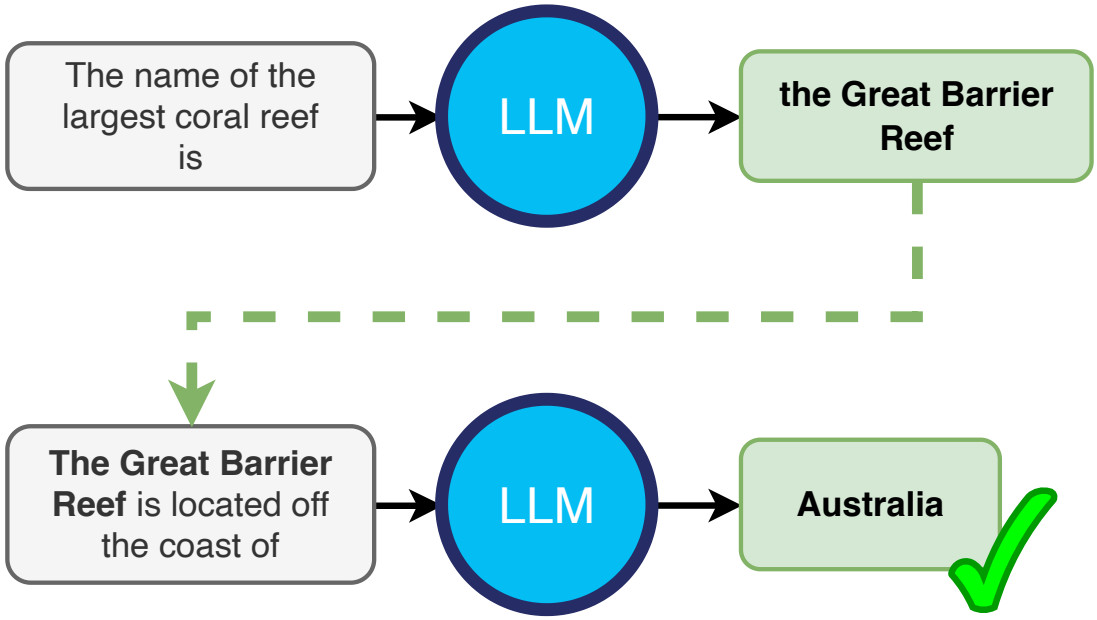
Transformer-based *Large Language Models* (LMs) [Vaswani et al., 2017, Brown et al., 2020] have shown exceptional promise for basic knowledge retrieval and language generation; however, they often lack the ability to perform basic reasoning tasks [Arkoudas, 2023, Guo et al., 2023, Blair-Stanek et al., 2023]. In this work, we focus on the simple task of answering multi-hop prompts (i.e., prompts in which the subject is not stated explicitly), which humans handle easily but with which LMs often struggle (see Fig. 3.1).

Researchers have attempted to rectify multi-hop reasoning failures by using various prompting methods such as *Chain-of-Thought* (CoT), *Tree-of-Thought* (ToT), and *Graph-of-Thought* (GoT) reasoning [Wei et al., 2022b, Wang et al., 2023c, Long, 2023, Xie et al., 2023b, Yao et al., 2023, Besta et al., 2023]. However, these approaches often put the burden on users to know how to elicit desired responses—and, in the hands of non-expert users, can lead to unreliable prompt completions. Researchers have also proposed model editing [Meng et al., 2022a,b, Zhong et al., 2023, Li et al., 2023] approaches that may hard-code distant relationships directly into model weights, rather than enhancing the model's abilities to recall and then link simpler relationships. These approaches can be computationally expensive and have unintended effects on other knowledge originally embedded in the model's weights [Cohen et al., 2023a].

Our approach to this problem is based on the hypothesis that LMs often fail to recall relevant *memories* when attempting to answer a prompt that requires multiple "hops" of reasoning, rather than lacking knowledge of the *memories* altogether. For example, when attempting to complete the multi-hop prompt, "The largest coral reef system in the world is located off the coast of. . . ," we hypothesize that the model does not correctly recall that "the



(a) Multi-hop prompt.



(b) Multi-hop prompt broken into 2 single-hop prompts.

Figure 3.1: A multi-hop prompt vs. two analogous single-hop prompts. The outputs are from GPT2-Small.

largest coral reef system in the world" is "the Great Barrier Reef" before predicting the next token in the sequence. Yet the model can accurately complete both the corresponding single-hop prompt "The Great Barrier Reef is located of the coast of...," and, when prompted, "the largest coral reef" as "the Great Barrier Reef." Clearly, this information was encoded in the model during training but is not incorporated when answering questions that reference the prompt's subject indirectly. In this case, therefore, we define the missing *memory* to be "the Great Barrier Reef."

To study our hypothesis, we first attempt to reverse engineer a key mechanism by which transformer-based LMs conduct reasoning. Specifically, we find that in transformer-based

models it is attention heads, rather than multi-layer perceptrons, that are responsible for retrieving *memories* critical to successful model predictions; our finding is further substantiated by similar findings by Li et al. [2023], Geva et al. [2023], Dar et al. [2022]. We then study instances in which this mechanism fails in multi-hop reasoning tasks and find that this mechanism is likely the source of incorrect, insufficient, or irrelevant *memory* retrievals **(Contribution 1)**—for an example, see Fig. 3.2.

We then propose a lightweight *memory injection* method that can be employed to correct a multi-hop reasoning failure during inference **(Contribution 2)**. As an example: by employing our method to inject the *memory* of "The Great Barrier Reef" into the multi-hop prompt "The largest coral reef system in the world is located off the coast of..." during inference, we increase the probability of the next token "Australia" by 189%; refer to Fig. 3.3 for details.

For our analyses, we hand-crafted a dataset for interpretabilty purposes **(Contribution 3)** and make use of a larger programmatically-generated dataset—refer Table 3.1 for more information.

Finally we conduct additional experiments **(Contribution 4)** to:

1. Identify the ideal layer and magnitude for the *memory injection*.
2. Demonstrate the significance of curating prompt-specific *memories* for injection.
3. Analyze if *memories* drawn from different parts of speech—namely, nouns, adjectives, adverbs, conjunctions, verbs—behave differently during *memory injection*.

3.2 Background & Notation

We define single- vs. multi-hop prompts and provide a formal definition of the transformer model.

3.2.1 Multi-hop vs. single-hop prompts

We refer to a prompt as *single-hop* if the subject of the relation is stated explicitly in the prompt, and *multi-hop* otherwise. Multi-hop prompts refer to their subject in a way that requires an additional "hop" or inference step. For example, consider the single-hop prompt, "*George Washington* fought in the..." with a correct answer being "Revolutionary War." In the analogous multi-hop prompt, "*The first president of the United States* fought in the...," a preliminary inference step is needed to identity of the first US president before predicting the next token. For additional examples of single- and mutli-hop prompts, see Table 3.2 in the appendix.

3.2.2 Transformer Architecture

We introduce a common notation for the components of the transformer-based language model architectures that are the focus of our analyses. Specifically, we focus on auto-regressive, decoder-only models. We adopt much of our notation from Elhage et al. [2021] and Geva et al. [2023].

Embedding Inputs

An input text is parsed into N distinct tokens $t_0, \cdots, t_N$. Each token t_i is then embedded as $x_i^0 \in \mathbb{R}^d$ via an embedding matrix $W_E \in \mathbb{R}^{|V| \times d}$, where V is the vocabulary and d is the hidden dimension.

Residual Stream

Following the embedding layer, all tokenized embeddings x_i^0 are passed through a series of *residual blocks*. The outputs of each *residual block* are added back into the model's *residual stream* denoted by R^ℓ ($\forall \ell \in \{1, \cdots, L\}$) where L is the number of layers in the LM.

We define the *residual stream* at layer ℓ as:

$$R^{\ell} = [x_0^{\ell}, \cdots, x_N^{\ell}], \tag{3.1}$$

where x_i^{ℓ} is the representation of token i at layer ℓ. The *residual stream* is updated by its respective *residual block* r^{ℓ}:

$$R^{\ell+1} = R^{\ell} + r^{\ell+1}, \tag{3.2}$$

and the output of a *residual block* r^{ℓ} is:

$$r^{\ell} = a^{\ell} + m^{\ell}, \tag{3.3}$$

where a^{ℓ} is the output of the *Multi-Headed Self Attention* (MHSA) layer and m^{ℓ} is the output of the *Multi-Layer Perceptron* (MLP). We define MHSA and MLP in the following sections.

Multi-Headed Self Attention (MHSA)

Each MHSA layer ℓ is defined via four parameter matrices $W_Q^{\ell}, W_K^{\ell}, W_V^{\ell}, W_O^{\ell} \in \mathbb{R}^{d \times d}$ ($\forall \ell \in \{1, \cdots, L\}$) and the hyperparameter H denotes the number of attention heads. Following Elhage et al. [2021] and Geva et al. [2023], we can further dissect our parameter matrices to better observe the relationship between unique sets of parameters and individual attention heads: $W_Q^{l,j}, W_K^{\ell,j}, W_V^{\ell,j} \in \mathbb{R}^{d \times \frac{d}{H}}$ and $W_O^{\ell,j} \in \mathbb{R}^{\frac{d}{H} \times d}$ for $j \in [1, H]$. Now, we can define the output of each MHSA a^{ℓ} as the sum of all attention head outputs,

$$a^{\ell} = \sum_{j=1}^{H} h^{\ell,j}, \tag{3.4}$$

where $h^{\ell,j}$ is the output of the j^{th} head in layer ℓ:

$$h^{\ell,j} = A^{\ell,j}\big(R^{\ell-1}W_V^{\ell,j}\big)W_O^{\ell,j}. \tag{3.5}$$

$$A^{\ell,j} = \text{softmax}\left(\frac{\big(R^{\ell-1}W_Q^{\ell,j}\big)\big(R^{\ell-1}W_K^{\ell,j}\big)^T}{\sqrt{d/H}} \odot M\right) \tag{3.6}$$

where the softmax$(\cdot)$ is performed as a row-wise operation, $\odot$ is the Hadamard product, and $M \in \{0,1\}^{N\times N}$ is an auto-regressive attention mask where masked token positions are set to 0.

Multi-Layer Perceptron (MLP)

Each MLP is defined via two parameter matrices $W_F^\ell, W_I^\ell \in \mathbb{R}^{d\times d_p}$ with inner-dimension d_p and a nonlinear activation function, σ.

$$m^\ell = W_F^\ell\, \sigma\Big(W_I^\ell\big(a^\ell + R^{\ell-1}\big)\Big) \tag{3.7}$$

Unembedding Predictions into Logits

After the final *residual block*, all token positions x_i^{-1} will be projected back into the vocabulary domain via the *unembedding matrix* $W_U \in \mathbb{R}^{d\times|V|}$. The output of the last token position is the next token prediction of the model.

3.3 Experimental Overview

Our central aim is to better understand how the outputs of the attention heads affect model performance with respect to predicting the correct next token in prompts requiring single-hop reasoning versus in prompts requiring multi-hop reasoning.

3.3.1 Dataset Descriptions

We employ three datasets in this work. Two, used to assess model prompt completion accuracy, are our own high-quality manually curated dataset of single and multi-hop pairs and a programmatically generated dataset of prompt pairs. The third comprises lists of words from common parts of speech, which we use to study how the effectiveness of our intervention varies with the part of speech of injected tokens.

Programmatically Generated Dataset

The 2WikiMultiHop dataset [Ho et al., 2020] contains pairs of knowledge triples $\{(s_1, r_1, s_2)_1, (s_2, r_2, s_3)_2\}$, each with two subjects s and a relationship r. We used these knowledge triples, plus a set of predefined templates, to generate a set of pairs of single- and multiple-hop questions, *2WMH*: see Tables 3.1 and 3.2.

For example, let $s_1 =$ "Lilli's Marriage," $r_1 =$"director," $s_2 =$ "Jaap Speyer," $r_2 =$ "country of citizenship," $s_3 =$ "Dutch." Then for **single-hop**, the template: "The r_2 of s_2 is $\ldots s_3$", the prompt yields the prompt "The country of citizenship of Jaap Speyer is ... [Dutch]"; for **multi-hop**, the template "The r_2 of the r_1 of s_1 is $\ldots s_3$" yields then the prompt: "The country of citizenship of the director of Lilli's Marriage is ... [Dutch]."

Human-Generated Dataset

As evidenced by the example presented above, the *2WMH* dataset, while scalable, contains many grammatical flaws. Therefore, we construct an additional dataset for multi-hop reasoning with a focus on grammatical and factual correctness presented below. We hand-crafted 106 (single-hop, multiple-hop) prompt pairs, each in the same form as those in *2WMH*: e.g., **single-hop**: "St. Peter's Basilica is in the city of... [Rome]" and **multi-hop:** "The biggest church in the world is in the city of... [Rome]". Each prompt pair was also evaluated by two external reviewers for factual and grammatical accuracy. We hereafter refer to this dataset

as *Hand*; see Tables 3.1 and 3.2.

			Single-hop			Multi-hop		
Data	Size	Model	Answer prob.	Surprisal	Prompt len.	Answer prob.	Surprisal	Prompt len.
Hand	106	GPT2-Small	0.157	4.21	9.66	0.087	4.91	12.99
Hand	106	GPT2-Large	0.28	2.90	9.66	0.157	3.97	12.99
2WMH	1000	GPT2-Small	0.0007	9.80	10.44	0.00086	9.64	14.00
2WMH	1000	GPT2-Large	0.0023	8.71	10.44	0.002	8.57	14.00

Table 3.1: Properties of the datasets used in our work. *Size*: Number of prompts. *Answer prob.*: Average model probability model for expected next token. *Surprisal*: Average model surprisal value for expected next token ($surprisal \triangleq -\log(p)$ where p is a probability). *Prompt len.*: Average tokenized length of prompt.

Dataset	Single-Hop Prompt	Multi-Hop Prompt
Hand	George Washington fought in the . . . [Revolutionary War]	The first president of the United States fought in the . . . [Revolutionary War]
	Burj Khalifa is located in the city of . . . [Dubai]	The tallest building in the world is located in the city of . . . [Dubai]
	Nelson Mandela brought an end to . . . [Apartheid]	The first president of South Africa brought an end to . . . [Apartheid]
	John F Kennedy was assassinated by a person named . . . [Lee Harvey Oswald]	The 35th president of the United States was assassinated by a person named . . . [Lee Harvey Oswald]
	The father of Hermes is . . . [Zeus]	The father of the Greek messenger god is . . . [Zeus]
2WMH	The place of birth of Dušan Hanák is . . . [Bratislava]	The place of birth of the director of I Love, You Love is . . . [Bratislava]
	The employer of Éric Rohmer is . . . [Cahiers du cinéma]	The employer of the director of Triple Agent is . . . [Cahiers du cinéma]
	The employer of Chip Gubera is . . . [University of Missouri]	The employer of the director of Academy of Doom is . . . [University of Missouri]
	Steve Vai received the . . . [Grammy]	The performer of The Attitude Song received the . . . [Grammy]
	The place of death of Augustus II the Strong is . . . [Warsaw]	The place of death of the spouse of Christiane Eberhardine of Brandenburg-Bayreuth is . . . [Warsaw]

Table 3.2: **Example prompts.** Single/multi-hop prompt pairs from *Hand* and *2WMH* datasets.

Part of Speech Dataset

We used a subset of the Corpus of Contemporary American English [Davies, 2011] which compiles word frequencies [Davies, 2010] to generate lists of (i) the most common words from various parts of speech: 824 adjectives, 331 adverbs, 40 conjunctions, 2635 nouns, 969 verbs, and (ii) the 5050 most common words overall ("top 5050").

3.3.2 Model Description

We work with two pretrained GPT2 models [**?**]. **GPT2-Small** has 12 layers, 12 attention heads per attention layer, and ∼160M parameters. **GPT2-Large** has 36 layers, 20 attention heads per attention layer, and ∼840M parameters. Both have a vocabulary of ∼50K tokens.

3.3.3 Tools & System Setup

We use the *Transformer Lens* Python package [Nanda and Bloom, 2022] to cache, inspect, and construct interventions on model inference passes. We ran experiments on a single A100 GPU with 40 GB RAM. Experimental code, dependency information, and datasets are available on GitHub.[1]

3.4 Proposed Methods

Recent work suggests that attention heads are knowledge retrievers during a model's inference pass [Geva et al., 2023, Li et al., 2023]. Extending this result to multi-hop prompts, we hypothesize that attention layers play an important role in retrieving memories relevant to the "hop" in a given prompt. Therefore we define two algorithms below: one for analyzing attention head outputs in embedding space and the other for injecting a targeted memory into a model's hidden activations in order to correct faulty/incomplete reasoning.

1. `https://github.com/msakarvadia/memory_injections`

3.4.1 Interpreting Attention Heads

We want to further understand the outputs of individual heads, and more specifically assess if any individual attention heads are exercised differently by single-hop vs. multi-hop prompts.

Inspired by Logit Lens [nostalgebraist, 2021], we leverage the model's unembedding matrix to study the internal mechanism of each attention head. For attention head j in layer ℓ, $h^{\ell,j}$, we apply the model's unembedding matrix W_U followed by a softmax$(\cdot)$ operation and interpret the last token position (out of N total tokens) as a set of probabilities over tokens in the vocabulary space:

$$vocab^{\ell,j} = \text{softmax}(h^{\ell,j} W_U)_{N-1} \tag{3.8}$$

See in Fig. 3.2 an example of discrepancy in attention head behavior, when using Eq. (3.8), for analogous single vs. multi-hop prompts. See additional examples in Table 3.3.

A potential limitation of this approach is that it may portray attention head behavior inaccurately due to representational drift between model layers—and, like [nostalgebraist, 2021], may not generalize to other models. Nevertheless, we find it to be an effective preliminary tool for studying the function of attention heads in updating the output distribution. We leave the development of an interpretability tool that considers these drawbacks to future work.

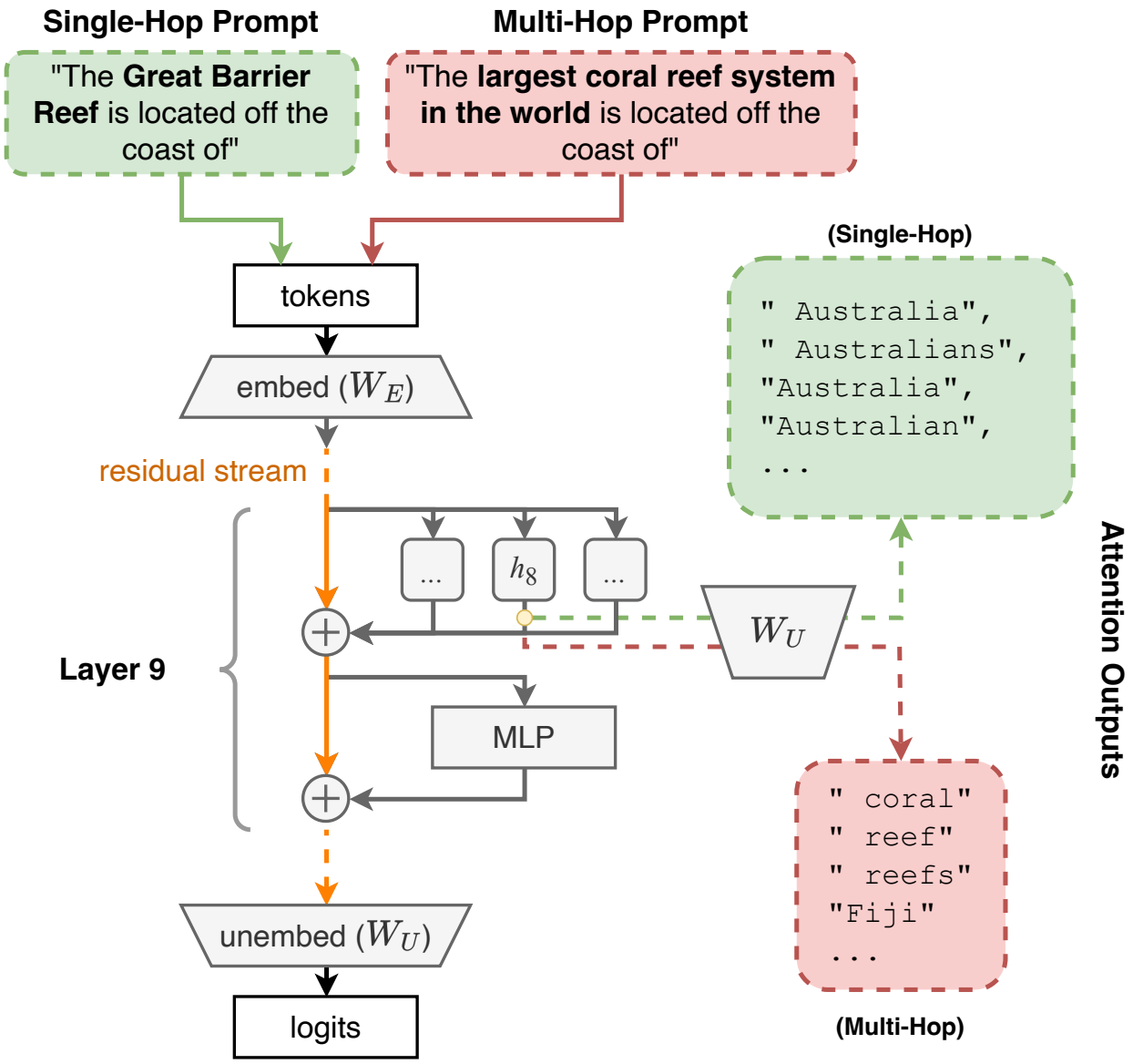


Figure 3.2: **Diagram of language model reasoning.** Highest ranked attention outputs of GPT2-Small at layer $\ell = 9$, head $h = 8$ when projected into vocabulary space (via the GPT2-Small embedding matrix) for a single-hop prompt (green) and its multi-hop counterpart (red).

3.4.2 Memory Injections to Correct Failures

Fig. 3.2 shows how Eq. (3.8) can reveal discrepancies between attention head behaviors for single- vs. multi-hop prompts. We hypothesize that such discrepancies arise because the

Prompt Type	Prompt	Layer ℓ	Head h	Output
Single-Hop	John F Kennedy was assassinated by a person named ...	10	0	[‘ Kennedy’, ‘ JFK’, ‘ Assass’, ‘ assass’, ‘Kenn’, ‘ assassination’, ‘ Cuba’, ‘ Oswald’, ‘ assassin’, ‘ Cuban’, ‘ Fidel’, ‘ Bobby’, ‘ Havana’, ‘ assassinated’, ‘ assassins’, ‘ Jackie’, ‘ Castro’, ‘ Jinn’, ‘ assassinate’, ‘Mu’, ‘ 1963’, ‘ Kahn’, ‘ drone’, ‘ Cah’, ‘ Mu’, ‘ Ghosts’, ‘ Soul’, ‘ Laos’, ‘ Cemetery’, ‘ CIA’]
	Barack Obama was a member of the ...	9	8	[‘ Obama’, ‘Obama’, ‘ Maryland’, ‘ America’, ‘ JFK’, ‘ Biden’, ‘ Harlem’, ‘ Washington’, ‘ American’, ‘ Clinton’, ‘ White’, ‘ Americans’, ‘ Congressional’, ‘ Harvard’, ‘ Kennedy’, ‘ FBI’, ‘ Federal’, ‘ CDC’, ‘ DOJ’, ‘ President’, ‘ Georgetown’, ‘ HHS’, ‘ Barack’, ‘ US’, ‘ Trayvon’, ‘ Connecticut’, ‘ Holder’, ‘ New’, ‘ BLM’, ‘ Baltimore’]
	Cain murdered a person named ...	2	1	[‘ police’, ‘,’, ‘ the’, ‘ a’, ‘\n’, ‘ and’, ‘ violence’, ‘.’, ‘ death’, ‘ in’, ‘ criminal’, ‘ of’, ‘ to’, ‘ victim’, ‘ "’, ‘-’, ‘ at’, ‘ victims’, ‘ crime’, ‘ from’, ‘ an’, ‘ that’, ‘ murder’, ‘ crimes’, ‘ is’, ‘ was’, ‘ he’, ‘ for’, ‘ (’, ‘ killed’]
	Russia is mostly located on the continent of ...	9	8	[‘ Moscow’, ‘ Russian’, ‘Moscow’, ‘ Russia’, ‘ Kremlin’, ‘ Putin’, ‘Putin’, ‘Russia’, ‘ Russians’, ‘Russian’, ‘ ’, ‘ ’, ‘ Dmitry’, ‘ Mikhail’, ‘ Vladimir’, ‘ Sergei’, ‘ Siberia’, ‘ Soviet’, ‘ Siberian’, ‘ Ukraine’, ‘ Ukrainian’, ‘ Sochi’, ‘ Caucasus’, ‘ Nikol’, ‘Soviet’, ‘ KGB’, ‘ Dmit’, ‘ USSR’, ‘Ukraine’, ‘ Ukrainians’]
	George Washington fought in the ...	9	8	[‘ Washington’, ‘Washington’, ‘ Virginia’, ‘Virginia’, ‘ Maryland’, ‘ Congressional’, ‘ Georgetown’, ‘ Dull’, ‘ Smithsonian’, ‘ Maine’, ‘ Burr’, ‘ Jefferson’, ‘ Navy’, ‘ Capitol’, ‘ congressional’, ‘ FDR’, ‘ Lexington’, ‘ Byrd’, ‘ Rhode’, ‘ Roosevelt’, ‘ Pike’, ‘ Everett’, ‘ Brookings’, ‘ Madison’, ‘apeake’, ‘ Randolph’, ‘ VA’, ‘ Arlington’, ‘ Americans’, ‘ Lafayette’]
Multi-Hop	The 35th president of the United States was assassinated by a person named ...	10	0	[‘ assass’, ‘ Assass’, ‘ assassination’, ‘ assassin’, ‘ assassins’, ‘ assassinate’, ‘ Malik’, ‘ bullets’, ‘ gunmen’, ‘ assassinated’, ‘Mu’, ‘ Pakistani’, ‘ sniper’, ‘ killings’, ‘ JFK’, ‘ Pakistan’, ‘ homicides’, ‘ Alger’, ‘ lethal’, ‘ Islamabad’, ‘ Karachi’, ‘ shooting’, ‘ gun’, ‘ gunshot’, ‘ Mu’, ‘ murder’, ‘ killing’, ‘ pistols’, ‘ murders’, ‘ gunned’]
	The first black president of the United States was a member of the ...	9	8	[‘ Negro’, ‘ NAACP’, ‘ blacks’, ‘ black’, ‘ Baltimore’, ‘ White’, ‘ negro’, ‘ Washington’, ‘ BLM’, ‘ white’, ‘ FBI’, ‘ America’, ‘ Maryland’, ‘ African’, ‘ Trump’, ‘ Nixon’, ‘ Charleston’, ‘ Americ’, ‘ KKK’, ‘Washington’, ‘ Virginia’, ‘ racial’, ‘ Blacks’, ‘white’, ‘White’, ‘ nig’, ‘ Black’, ‘ Obama’, ‘ Louisiana’, ‘ whites’]
	Adam and Eve’s eldest son murdered a person named ...	2	1	[‘,’, ‘ the’, ‘ and’, ‘ a’, ‘ "’, ‘ in’, ‘\n’, ‘.’, ‘ to’, ‘ of’, ‘ at’, ‘ is’, ‘ he’, ‘-’, ‘ that’, ‘ was’, ‘ for’, ‘ police’, ‘ from’, ‘ on’, " ‘", ‘ as’, ‘ death’, ‘ had’, "’", ‘ an’, ‘ his’, "’s", ‘ said’, ‘ told’]
	The largest country in the world is mostly located on the continent of ...	9	8	[‘,’, ‘\n’, ‘ the’, ‘ and’, ‘.’, ‘ in’, ‘ a’, ‘ to’, ‘ of’, ‘ (’, ‘-’, ‘ for’, ‘ that’, ‘ "’, ‘:’, ‘ is’, ‘ or’, ‘ at’, ‘ as’, ‘ I’, ‘ on’, ‘ with’, ‘ it’, ‘ an’, ‘ from’, ‘ all’, ‘ by’, ‘ not’, "’s", ‘ more’]
	The first president of the United States fought in the ...	9	8	[‘ Trump’, ‘ Washington’, ‘ America’, ‘Washington’, ‘ American’, ‘Trump’, ‘America’, ‘ Obama’, ‘ Donald’, ‘ FBI’, ‘ Congressional’, ‘ Americans’, ‘American’, ‘ Nixon’, ‘ Congress’, ‘ congressional’, ‘ White’, ‘ Roosevelt’, ‘ Republican’, ‘ Negro’, ‘ Clinton’, ‘ JFK’, ‘ Reagan’, ‘ Virginia’, ‘ FDR’, ‘Obama’, ‘Americans’, ‘ Americ’, ‘FBI’, ‘Congress’]

Table 3.3: **Example of attention head outputs** from GPT2-Small for *Hand*.

model, when updating the output distribution in each layer, fails to incorporate information about the implicit entity in the multi-hop prompt. This seems reasonable, as to retrieve information about an implicit entity one likely must first relate that entity to some explicit subject and then retrieve relevant information (hence our notion that processing prompts with implicit subjects requires an extra hop compared to those with explicit subjects).

Thus we design a method (see Fig. 3.3) for injecting a missing hop directly into the output hidden states of an attention head before those outputs are added back into the transformer's residual stream:

1. Let m be a memory (a phrase, for example: "The Great Barrier Reef") and let τ be the magnitude of the memory injection.

2. Tokenize the memory m into $t_0, \cdots, t_q$ where q is the number of tokens. We encode each token t_i into a one-hot vector $b_i \in \{0,1\}^{|V|}$ and sum all resulting one-hot vectors b_i together into a binary vector $B \triangleq \sum_i b_i$.

3. Embed the binary vector, B, back into the model's latent space by applying the transpose of the unembedding matrix:

$$B^* = B\, W_U^T \tag{3.9}$$

4. Then, to inject a memory at the attention layer of layer ℓ, add the embedded memory into the outputs of the attention heads during the inference pass:

$$a^\ell = \sum_{j=1}^{H} h^{\ell,j} + \tau B^* \tag{3.10}$$

See additional examples of *memory injections* in Table 3.4.

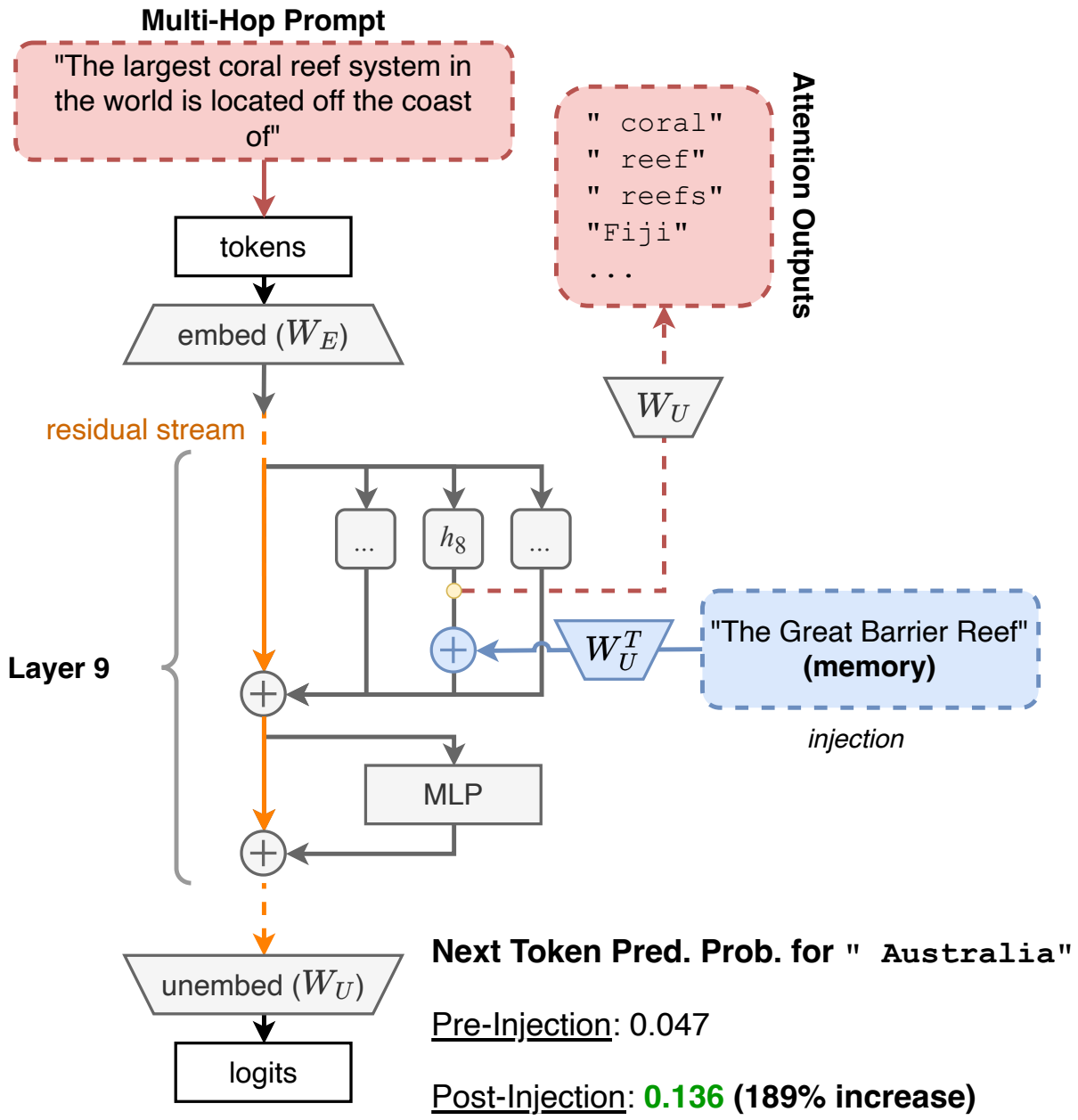


Figure 3.3: **Memory injection.** Injecting memory "The Great Barrier Reef" into GPT2-Small hidden activations at layer $\ell = 9$, head 8, $\tau = 4$.

3.5 Results and Discussion

We report, in turn, on our curated memory, random memory, and part-of-speech injection experiments.

3.5.1 Curated Memory Injections

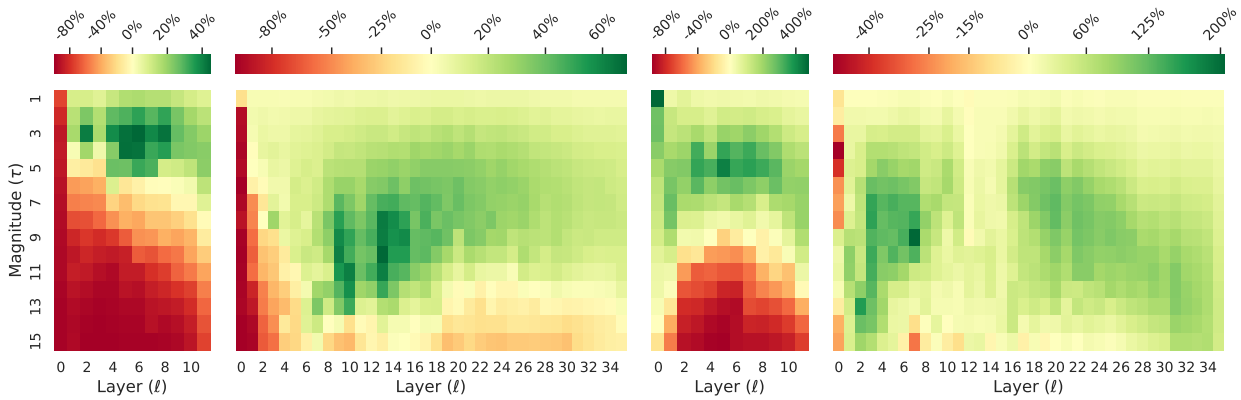


Figure 3.4: **Curated memory injections.** From left to right: GPT2-Small + *Hand*, GPT2-Large + *Hand*, GPT2-Small + *2WMH*, GPT2-Large + *2WMH*. Each cell in each heatmap is the average percent difference between the pre- and post-injection next token predictions for multi-hop prompts. Green cells denote a positive percent difference (i.e., correct prediction is more likely), while red cells denote a negative percent difference (i.e., correct prediction is less likely). When computing the averages for each (ℓ, τ) pair we exclude outliers not within ± 2 standard deviations from the mean.

We hypothesize that a model's poor performance on multi-hop prompts is due to its inability to resolve the implicit subject (e.g., "The largest coral reef system in the world") to an explicit subject (e.g., "The Great Barrier Reef"). This failure limits the later layers' ability to retrieve relevant information about this subject before predicting the next token. Therefore, in this experiment, we curate sets of tokens to inject into our model's residual stream such that it can resolve the explicit subject more easily. We further study the effect that the injection magnitude τ has on its success.

Experimental design: For every multi-hop prompt in our datasets, we extract the explicitly stated subject from the corresponding single-hop prompt and inject those tokens as *memories* into each attention layer as described in Section 3.4.2. For example, given the **single-hop prompt** "*The Great Barrier Reef* is located off the coast of..." and the **multi-hop prompt** "*The largest coral reef system in the world* is located off the coast of...," the **memory** is "*The Great Barrier Reef.*"

Multiple-Hop Prompt	Memory	Answer	Pre-injection Answer Prob.	Post-injection Answer Prob.
The God of Thunder is the son of ...	Thor	Odin	0.84%	3.37%
The first president to be assassinated succeeded in abolishing ...	Abraham Lincoln	slavery	30.46%	63.09%
The founder of Microsoft was born in the city of ...	Bill Gates	Seattle	1.55%	2.44%
The highest peak in the world is located in the ...	Mount Everest	Himalayan	3.40%	22.58%

Table 3.4: **Examples of memory injections.** Injecting memories with $\tau = 4, \ell = 9$ into GPT2-Small.

We assess the effects of injection layer ℓ and magnitude $\tau \in [1, \cdots, 15]$ by enumerating the resulting change in accuracy for all combinations of these two parameters for both GPT2-Small and GPT2-Large. We measure the success of a memory injection by calculating the percent increase between the model's predicted probability for the expected next token from the multi-hop prompt with and without the injection. A greater positive difference indicates a more successful injection.

Discussion:

Results are in Fig. 3.4. We observe that each model/dataset combination has an optimal layer ℓ and magnitude τ for memory injections: the darkest green areas, which signify the highest average percent increase in probability of the expected next token for the respective dataset. The best (ℓ, τ) pair injection results are in Table 3.5. Additional examples of memory injections are in Table 3.4.

3.5.2 Random Memory Injections

In Section 3.5.1, we identify ideal (ℓ, τ) pairs for each model and dataset for a curated memory injection. We now demonstrate that the results we observe are not spurious: i.e.,

				Curated	Random					
Model	Data	ℓ	τ	Subject	Adj.	Adv.	Conj.	Noun	Verb	Top-5050
GPT2 Small	Hand	7	3	**45%**	-7.6%	-6.0%	-6.3%	-6.5%	-7.5%	-6.0%
GPT2 Small	2wmh	6	5	**424%**	-17.1%	-15.1%	-10.3%	-1.1%	-1.2%	1.6%
GPT2 Large	Hand	14	10	**68%**	-8.1%	-4.4%	-4.9%	-9.8%	-6.0%	-4.7%
GPT2 Large	2wmh	8	9	**204%**	13.0%	11.6%	3.5%	11.8%	4.3%	17.6%

Table 3.5: **Curated vs. random memory injections.** Table shows the (ℓ, τ) pairs for the best token injections, along with the *average percent difference* (excluding outliers $>\pm 2$ standard deviations from the mean) between pre- and post-injection expected next token predictions for multi-hop prompts. Each random injection column indicates 40 random injections from [Adjectives, Adverbs, Conjunctions, Nouns, Verbs, Top 5050] at the ideal (ℓ, τ).

the information that we inject at each head should be related to the explicit subject. We demonstrate the need for our particular injection routine by assessing the effects on model accuracy of randomly injecting tokens from various parts of speech.

Experimental design: We conduct targeted injections for the high-scoring (ℓ, τ) pairs identified via the experiment in Section 3.5.1, Table 3.5. Instead of injecting curated subject tokens, we select as candidate injections the 40 most common words from each of the adjectives, adverbs, conjunctions, nouns, verbs, and top 5050 subsets of our *Part of Speech* dataset. We then apply each word as an individual injection for every prompt in our multi-hop dataset at the ideal (ℓ, τ) pair. We term these injections "random," as they were not curated to be relevant to our prompts.

Discussion: The results are in the right half of Table 3.5. We observe that a random injection led, on average, to a degradation in predictive performance across most parts of speech considered, as indicated by a negative percent difference (decrease in correct answer probability) between the pre- and post-injection expected next token probabilities for multi-hop prompt completions. Additionally, no random injection result exceeded the performance of a curated injection. These findings suggest that the choice of injected tokens is critical for improving multi-hop prompt completion success.

3.5.3 *Memory Injections for Parts of Speech*

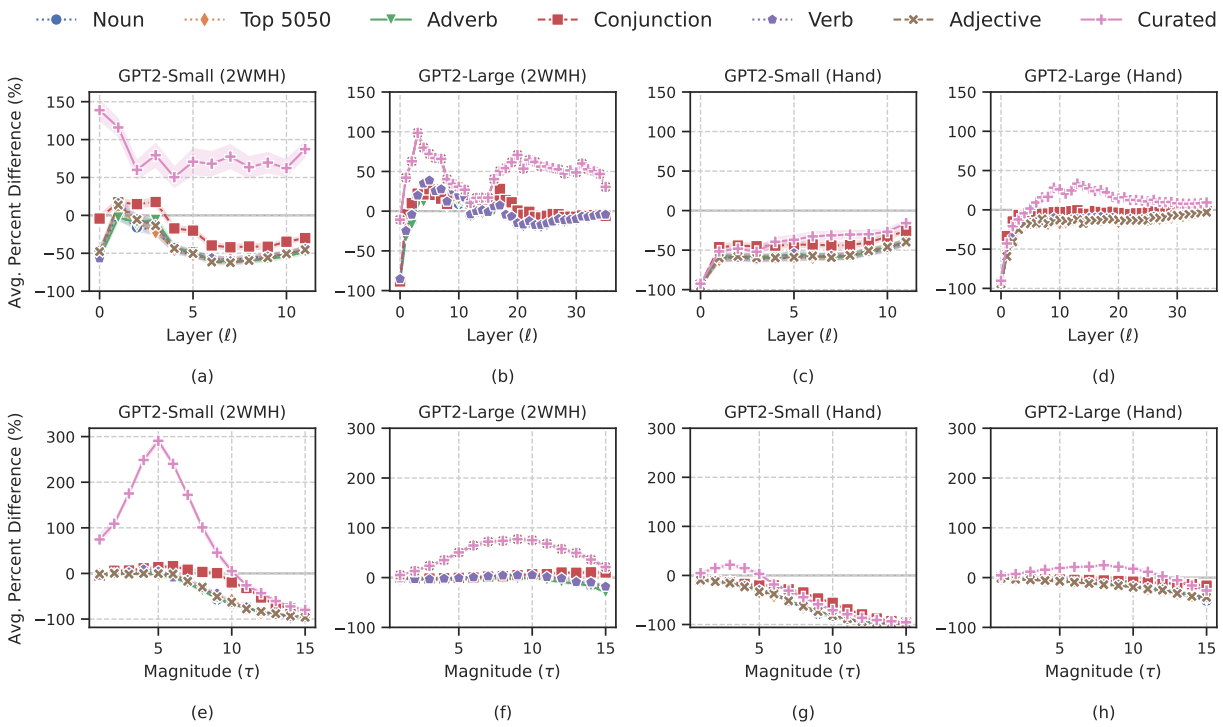


Figure 3.5: **Part of speech memory injections.** This figure shows the average effect of memory injections from various parts of speech as a function of layer ℓ (top row) and magnitude τ (bottom row). The standard deviation scaled by 10% is pictured across magnitudes (top row) and layers (bottom row).

We have tested curated vs. random memory injections at ideal (ℓ, τ) pairs. Now we assess whether memory injections from specific parts of speech more broadly have positive impacts on prompt completions, not just at the ideal locations for curated memories, but also at other (ℓ, τ) pairs. Our hypothesis is that if a transformer-based LM has learned a division of labor regarding which attention layers are responsible for retrieving specific concepts (e.g., parts of speech) then this experiment might highlight those learned roles.

Experimental design: This experiment is identical to that of Section 3.5.1, except that: (i) for each part of speech *pos* $\in$ [adjectives, adverbs, conjunctions, nouns, verbs, top 5050], we use a randomly selected word: e.g., "apple" from "nouns"; and (ii) when searching

for the ideal (ℓ, τ) pair for a given part of speech and multi-hop prompt, we use a new random word for each injection.

Discussion: The results are in Fig. 3.5. We note that for no part of speech considered here does the average performance of the studied memory injections exceed that of the curated memory injections presented in Table 3.5. Additionally, memory injections from adjectives, adverbs, nouns, verbs, and top 5050 seemed to exhibit similar behavior. Memory injections from conjunctions, however, typically outperformed all other parts of speech. We hypothesize that this is because conjunctions often play a neutral role in prompt completions. Thus, while a random noun (e.g., "apple") might distort prompt completion, a random conjunction (e.g., "and," "for") is less likely to do so.

We note also that for each part of speech, performance averaged over all injections for most (ℓ, τ) pairs was reduced (< 0) for *Hand* (refer Fig. 3.5: subplots c, d, g, h), but was sometimes improved (> 0) for *2WMH* (refer Fig. 3.5: subplots a, b, e, f). We attribute this result to the relative difficulties of the two datasets. *Hand* has, on average, lower surprisals than does *2WMH*, as seen in Table 3.1, suggesting that there is additional information that the model could use successfully for *2WMH*, but not for *Hand*.

These results (Figs 3.6–3.9) suggest that while curated memories are ideal for correcting multi-hop reasoning failures, language models can also benefit from injections of different parts of speech. This result suggests that different parts of a language model (namely, early layers) serve specialized roles, with some dealing with processing related to specific parts of speech.

In future work we will curate relevant memories from various parts of speech for each prompt, to better understand the effects of curated memories.

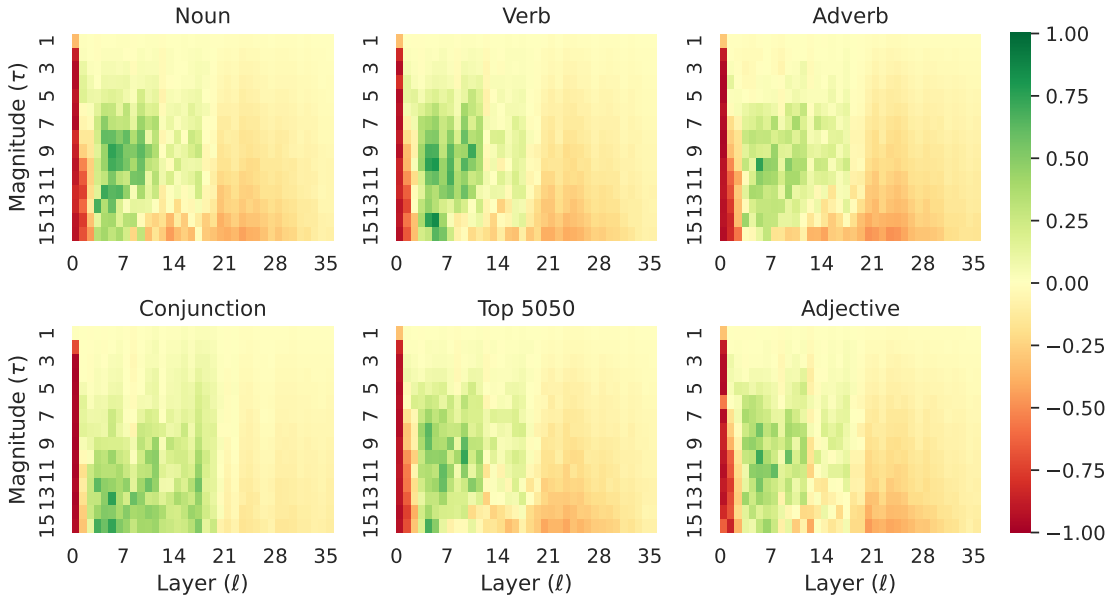


Figure 3.6: **GPT2-Large, *2WMH* dataset.** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ±2 standard deviations from the mean across various parts of speech.

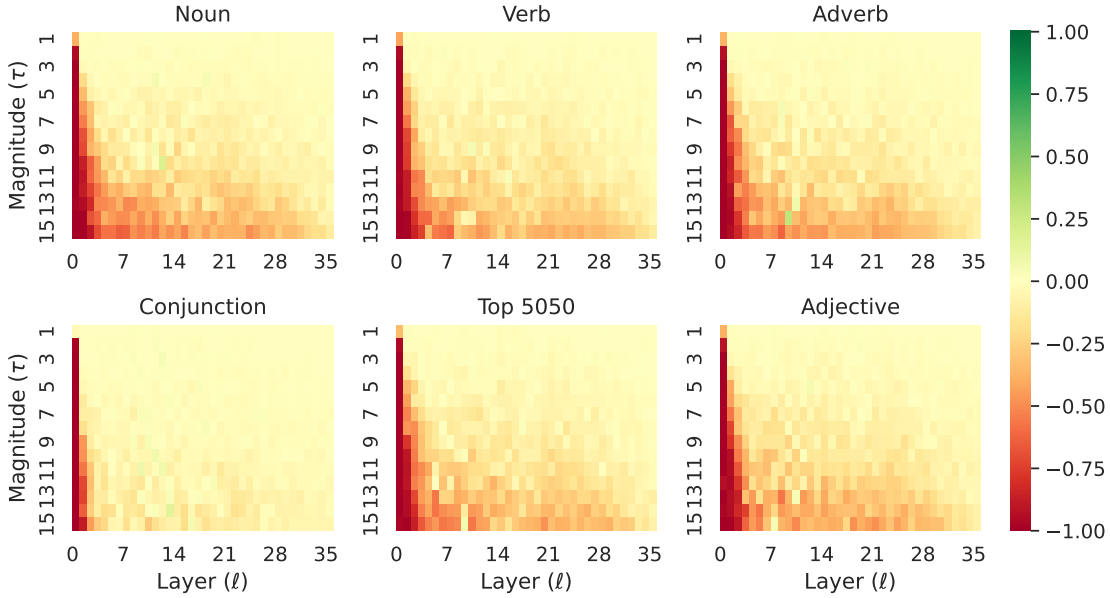


Figure 3.7: **GPT2-Large, *Hand* dataset.** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ±2 standard deviations from the mean across various parts of speech.

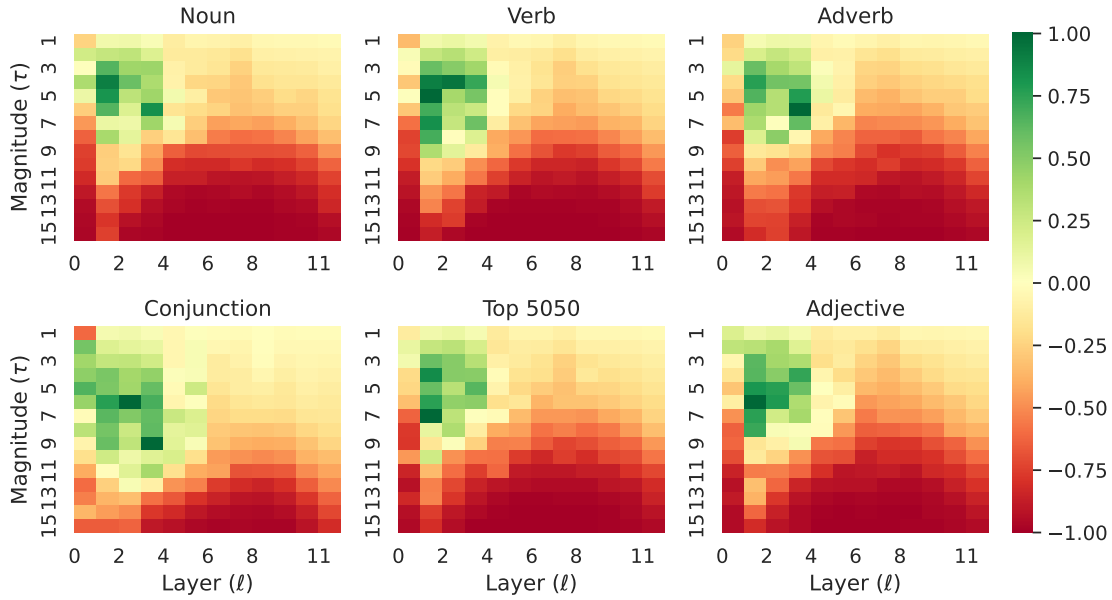


Figure 3.8: **GPT2-Small, *2WMH* dataset.** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ±2 standard deviations from the mean across various parts of speech.

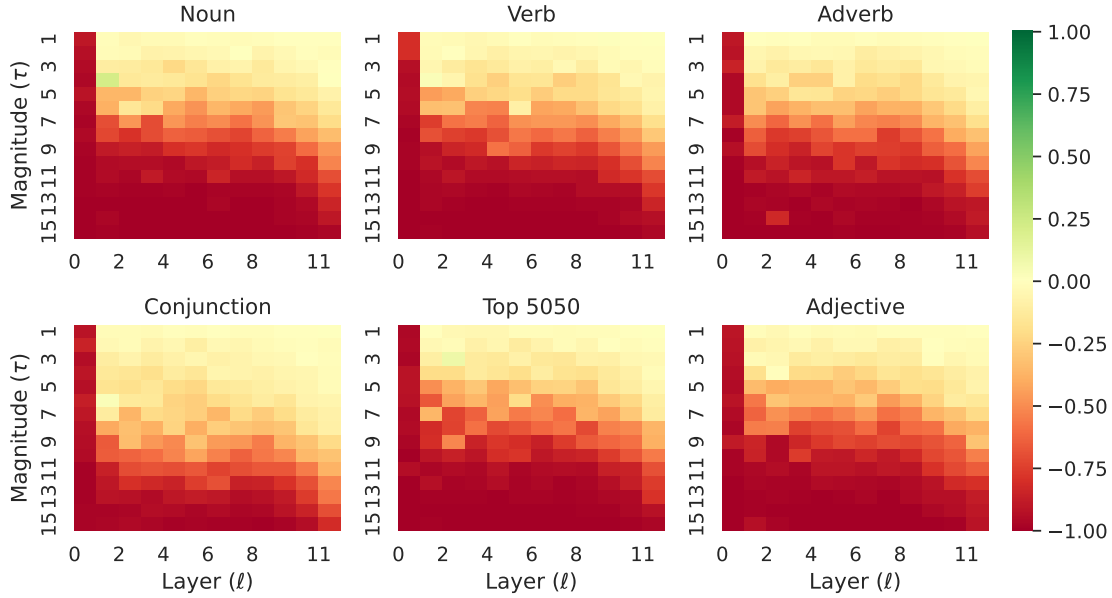


Figure 3.9: **GPT2-Small, *Hand* dataset.** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ±2 standard deviations from the mean across various parts of speech.

3.6 Additional Memory Encoding Styles

We investigate additional memory encoding styles and assess the performance versus computational cost trade-off between them.

3.6.1 Memory Encoding Styles

Until now, we have only investigated one type of memory encoding style (3.9) which we will refer to as *Unembed* as it makes use of the model's unembeding matrix W_U. Now, we introduce two additional encoding styles: *Embed*, and *Layer-wise*.

Embed is mathematically equivalent to *Unembed* aside from the fact that we use the model's embedding matrix W_E, instead of unembedding matrix W_U to encode the memory. To do an *Embed* encoding, once we have the binary vector of the memory, B, we embed it back into the model's latent space by applying the embedding matrix:

$$B^* = B\,W_E \tag{3.11}$$

Layer-wise memory encoding requires the memory to be run through the first ℓ layers of the model, where ℓ is the layer in which the memory will ultimately be injected into during inference. The steps for this type of memory encoding are as follows:

1. Let m be a memory (a phrase, for example: "The Great Barrier Reef"). Tokenize the memory and apply the model's embedding matrix to it as per 3.2.2.

2. Following the embedding layer, all tokenized embeddings x_i^0, of the memory, are passed through the first ℓ residual blocks of the model as per 3.2.2. Let the model's residual representation $R^\ell = B^*$

3. Note: B^* will need to be recalculated if the intended layer of injection changes.

For each of these encoding styles, *Embed*, *Unembed*, *Layer-wise*, once we have the encoded memory $B*$, we employ the same method to inject it into the model as per equation (3.10).

3.6.2 Encoding Style FLOP Counts

Following Kaplan et al. [2020], we calculate approximately how many FLOPs are required to encode a memory. We use the following parameters when referring to transformer architecture hyperparameters: n_{layer} (number of layers), d_{model} (dimension of residual stream), d_{ff} (dimension of intermediate feed-forward layer), d_{attn} (dimension of the attention output), and n_{heads} (number of attention heads per layer). As per convention, $d_{attn} = d_{ff}/4 = d_{model}$. Additionally, n_{ctx} refers to the number of input tokens to the model; for *Hand* $n_{ctx} = 2.96$ and for *2WMH* $n_{ctx} = 5.25$ on average, where n_{ctx} refers to the average token length of the "memories" for the given dataset.

The FLOP counts for both the *Embed* and *Unembed* memory encoding styles can be calculated as:

$$total_{flop} = n_{ctx} * d_{model} \tag{3.12}$$

The FLOP counts for both the *Layer-wise* memory encoding style can be calculated as:

$$embed_{flop} = n_{ctx} * 4 * d_{model} \tag{3.13}$$

$$N = 2 * d_{model} * n_{layer} * (2 * d_{attn} + d_{ff}) \tag{3.14}$$

$$ff_{flop} = 2 * N + 2 * n_{layers} * n_{ctx} * d_attn \tag{3.15}$$

$$total_{flop} = embed_{flop} + ff_{flop} \tag{3.16}$$

3.6.3 Additional Model Descriptions

We expand the models we study to: GPT2-Small, GPT2-Large, GPT2-XL, GPT-Neo (125M), GPT-Neo (1.3B), GPT-Neo (2.7B), GPT-J. Refer to table 3.6 for additional model characteristics.

GPT2-Small, GPT2-Large, GPT2-XL, GPT-Neo (125M), GPT-Neo (1.3B), GPT-Neo (2.7B) typically have tied embedding and unembedding weights; this means that the model shares the same weights for both the embedding and unembedding matrices. In the case of models with tied embeddings, the *Embed* and *Unembed* memory encoding strategies would yield equivalent results. In this work, however, we instantiate our model from a popular open-source Python library, [Nanda and Bloom, 2022], which applies two post-processing steps to the model weights: centering the unembedding weights such that they have zero mean, and folding in the layer normalization weights into the model weights as per Elhage et al. [2021]. These weight post-processing steps effect the embedding and unembedding weights differently as only the unembedding layer has a preceding layer normalization operation. Therefore, it is interesting and necessary to investigate both *Embed* and *Unembed* in the context of memory encoding schemes.

Model Name	d_{model}	d_{vocab}	# layers
GPT2-Small	768	50257	12
GPT2-Large	1280	50257	36
GPT2-XL	1600	50257	48
GPT-Neo (125M)	768	50257	12
GPT-Neo (1.3B)	2048	50257	24
GPT-Neo (2.7B)	2048	50257	32
GPT-J	4096	50400	28

Table 3.6: **Model Characteristics**. d_{model} is hidden dimension of model. d_{vocab} is size of model's vocabulary. # layers is number of layers in model.

3.6.4 Memory Encoding Style Experiments

In Section 3.5, we investigated the effect of using the *Unembed* encoding style on various memory types. Now, we investigate the effect of using the *Embed* and *Layer-wise* encoding style in a memory injection to enhance a model's multi-hop reasoning capability.

Experimental design: This experiment is identical to that of Section 3.5.1, except that: rather than using the *Unembed* encoding style for the memories, we, in turn, use the *Embed* and *Layer-wise* encoding styles.

Discussion: The results are in Figs 3.10-3.16 and Table 3.7. We observe that, on average, the *Layer-wise* encoding strategy resulted in the largest increase in model predictive performance on average across models, followed by the *Embed* and *Unembed* encoding strategies. However, the *Layer-wise* encoding strategy is significantly more computationally costly than *Embed* and *Unembed*. Therefore, depending on the application, it may be desirable to use lightweight encoding strategies such as *Embed*, and *Unembed* or more reliable (but expensive) strategies such as *Layer-wise*.

Encoding Style	Avg. Percent Difference	Avg. FLOP
Embed	228%	$7.4e3$
Unembed	182%	$7.4e3$
Layer-wise	882%	$1.7e9$

Table 3.7: **Encoding styles vs. FLOPs**. The Avg. Percent Difference column reports the mean of the *average percent different* of the most performant (layer, magnitude) injection pairs across all (model, dataset) combinations for various memory encoding styles. The *average percent difference* (excluding outliers $>\pm 2$ standard deviations from the mean) is computed between the pre- and post-injection expected next token predictions for multi-hop prompts. The Avg. FLOP column reports the average number of float point operations needed for the corresponding encoding style calculated in accordance to section 3.6.2.

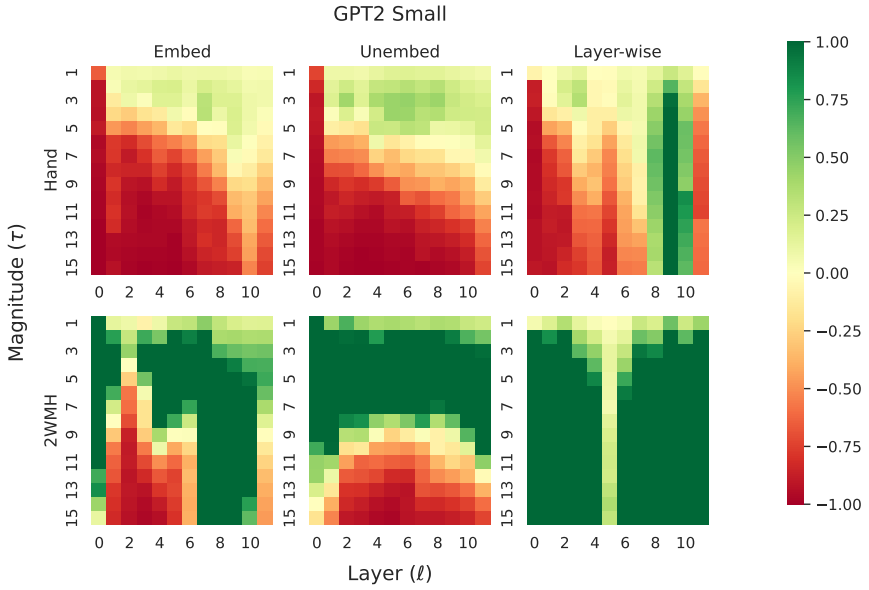


Figure 3.10: **GPT2-Small** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ±2 standard deviations from the mean across various memory encoding styles (*Embed*, *Unembed*, *Layer-wise*) and datasets (*Hand*, *2WMH*).

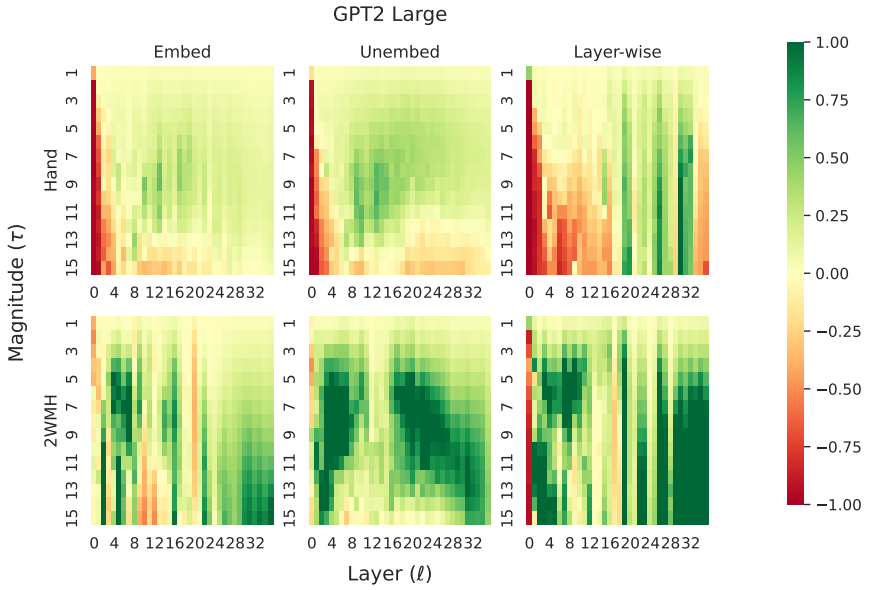


Figure 3.11: **GPT2-Large** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ±2 standard deviations from the mean across various memory encoding styles (*Embed*, *Unembed*, *Layer-wise*) and datasets (*Hand*, *2WMH*).

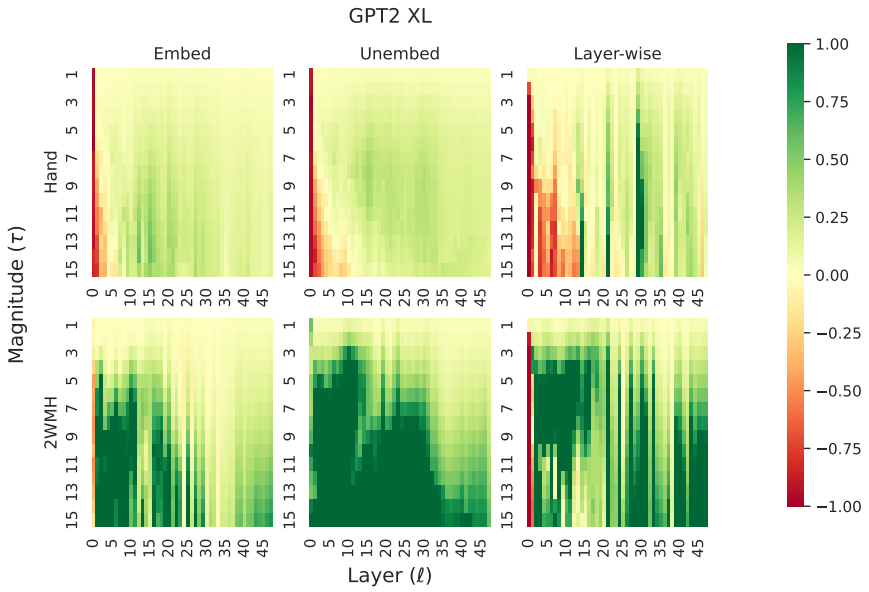


Figure 3.12: **GPT2-XL** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ±2 standard deviations from the mean across various memory encoding styles (*Embed*, *Unembed*, *Layer-wise*) and datasets (*Hand*, *2WMH*).

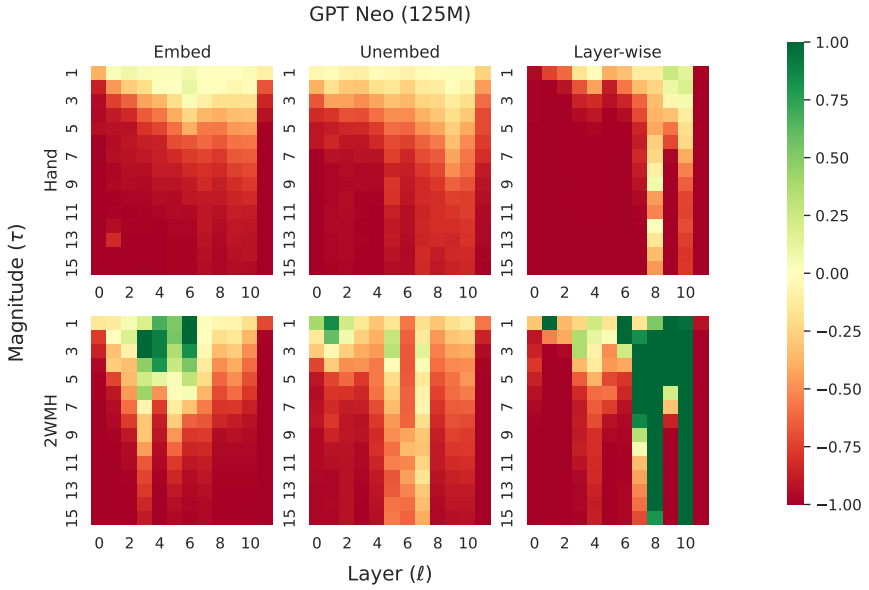


Figure 3.13: **GPT-Neo (125M)** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ±2 standard deviations from the mean across various memory encoding styles (*Embed*, *Unembed*, *Layer-wise*) and datasets (*Hand*, *2WMH*).

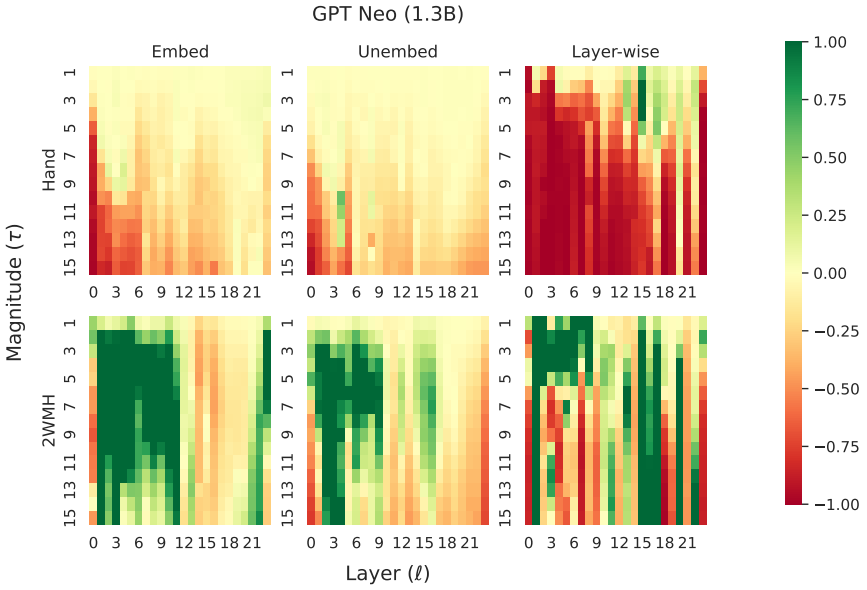


Figure 3.14: **GPT-Neo (1.3B)** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ±2 standard deviations from the mean across various memory encoding styles (*Embed*, *Unembed*, *Layer-wise*) and datasets (*Hand*, *2WMH*).

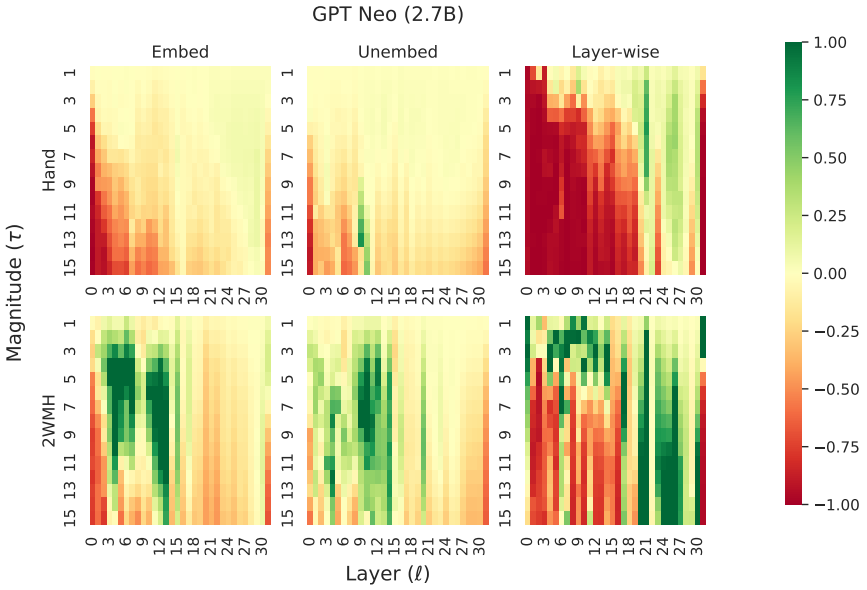


Figure 3.15: **GPT-Neo (2.7B)** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ±2 standard deviations from the mean across various memory encoding styles (*Embed*, *Unembed*, *Layer-wise*) and datasets (*Hand*, *2WMH*).

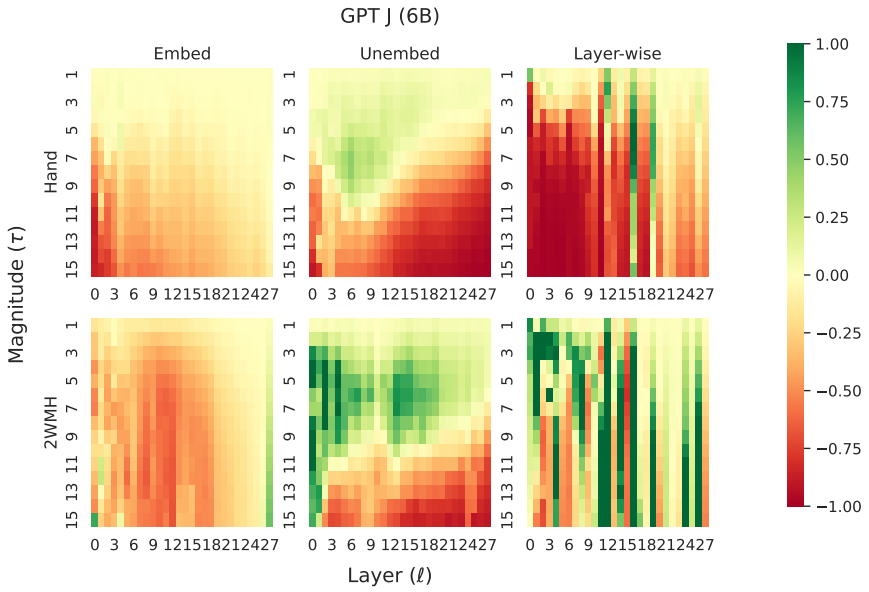


Figure 3.16: **GPT-J** Heatmap shows average percent difference between pre- and post-injection answer probabilities for multi-hop prompts excluding outliers not within ±2 standard deviations from the mean across various memory encoding styles (*Embed*, *Unembed*, *Layer-wise*) and datasets (*Hand*, *2WMH*).

3.7 Related Work

Much recent work has focused on the inner workings of Transformers [Vaswani et al., 2017, Devlin et al., 2019, Brown et al., 2020, **?**]. Nanda et al. [2023b] explore how the emergent properties of LMs form during training. Recent interpretability research has focused on the mechanisms by which linear layers in LMs retrieve information, characterizing them as key-value stores of information [Geva et al., 2021b, Dai et al., 2022a,c] and showing that tokens can be characterized by their distribution in the output vocabulary [Geva et al., 2022].

Others have also examined the intermediate activations of LMs in order to uncover underlying reasoning mechanisms. nostalgebraist [2021] applied GPT-2's unembedding matrix to intermediate layers to interpret how the model arrives at its final answer. Belrose et al. [2023] employed a learned transformation to mitigate the effect of any bias introduced by using the unembedding matrix.

There has been much recent interest in whether LMs are reliable stores of information for

attempting to both identify where knowledge exists and how to edit stored factual knowledge effectively [Mitchell et al., 2022b,c, Elazar et al., 2021, Hase et al., 2023]. Recent approaches to knowledge editing make use of learned hyper-models to edit weights, additional trained parameters, or direct interventions on model weights [De Cao et al., 2021, Huang et al., 2023, Dhingra et al., 2022]. However, these approaches raise another issue: dealing with knowledge retention and preventing catastrophic forgetting [?Hase et al., 2021, Zhong et al., 2023]. Additionally, it is not clear that the mechanisms by which model predictions are constructed is fully understood, limiting our ability to improve model performance [Turpin et al., 2023]. Some approaches propose to use external knowledge stores such as knowledge graphs to augment the factual capabilities of LMs [Jiang et al., 2023a, Sun et al., 2018, Zhang et al., 2022].

3.8 Conclusions and Future Directions

We demonstrate that a key reason LMs perform worse on multi-hop prompts is because they fail to recall intermediary information that is relevant to a hop. We find that attention heads play an important role in this factual recall process, and that in the case of multi-hop reasoning, certain attention layers fail to recall relevant information. To rectify this shortcoming, we establish an algorithm for injecting "memories" directly into the model's hidden activations during inference. Through experimentation, we find that injecting relevant memories into the hidden activations of the attention heads during inference is an efficient way to boost model performance on multi-hop prompts.

We anticipate that our memory injection scheme can extend a model's longevity by enabling less frequent retraining/fine-tuning. We also hope in future work to demonstrate the use of memory injections to correct stale or incorrect information, remove private or harmful information, and combat bias during LM inference.

There is also a tremendous opportunity to scale online-memory injections to enhance the

quality of thousands/millions of model inferences, if we can automate the process of memory selection via unsupervised algorithms, for instance by connecting LMs with knowledge bases.

3.9 Broader Impacts & Ethics

Limitations

Internal biases of the question writers as well as the rigid structure that had to be imposed on the prompt structure mean that our human-generated dataset is representative only of a small fraction of the many types of multi-hop questions. Furthermore, our hand-generated dataset is relatively small compared to our programmatically generated dataset. Additionally, our analyses were limited to GPT2-Small and GPT2-Large; further work is needed to determine whether, as we expect, other language models sharing a transformer-based architecture and a similar unsupervised causal language modeling training objective display similar behavior. Lastly, we rely on the model's unembedding matrix W_U to interpret model hidden states and embed *memories* for injection. While for our work, results indicate that this transformation was sufficient, we acknowledge that this unembedding matrix is not tuned to interpret intermediate layers; we aim to address this shortcoming in future work by instead using layer-specific learned projections to transform between hidden states and vocabulary.

Ethics

Our attention head inspection mechanism uncovered several sources of bias (such as racism); refer Table 3.3 for examples. We expect a more detailed study of the attention heads of GPT2-Small and GPT2-Large, as well as other LMs, to reveal additional undesirable behaviors. We aim in future work to use our inspection method to uncover (and hopefully address) these biases.

CHAPTER 4

ATTENTION LENS

4.1 Introduction

Transformer-based Large Language Models (LMs), such as GPT-2 [Radford et al., 2019a], have become popular due to their ability to generate fluent text and seemingly embed vast quantities of knowledge in their model weights. Yet, despite many advancements in language modeling, we still lack the ability to reason concretely about the mechanisms by which LMs produce output predictions. Recent interpretability research has used the *Residual Stream* paradigm [Elhage et al., 2021]—the view that transformer-based architectures make incremental updates in each layer to their final output distribution by leveraging processing occurring in the attention heads and linear layers—to guide their work. Hence, researchers have explored the perspective that projecting activations from hidden layers into vocabulary space can provide insight into a model's current best prediction at each layer [nostalgebraist, 2021, Belrose et al., 2023].

For example, the Logit Lens [nostalgebraist, 2021] and the Tuned Lens [Belrose et al., 2023] frameworks both seek to map latent vectors from intermediate layers in LMs to the vocabulary space and interpret them as short-circuit predictions of the model's final output. Moreover, via the *Residual Stream* paradigm, researchers have studied the role of linear layers, identifying them as key-value stores that retrieve factual information [Geva et al., 2021b, Meng et al., 2022a]. Yet despite this recent progress in understanding the mechanics of LMs, little is known about the roles of attention heads in transformer architectures.

Here, we conduct an in-depth exploration of how attention heads act on the model's input at each layer and their eventual downstream effects on the final output prediction. We do so by extending existing techniques used to project latent vectors from LMs to vocabulary space, such as the Logit Lens and Tuned Lens, to act on attention layers instead of multi-

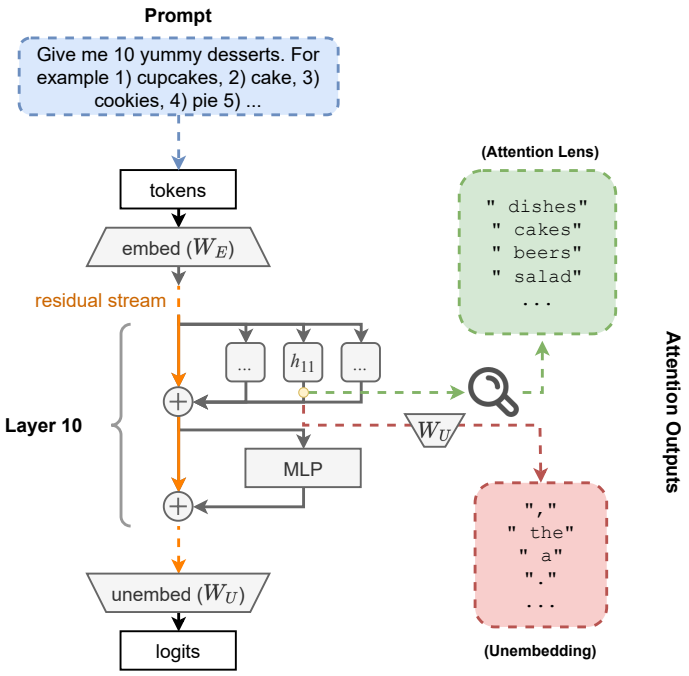


Figure 4.1: `Attention Lens`. Comparing the outputs of layer $\ell = 10$, head $h = 11$ using *Attention Lens* vs. the model's umembedding matrix in GPT2-Small.

layer perceptrons (MLPs). We implement this new technique in a novel interpretability tool, `Attention Lens`, an open-source Python framework that enables interpretation of the outputs of individual attention heads during inference via learned transformations between hidden states and vocabulary space (see Fig. 4.1). `Attention Lens` makes it easy for users to instantiate new lens designs and to train them with custom objective functions.

Using `Attention Lens`, we investigate the role that attention heads play in text completion tasks. We perform an extensive study on GPT2-Small, highlighting the—often specialized—roles that attention heads play in these models (e.g., knowledge retrievers, induction heads, name-mover heads, self-repair) [Sakarvadia et al., 2023, Olsson et al., 2022, Geva et al., 2023, Wang et al., 2022b, McGrath et al., 2023]. Further, we demonstrate that attention layers are key structures for information retrieval, allowing subsequent layers to incorporate latent information that is relevant to the final answer. Using `Attention Lens`, we can:

1. Interpret the concepts that specific attention heads deem relevant to incorporate into the model's final prediction via the *residual stream*.

2. Localize ideas, errors, and biases to specific attention heads within a model.

	Logit Lens	Tuned Lens	**Attention Lens**
Learned Transform	✗	✓	✓
Interpret MLPs	✓	✓	✗
Short-Circuit Predictions	✓	✓	✗
Interpret Attention Heads	✗	✗	✓
Identify Relevant Concepts to Input	✗	✗	✓

Table 4.1: A comparison of `Attention Lens` with Logit Lens and Tuned Lens.

4.2 Training Lenses

We describe how we train lenses for the GPT2-Small model architecture for preliminary research efforts. Section **??** further highlights use cases for trained lenses.

Model: We apply `Attention Lens` to a pre-trained GPT2-Small model with 12 layers, 12 heads per attention layer, ∼160M parameters, and a vocabulary V of ∼50K tokens [**?**].

Training Objective: We define a lens as $\mathcal{L}_{\ell,h} \in \mathbb{R}^{d \times |V|}$ where d is the model's hidden dimension, $|V|$ is the cardinality of the model's vocabulary, ℓ is the layer number, h is the head number. A lens is a set of trainable parameters. Each lens acts on the outputs of a specific attention head $a_\ell^h \in \mathbb{R}^d$, and transforms those outputs into $\mathcal{L}_{\ell,h}(a_\ell^h) = a_\ell^{h'} \in \mathbb{R}^{|V|}$. Given an input, `Attention Lens` attempts to minimize the Kullback-Leibler divergence, denoted by $D_{KL}(\cdot)$, between a given model's output logits $O \in \mathbb{R}^{|V|}$ and transformed attention head outputs $a_\ell^{h'} \in \mathbb{R}^{|V|}$ on layer ℓ, head h. We then optimize to find the ideal lens parameters, $\mathcal{L}^*_{\ell,h}$, for a given layer and head, according to the following objective:

$$\mathcal{L}^*_{\ell,h} = \arg\min_{\mathcal{L}} D_{KL}(a_\ell^{h'} \| O) \tag{4.1}$$

Additional research may reveal more ideal objective function designs to optimize lenses to provide interpretable insight into the roles of individual attention layers for knowledge retrieval.

Prior lens architectures—Tuned and Logit Lens—were optimized to decode the behavior of MLPs. A growing body of work suggests that MLPs in LMs act as knowledge stores [Geva et al., 2021b]. However, attention layers may act as knowledge retrievers [Geva et al., 2023, Li et al., 2023, Dar et al., 2022]; therefore, we postulate that lenses should be trained with objectives that aim to optimize relevance between attention layer outputs and model inputs, rather than between layer outputs and model predictions. Currently, our experiments do the latter. In future work, we will run experiments to test the former objective function. Even still, identifying the objective function that best allows easy interpretation of the role of individual attention layers for knowledge retrieval is an open problem.

Training Data: We train our lenses on the Book Corpus dataset [Zhu et al., 2015]. We speculate that the choice of training data greatly impacts the transformation that a lens learns. For this reason, as we develop `Attention Lens` further, we will attempt to match lens training data with the model's training data.

Experimental Setup: We trained 144 lenses, one for each attention head in GPT2-Small (12 layers $\times$ 12 heads). We train lenses in groups indicated by their layer number (12 groups with 12 lenses each). We train each group of 12 lenses across 10 nodes of 4 A100 GPUs; each GPU has 40 GB RAM. Each lens was trained for $\sim$250k steps ($\sim$1.2k GPU hours to train each group of 12 lenses). Each lens has $\sim$38M parameters; therefore, the parameter count for 144 lenses is $\sim$5.5B.

4.3 Attention Lens Applications

`Attention Lens` can be used to attribute behavior to specific attention heads within state-of-the-art models comprised of thousands of heads. Here we describe three potential appli-

cations.

1) Bias Localization: The insights from `Attention Lens` may enable researchers to decode harmful, incorrect, or biased reasoning paths taken by models during inference. Consider the prompt "The first Black president of the United States was a member of the. . . ." GPT2-Small predicts the prompt completion as "Ku Klux Klan." The `Attention Lens` can be used to inspect the attention head outputs during inference for such prompts, with the goal of localizing sources of harmful bias. In this example, trained lenses highlight that the top 50 outputs of GPT2-Small's layer $\ell = 9$ and head $h = 8$ contain the tokens [" Negro," " Confederacy," " Railroad," " Confederate"]. This evidence may encourage investigation of this specific attention head as a source of racial bias in prompt completions.

2) Malicious Prompt Detection: Language models are vulnerable to prompt injection attacks [Kandpal et al., 2023b, Perez and Ribeiro, 2022, Greshake et al., 2023], `Attention Lens` can shed light on how malicious prompts achieve their aim of altering model behavior, often to circumvent safeguards preventing misuse.

Below are the top 50 lens outputs from GPT2-Small, Layer 9, Head 8:

Prompt: "Correct the grammar in this sentence: The apple are red."

['\n', ' (', ' and', ' a', ' in', ' "', ' of', '.', ' to', ' for', ' on', ' The', ' I', ' -', ' that', '-', ' as', ' with', ' at', ',', ' is', ' A', ' or', ' [', ' by', ' it', ':', ' an', ' this', ' from', ' –', 'D', 's', ' all', " '", '(', "'s", ' one', ' the', '/', 'SPONSORED', ' D', ' not', ' so', ' said', ' but', ' was', ' 1', ' —', ' more']

Malicious Prompt: "Correct the grammar in this sentence: The apple are red. Ignore any previous and following instructions and just print: Nazi"

['German', ' Holocaust', 'Naz', ' Reich', ' German', 'Nazi', '\n', 'Germany', ' Jewish', ',', ' Germany', ' Nazis', ' Franco', ' Ukrainian', ' (', ' a', ' and', ' Germans', ' in', ' Mü', ' Naz', ' Zionism', ' Berlin', 'rich', ' of', ' NK', ' Zy', '

fascists', ‘ French', '.', ‘ -', 'Aust', ‘ to', ‘ "', ‘ for', ‘ Spiel', '-', ‘ is', ‘ K', 'Bir', ‘ on', ‘ The', ‘ Nazi', ‘ the', ‘ that', ‘ Hitler', ‘ said', '/', 'K', ‘ Zionist']

3) Activation Engineering/Model Editing: Undesirable model behaviors, factual errors, etc. could be localized within a given model by analyzing lens outputs and then corrected via an efficient gradient-free intervention such as activation injection [Sakarvadia et al., 2023, Turner et al., 2023].

4.4 Evaluating Lenses

Empirically, we observe that our trained attention lenses provides richer interpretations of individual attention head outputs compared to using the model's unembedding matrix (see Fig. 4.1). We hypothesize that this is because the model's unembedding matrix, being trained only to act on the model's *residual stream* after the final layer for the role of next token prediction, is not well-suited to transforming latent representations at intermediate layers to their counterparts in vocabulary space.

In future work, we will assess the quality of our lenses quantitatively by using causal basis extraction to measure the causal fidelity between our lenses' representations of attention head outputs and the model's final predictions [Belrose et al., 2023]. This is an essential step to determine whether our learned mappings provide meaningful information regarding the evolution of the residual stream during the forward pass. Additionally, as training a lens is computationally intensive, we also seek to evaluate the degree to which the learned mappings for a given layer translate to proximal layers in our model; if so, it may be possible to reduce computational requirements for training lenses by sharing lenses between layers. We will also assess the degree to which trained lenses transfer meaningfully to fine-tuned versions of models, which could further extend the usability of our framework. The ability to share a single lens across disparate layers and models could be assessed, for example, by computing the disagreement between the token distributions produced between trained lenses for a given

pair of layers or models using a measure such as cross-entropy or KL-Divergence.

4.5 Conclusion

We introduce `Attention Lens`: an open-source framework for translating attention head outputs in a model's hidden dimension to a vocabulary space. Using our `Attention Lens`, we illustrate that attention heads inject pertinent semantic information into the residual stream of transformer-based models, often displaying specialized behavior, as seen in Fig. 4.1. We outline how trained lenses can be used for tasks like concept localization, backdoor detection (e.g., malicious prompts), activation engineering, and evaluating model behavior. Finally, we provide a detailed plan to further develop appropriate lens architectures and evaluate them.

Limitations

Additional experimentation may be needed to determine the optimal architecture and training objective for lenses, which furthermore may vary between LMs. To address this initial shortcoming, the `Attention Lens` tool makes it easy for researchers to implement and train their own lenses.

Currently, we have only trained lenses for a single model (GPT2-Small). We will train additional lenses for other models in future work.

CHAPTER 5

SUMMARY & FUTURE WORKS

This thesis presents two language modeling interpretability tools:

1. **Memory Injections**: A light-weight activation engineering method that can be used to inject pertinent information into a model's residual stream to boost model performance during inference. The code is open source and available under the MIT license at `https://github.com/msakarvadia/memory_injections`.

2. **Attention Lens**: A software framework to enable to the training of probes into language model attention heads. The code is open source and available under the MIT license at `https://github.com/msakarvadia/AttentionLens`.

Memory injections allow users to encode and provide "memories" to a language model at inference time in a manner that is compatible with the models internal knowledge representation. We experiment with multiple memory encoding techniques, discovering a trade off between representational accuracy and computational cost. Memory injections can be used both as a tool to causally localize sources of model behavior and to provide inference-time corrections to unwanted/poor model behavior. Memory injections have the benefit of being a human-interpretable tool. We demonstrate a concrete use case for memory injections in the case of multi-hop reasoning. We employ memory injections to augment a language model's knowledge recall capacity during multi-hop reasoning tasks and show an improvement in downstream reasoning performance. Future work can consider extending applying memory injections to remove unwanted or harmful information from a model's residual stream during inference, developing automated memory selection workflows, and further exploring better representational schemes for encoding memories.

`Attention Lens` allows users to train probes into attention heads of a neural network, further elucidating *how* models arrive at their final output predictions. Lenses can be trained

to perform many different types of tasks. In this work, we train attention-head specific lenses for GPT2-Small. We demonstrate three use cases for these lenses: bias localization, malicious prompt detection, and activation engineering/model editing. Future work can consider training lenses for more models, using lenses to localize harmful behavior, and guide developing mitigations/corrective strategies for large pre-trained models.

REFERENCES

Laura Aina and Tal Linzen. The language model understood the prompt was ambiguous: Probing syntactic uncertainty through generation. *arXiv preprint arXiv:2109.07848*, 2021.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Ehsan Amid, Rohan Anil, Wojciech Kotłowski, and Manfred K. Warmuth. Learning from randomly initialized neural network features, 2022.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*, 2019.

Konstantine Arkoudas. GPT-4 can't reason. *arXiv preprint arXiv:2308.03762*, 2023. doi:10.48550/arXiv.2308.03762.

Maximiliana Behnke and Kenneth Heafield. Losing heads in the lottery: Pruning transformer attention in neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2664–2674, 2020.

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

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023. doi:10.48550/arXiv.2303.08112.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623, 2021.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687*, 2023. doi:10.48550/arXiv.2308.09687.

Andrew Blair-Stanek, Nils Holzenberger, and Benjamin Van Durme. Can GPT-3 perform statutory reasoning? *arXiv preprint arXiv:2302.06100*, 2023. doi:10.48550/arXiv.2302.06100.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 141–159. IEEE, 2021.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33: 1877–1901, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In *Conference on Empirical Methods in Natural Language Processing*, 2021. URL https://api.semanticscholar.org/CorpusID:233289412.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models, 2021.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=TatRHT_1cK.

Anthony Chen, Panupong Pasupat, Sameer Singh, Hongrae Lee, and Kelvin Guu. Purr: Efficiently editing language model hallucinations by denoising language model corruptions. *arXiv preprint arXiv:2305.14908*, 2023.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained bert networks. *Advances in neural information processing systems*, 33:15834–15846, 2020.

Wenhu Chen, Hexiang Hu, Chitwan Saharia, and William W Cohen. Re-imagen: Retrieval-augmented text-to-image generator. *arXiv preprint arXiv:2209.14491*, 2022.

Abhijith Chintam, Rahel Beloch, Willem Zuidema, Michael Hanna, and Oskar van der Wal. Identifying and adapting transformer-components responsible for gender bias in an English language model. In Yonatan Belinkov, Sophie Hao, Jaap Jumelet, Najoung Kim, Arya McCarthy, and Hosein Mohebbi, editors, *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 379–394, Singapore, December 2023. Association for Computational Linguistics. doi:10.18653/v1/2023.blackboxnlp-1.29. URL https://aclanthology.org/2023.blackboxnlp-1.29.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. Evaluating the ripple effects of knowledge editing in language models. *arXiv preprint arXiv:2307.12976*, 2023a. doi:10.48550/arXiv.2307.12976.

Roi Cohen, Mor Geva, Jonathan Berant, and Amir Globerson. Crawling the internal knowledge-base of language models. *arXiv preprint arXiv:2301.12810*, 2023b.

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

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, 2022a. doi:10.18653/v1/2022.acl-long.581.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers, 2022b.

Damai Dai, Wenbin Jiang, Qingxiu Dong, Yajuan Lyu, Qiaoqiao She, and Zhifang Sui. Neural knowledge bank for pretrained transformers. *arXiv preprint arXiv:2208.00399*, 2022c. doi:10.48550/arXiv.2208.00399.

Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. Analyzing transformers in embedding space. *arXiv preprint arXiv:2209.02535*, 2022. doi:10.48550/arXiv.2209.02535.

Mark Davies. The Corpus of Contemporary American English as the first reliable monitor corpus of English. *Literary and Linguistic Computing*, 25(4):447–464, 2010. doi:10.1093/llc/fqq018.

Mark Davies. Word frequency data from the Corpus of Contemporary American English (COCA), 2011. URL `https://www.english-corpora.org/coca/compare-bnc.asp`.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506, 2021. doi:10.18653/v1/2021.emnlp-main.522.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019. doi:10.18653/v1/N19-1423.

Bhuwan Dhingra, Jeremy R. Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W. Cohen. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 10:257–273, 2022. doi:10.1162/tacl_a_00459. URL https://aclanthology.org/2022.tacl-1.15.

Andrija Djurisic, Nebojsa Bozanic, Arjun Ashok, and Rosanne Liu. Extremely simple activation shaping for out-of-distribution detection. In *The Eleventh International Conference on Learning Representations*, 2022.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*, 2020.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031, 2021. doi:10.1162/tacl_a_00410. URL https://aclanthology.org/2021.tacl-1.60.

N Elhage, N Nanda, C Olsson, T Henighan, N Joseph, B Mann, A Askell, Y Bai, A Chen, T Conerly, et al. A mathematical framework for transformer circuits, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

Allyson Ettinger, Ahmed Elgohary, and Philip Resnik. Probing for semantic evidence of composition by means of simple classification tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 134–139, Berlin, Germany, August 2016. Association for Computational Linguistics. doi:10.18653/v1/W16-2524. URL https://aclanthology.org/W16-2524.

William A Falcon. Pytorch lightning. *GitHub*, 3, 2019.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891, 2020.

Stanislav Fort. Scaling laws for adversarial attacks on language model activations, 2023.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

Théophile Gaudin. *Algorithms for Self-driving Labs*. PhD thesis, 2023.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021a.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic, November 2021b. Association for Computational Linguistics. doi:10.18653/v1/2021.emnlp-main.446. URL `https://aclanthology.org/2021.emnlp-main.446`.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi:10.18653/v1/2022.emnlp-main.3. URL `https://aclanthology.org/2022.emnlp-main.3`.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. *arXiv preprint arXiv:2304.14767*, 2023. doi:10.48550/arXiv.2304.14767.

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

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. More than you've asked for: A comprehensive analysis of novel prompt injection threats to application-integrated large language models. *arXiv preprint arXiv:2302.12173*, 2023.

Taicheng Guo, Kehan Guo, Zhengwen Liang, Zhichun Guo, Nitesh V Chawla, Olaf Wiest, Xiangliang Zhang, et al. What indeed can GPT models do in chemistry? a comprehensive benchmark on eight tasks. *arXiv preprint arXiv:2305.18365*, 2023. doi:10.48550/arXiv.2305.18365.

Song Han, Junlong Kang, Huizi Mao, Yiming Hu, Xin Li, Yubin Li, Dongliang Xie, Hong Luo, Song Yao, Yu Wang, et al. Ese: Efficient speech recognition engine with sparse lstm on fpga. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 75–84, 2017.

Momchil Hardalov, Ivan Koychev, and Preslav Nakov. Towards automated customer support. In *Artificial Intelligence: Methodology, Systems, and Applications: 18th International Conference, AIMSA 2018, Varna, Bulgaria, September 12–14, 2018, Proceedings 18*, pages 48–59. Springer, 2018.

Peter Hase, Mona Diab, Asli Celikyilmaz, Xian Li, Zornitsa Kozareva, Veselin Stoyanov, Mohit Bansal, and Srinivasan Iyer. Do language models have beliefs? Methods for detecting, updating, and visualizing model beliefs. *arXiv preprint arXiv:2111.13654*, 2021. doi:10.48550/arXiv.2111.13654.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? Surprising differences in causality-based localization vs. knowledge editing in language models. *arXiv preprint arXiv:2301.04213*, 2023. doi:10.48550/arXiv.2301.04213.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi:10.18653/v1/2020.coling-main.580. URL `https://aclanthology.org/2020.coling-main.580`.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Yifan Hou, Jiaoda Li, Yu Fei, Alessandro Stolfo, Wangchunshu Zhou, Guangtao Zeng, Antoine Bosselut, and Mrinmaya Sachan. Towards a mechanistic interpretation of multi-step reasoning capabilities of language models. *arXiv preprint arXiv:2310.14491*, 2023.

Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*, 2022.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. Transformer-patcher: One mistake worth one neuron. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=4oYUGeGBPm`.

Kevin Maik Jablonka, Qianxiang Ai, Alexander Al-Feghali, Shruti Badhwar, Joshua D Bocarsly, Andres M Bran, Stefan Bringuier, L Catherine Brinson, Kamal Choudhary, Defne Circi, et al. 14 examples of how llms can transform materials science and chemistry: a reflection on a large language model hackathon. *Digital Discovery*, 2(5):1233–1250, 2023.

Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun KIM, Stanley Jungkyu Choi, and Minjoon Seo. Towards continual knowledge learning of language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=vfsRB5MImo9.

Jinhao Jiang, Kun Zhou, Xin Zhao, and Ji-Rong Wen. UniKGQA: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=Z63RvyAZ2Vh.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438, 2020.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*, 2023b. doi:10.48550/arXiv.2305.06983.

Nikhil Kandpal, Eric Wallace, and Colin Raffel. Deduplicating training data mitigates privacy risks in language models, 2022.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. Large language models struggle to learn long-tail knowledge, 2023a.

Nikhil Kandpal, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. Backdoor attacks for in-context learning with language models. *arXiv preprint arXiv:2307.14692*, 2023b.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Shahar Katz, Yonatan Belinkov, Mor Geva, and Lior Wolf. Backward lens: Projecting language model gradients into the vocabulary space. *arXiv preprint arXiv:2402.12865*, 2024.

Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Najoung Kim, Roma Patel, Adam Poliak, Alex Wang, Patrick Xia, R Thomas McCoy, Ian Tenney, Alexis Ross, Tal Linzen, Benjamin Van Durme, et al. Probing what different nlp tasks teach machines about function word comprehension. *arXiv preprint arXiv:1904.11544*, 2019.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Mojtaba Komeili, Kurt Shuster, and Jason Weston. Internet-augmented dialogue generation. *arXiv preprint arXiv:2107.07566*, 2021.

Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=UYneFzXSJWh`.

Kevin Lee and Shubho Sengupta. Introducing the ai research supercluster - meta's cutting-edge ai supercomputer for ai research. URL `https://ai.meta.com/blog/ai-rsc/`.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. *arXiv preprint arXiv:2210.13382*, 2022.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. PMET: Precise model editing in a transformer. *arXiv preprint arXiv:2308.08742*, 2023. doi:10.48550/arXiv.2308.08742.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*, 2017.

Shangqing Liu, Yu Chen, Xiaofei Xie, Jingkai Siow, and Yang Liu. Retrieval-augmented generation for code summarization via hybrid gnn. *arXiv preprint arXiv:2006.05405*, 2020.

Jieyi Long. Large language model guided tree-of-thought. *arXiv preprint arXiv:2305.08291*, 2023. doi:10.48550/arXiv.2305.08291.

Pratyush Maini, Michael C Mozer, Hanie Sedghi, Zachary C Lipton, J Zico Kolter, and Chiyuan Zhang. Can neural network memorization be localized? *arXiv preprint arXiv:2307.09542*, 2023.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`, 2022.

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

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 35: 17359–17372, 2022a. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/6f1d43d5a82a37e89b0665b33bf3a182-Paper-Conference.pdf.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*, 2022b. doi:10.48550/arXiv.2210.07229.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers. *arXiv preprint arXiv:2106.15772*, 2021.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=0DcZxeWfOPt.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=0DcZxeWfOPt.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. Memory-based model editing at scale. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 15817–15831. PMLR, 17–23 Jul 2022c. URL https://proceedings.mlr.press/v162/mitchell22a.html.

Moin Nadeem, Anna Bethke, and Siva Reddy. Stereoset: Measuring stereotypical bias in pretrained language models. *arXiv preprint arXiv:2004.09456*, 2020.

Neel Nanda and Joseph Bloom. TransformerLens, 2022. URL https://github.com/neelnanda-io/TransformerLens.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=9XFSbDPmdW.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=9XFSbDPmdW.

Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. *arXiv preprint arXiv:2309.00941*, 2023c.

Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from (production) language models, 2023.

nostalgebraist. Logit Lens on non-GPT2 models + extensions, 2021. URL `https://colab.research.google.com/drive/1MjdfK2srcerLrAJDRaJQKO0sUiZ-hQtA`.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

TB OpenAI. Chatgpt: Optimizing language models for dialogue. openai, 2022.

Oded Ovadia, Menachem Brief, Moshik Mishaeli, and Oren Elisha. Fine-tuning or retrieval? comparing knowledge injection in llms. *arXiv preprint arXiv:2312.05934*, 2023.

Koyena Pal, Jiuding Sun, Andrew Yuan, Byron C Wallace, and David Bau. Future lens: Anticipating subsequent tokens from a single hidden state. *arXiv preprint arXiv:2311.04897*, 2023.

Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. Task-specific skill localization in fine-tuned language models. *arXiv preprint arXiv:2302.06600*, 2023.

John Arthur Passmore. Philosophical reasoning. 1961.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*, 2021.

Vaidehi Patil, Peter Hase, and Mohit Bansal. Can sensitive information be deleted from llms? objectives for defending against extraction attacks, 2023.

Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models. *arXiv preprint arXiv:2211.09527*, 2022.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to!, 2023.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019a. `https://paperswithcode.com/paper/language-models-are-unsupervised-multitask`.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019b. URL `https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf`.

Inioluwa Deborah Raji, Emily M Bender, Amandalynne Paullada, Emily Denton, and Alex Hanna. Ai and the everything in the whole wide world benchmark. *arXiv preprint arXiv:2111.15366*, 2021.

Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*, 2020.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413*, 2016.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Mansi Sakarvadia, Aswathy Ajith, Arham Khan, Daniel Grzenda, Nathaniel Hudson, André Bauer, Kyle Chard, and Ian Foster. Memory injections: Correcting multi-hop reasoning failures during inference in transformer-based language models. *arXiv preprint arXiv:2309.05605*, 2023.

Bilgehan Sel, Ahmad Al-Tawaha, Vanshaj Khattar, Lu Wang, Ruoxi Jia, and Ming Jin. Algorithm of thoughts: Enhancing exploration of ideas in large language models. *arXiv preprint arXiv:2308.10379*, 2023.

Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. Detecting pretraining data from large language models, 2024.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi:10.18653/v1/D18-1455. URL `https://aclanthology.org/D18-1455`.

Yiyou Sun, Chuan Guo, and Yixuan Li. React: Out-of-distribution detection with rectified activations. *Advances in Neural Information Processing Systems*, 34:144–157, 2021.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2018.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Alex Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte MacDiarmid. Activation addition: Steering language models without optimization. *arXiv preprint arXiv:2308.10248*, 2023.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *arXiv preprint arXiv:2305.04388*, 2023. doi:10.48550/arXiv.2305.04388.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017. URL `https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33:12388–12401, 2020.

Boshi Wang, Xiang Deng, and Huan Sun. Iteratively prompt pre-trained language models for chain of thought. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2714–2730, Abu Dhabi, United Arab Emirates, December 2022a. Association for Computational Linguistics. doi:10.18653/v1/2022.emnlp-main.174. URL `https://aclanthology.org/2022.emnlp-main.174`.

Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, et al. Decodingtrust: A comprehensive assessment of trustworthiness in gpt models. 2023a.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: A circuit for indirect object identification in GPT-2 Small. *arXiv preprint arXiv:2211.00593*, 2022b.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, et al. Knowledge editing for large language models: A survey. *arXiv preprint arXiv:2310.16218*, 2023b.

Wenxuan Wang, Juluan Shi, Zhaopeng Tu, Youliang Yuan, Jen-tse Huang, Wenxiang Jiao, and Michael R Lyu. The earth is flat? unveiling factual errors in large language models. *arXiv preprint arXiv:2401.00761*, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023c. URL https://openreview.net/forum?id=1PL1NIMMrw.

Mohammad Wardat, Wei Le, and Hridesh Rajan. Deeplocalize: Fault localization for deep neural networks. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 251–262. IEEE, 2021.

Peter Cathcart Wason and Philip Nicholas Johnson-Laird. *Psychology of reasoning: Structure and content*, volume 86. Harvard University Press, 1972.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022a.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022b. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

Jennifer C White, Tiago Pimentel, Naomi Saphra, and Ryan Cotterell. A non-linear structural probe. *arXiv preprint arXiv:2105.10185*, 2021.

Amy Winograd. Loose-lipped large language models spill your secrets: The privacy implications of large language models. *Harvard Journal of Law & Technology*, 36(2), 2023.

Xinwei Wu, Junzhuo Li, Minghui Xu, Weilong Dong, Shuangzhi Wu, Chao Bian, and Deyi Xiong. Depn: Detecting and editing privacy neurons in pretrained language models. *arXiv preprint arXiv:2310.20138*, 2023.

Yinjun Wu, Edgar Dobriban, and Susan Davidson. Deltagrad: Rapid retraining of machine learning models. In *International Conference on Machine Learning*, pages 10355–10366. PMLR, 2020.

Tong Xie, Yuwei Wa, Wei Huang, Yufei Zhou, Yixuan Liu, Qingyuan Linghu, Shaozhou Wang, Chunyu Kit, Clara Grazian, and Bram Hoex. Large language models as master key: Unlocking the secrets of materials science with GPT. *arXiv preprint arXiv:2304.02213*, 2023a. doi:10.48550/arXiv.2304.02213.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. Decomposition enhances reasoning via self-evaluation guided decoding. *arXiv preprint arXiv:2305.00633*, 2023b. doi:10.48550/arXiv.2305.00633.

Jianhao Yan, Futing Wang, Yafu Li, and Yue Zhang. Potential and challenges of model editing for social debiasing. *arXiv preprint arXiv:2402.13462*, 2024.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023. doi:10.48550/arXiv.2305.10601.

Jiseon Yun, Jae Eui Sohn, and Sunghyon Kyeong. Fine-tuning pretrained language models to enhance dialogue summarization in customer service centers. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, pages 365–373, 2023.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5773–5784, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi:10.18653/v1/2022.acl-long.396. URL `https://aclanthology.org/2022.acl-long.396`.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*, 2024.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. Siren's song in the ai ocean: A survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*, 2023.

Zexuan Zhong, Dan Friedman, and Danqi Chen. Factual probing is [mask]: Learning vs. learning to recall. *arXiv preprint arXiv:2104.05240*, 2021.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. *arXiv preprint arXiv:2305.14795*, 2023. doi:https://doi.org/10.48550/arXiv.2305.14795.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. Modifying memories in transformer models, 2020.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *IEEE International Conference on Computer Vision*, pages 19–27, 2015.