Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Understanding and Controlling the Activations of Language Models

Author:
Ole Jorgensen

Supervisors:
Murray Shanahan, Nandi Schoots and Dylan Cope

Submitted in partial fulfillment of the requirements for the MSc degree in MSc in Artificial Intelligence of Imperial College London

September 2023

Contents

Abstract

A description of decoder-only transformer models is presented, including a brief summary of the impact of LayerNorm on the latent space. A background report summarising the current state-of-the-art on language model interpretability is then conducted, focusing on how better understanding the structure of activations could lead to better controlling language models.

This is followed by an empirical investigation into how different features are represented by language models, using multiple methods to demonstrate similarities and differences between the activations associated with different datasets. Previous work concerning whether Residual Stream activations are centred about the origin is extended, providing a negative answer.

These empirical findings are complimented by an investigation into how to approximate the feature representations of language models, using the hypothesis that features of text in GPT-2 style models are represented as directions in activation space. This presents a potential improvement over existing methods.

This more principled understanding of the activations of language models allows for the improvement of new methods for controlling language models, by adding more precisely constructed vectors at inference time. A new method is proposed based on constructing feature vectors from datasets of text, with a solution to account for the bias in Residual Stream activations. Another method is presented which uses similar ideas to classify text using only the intermediate activations of language models. Some examples of these methods in use are demonstrated.

Acknowledgements

I would like to extend a huge amount of gratitude to all three of my supervisors: Murray Shanahan, Nandi Schoots, and Dylan Cope. They have all been incredibly generous with their time, and were very supportive whenever I decided to explore a new research direction. They contributed a great deal to my enjoyment of the project, and I struggle to imagine a better supervision arrangement.

I would also like to thank Open Philanthropy for providing the funding which enabled me to complete this degree, and the London Effective Altruism Hub for providing office space for me to work on this project.

Finally, I would like to thank my family and friends for all their support this year. In particular I'd like to thank my mum and sister for their constant support.

Chapter 1

Introduction

1.1 Understanding Language Models

The current paradigm of deep learning has led to significant progress in a range of machine learning tasks, from computer vision to natural language processing. However, although we know how these systems are trained, we currently lack comprehensive tools for verifying the behaviour of these systems [1]. The black-box nature of these systems means that we cannot have confidence in their reliability, which is crucial if they are to be used in high-stakes scenarios such as medicine or security.

There is a growing body of work that tries to interpret the underlying algorithms implemented by machine learning algorithms [2], although this work has so far proved insufficient for understanding the behaviour of state-of-the-art systems. This is particularly true of large language models such as GPT-4 [3] and Palm-2 [4], where models have demonstrated a fast increase in capabilities with little progress in understanding the algorithms learned by these systems.

Some previous work on neural network interpretability, such as Olah et al. [5], Goh et al. [6], and Radford et al. [7], has focused on studying the structure of their latent activations. In particular, it was demonstrated that a range of neural network architectures often learned to represent features of their inputs in their latent representations. In some contexts, this knowledge of the structure of the activations could be exploited to better control the behaviour of these models [8]. However, these methods and results were highly dependent on the structure of each network, and analogous investigations for transformer language models have yet to reach the same level of sophistication as investigations into other architectures. This thesis aims to address this gap in the literature, taking inspiration from work on other architectures to better understand the structure of the activations of language models. It further aims to develop new methods that improve our understanding and control of large language models, addressing concerns about their safety and trustworthiness.

Specifically, we will perform an exploratory analysis of the activations of GPT-2 style language models. We selected GPT-2 style models due to their architectural resemblance to more powerful successors, such as GPT-3. This suggests that insights and methods gleaned from GPT-2 could be applicable to more advanced models. Open-source versions of GPT-2 small, medium, large and XL will be investigated, using tooling provided by Nanda [9].

1.2 My Contributions

The contributions of this thesis are threefold. Our findings suggest that **the activations of language models are highly structured**, sharing similarities across different textual examples which share some common feature. Different aspects of this structure are analysed in detail, including the result that activations across all GPT-2 models, for all layers, are not centred about the origin.

The second contribution uses this understanding of the structure of language model activations to develop new **methods of forming feature vectors**. This offers a new way to find vectors which correspond to how transformers represent features of text, which could prove useful for better controlling language models.

Indeed, the third contribution of this thesis is to use these methods of forming feature vectors to **extend methods for controlling language models**. This involves the description and demonstration of **two new exploratory methods for manipulating and analysing language models**, which extends previous algorithms designed for language and image models.

The structure of this thesis is as follows:

Chapter 2 presents a description of decoder-only transformers, aiming to provide better intuitions for each part of the architecture. LayerNorm is also treated in some depth, which is often ignored in other expositions on transformers.

Chapter 3 provides a review of the literature relevant to our investigation. It will start with a broad overview of transformer models and trustworthiness, before surveying current progress in interpretability of neural networks and transformers. This will culminate in a review of work in activation interpretability, presenting work which will directly inform the ensuing investigations.

Chapter 4 presents experimental methods that can be used to obtain further empirical evidence for how different inputs are processed and manipulated by transformer models. This is followed by a series of investigations into how different datasets are processed by GPT-2 models, providing evidence that there are significant similarities in activations across inputs which share some human-interpretable features. Further similarities and differences across activations are investigated.

Chapter 5 briefly argues that human-interpretable features are represented as directions in the Residual Stream of GPT-2 style models. Assuming this hypothesis leads to a new method for developing feature vectors: directions in the Residual Stream which correspond to features of text. Mirroring the techniques developed for generative image models by White [8], two new techniques are developed to better understand and control language models: Feature Classification and Activation Additions. We demonstrate some early successful applications of these techniques.

Chapters 6 and 7 evaluate the implications and limitations of our work, suggesting avenues for further development as well as potential applications for improving the trustworthiness of language models.

1.3 Ethical Considerations

The following table outlines a list of legal, social, ethical and professional considerations related to my project.

Due to the theoretical nature of my project, there were few immediate ethical concerns. The only consideration that applied was the use of elements that may cause harm to the environment, since remote GPUs were used during the project to perform inference on large language models. Bender et al. [10] highlighted issues surrounding the carbon emissions associated with running inference on large language models, but this will have been relatively minimal for my own project since only single GPUs were used for relatively short periods of time, as training was not performed.

	Yes	**No**
Section 1: HUMAN EMBRYOS/FOETUSES		
Does your project involve Human Embryonic Stem Cells?		X
Does your project involve the use of human embryos?		X
Does your project involve the use of human foetal tissues / cells?		X
Section 2: HUMANS		
Does your project involve human participants?		X
Section 3: HUMAN CELLS / TISSUES		
Does your project involve human cells or tissues? (Other than from "Human Embryos/Foetuses" i.e. Section 1)?		X
Section 4: PROTECTION OF PERSONAL DATA		
Does your project involve personal data collection and/or processing?		X
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?		X
Does it involve processing of genetic information?		X
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.		X
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing data sets?		X
Section 5: ANIMALS		
Does your project involve animals?		X
Section 6: DEVELOPING COUNTRIES		
Does your project involve developing countries?		X
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?		X
Could the situation in the country put the individuals taking part in the project at risk?		X
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY		
Does your project involve the use of elements that may cause harm to the environment, animals or plants?	X	
Does your project deal with endangered fauna and/or flora /protected areas?		X
Does your project involve the use of elements that may cause harm to humans, including project staff?		X

	Yes	No
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?		X
Section 8: DUAL USE		
Does your project have the potential for military applications?		X
Does your project have an exclusive civilian application focus?		X
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?		X
Does your project affect current standards in military ethics – e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?		X
Section 9: MISUSE		
Does your project have the potential for malevolent/criminal/terrorist abuse?		X
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?		X
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?		X
Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?		X
SECTION 10: LEGAL ISSUES		
Will your project use or produce software for which there are copyright licensing implications?		X
Will your project use or produce goods or information for which there are data protection, or other legal implications?		X
SECTION 11: OTHER ETHICS ISSUES		
Are there any other ethics issues that should be taken into consideration?		X

Chapter 2

Transformer Background

This chapter will provide an explanation of how GPT-2 style transformers work, defining various pieces of terminology that will be used throughout the rest of this thesis.

2.1 GPT-2 Training and Inference

Decoder-only models such as GPT-2 style models are typically trained through next token prediction. This means that cross-entropy loss is applied to each token, with the correct label being the next token in the tokenised text.

These models can hence be used autoregressively at inference time by taking the argmax of the output associated with the final token. This corresponds to the model's prediction of the subsequent token. This can be repeated autoregressively to form longer passages of text.

Subsequent models such as GPT-3.5 and GPT-4 are trained using this pre-training followed by further training through processes such as Reinforcement Learning from Human Feedback [11]. This helps to make models more aligned with human preferences, and not just repeat the most likely continuations of text as found in the training set. However, *base models* (such as all the GPT-2 models) are only trained using pre-training.

Understanding how models were trained, and on what datasets, can be useful when trying to understand the capabilities and failures of different models. For example, only using pre-training means that models are particularly likely to demonstrate biases which are present in their training data, such as anti-Muslim biases [12].

2.2 GPT-2 Transformer Architecture

There is some variety in how different transformers are implemented. For my project, I will only consider GPT-2 style transformers. The following description draws heavily from Elhage et al. [13].

The forward pass of GPT-2 style models are computed as follows:

- Given an input string s, use a tokeniser to split this into a tensor of tokens $\mathbf{t} \in \mathbb{R}^{d_{\text{vocab}} \times l}$, where l is the number of tokens in $\mathbf{t}$. Each token can be considered a section of the string which the model is capable of processing.

- Embed these tokens using an embedding matrix and positional encoding $\mathbf{W}_E \in \mathbb{R}^{d_{\text{model}} \times d_{\text{vocab}}}$:

$$\mathbf{x}_0 = \mathbf{W}_E \cdot \mathbf{t},$$

  where $\mathbf{x}_0 \in \mathbb{R}^{d_{\text{model}} \times l}$, since it constitutes l vectors of dimension $\mathbb{R}^{d_{\text{model}}}$, where each vector corresponds to a single token of t.

- For $i \in \{0, 1, \ldots, n-1\}$, where n is the number of layers in the transformer:
  - Compute the output of the *Attention Layer*:

$$\mathbf{x}'_i = \mathbf{x}_i + \sum_{h \in H_i} h(\text{LayerNorm}(\mathbf{x}_i)),$$

    where each h is known as an *Attention Head*.
  - Compute the output of the *MLP Layer*:

$$\mathbf{x}_{i+1} = \mathbf{x}'_i + m_i(\text{LayerNorm}(\mathbf{x}'_i)).$$

  - Note that both $\mathbf{x}_i, \mathbf{x}'_i \in \mathbb{R}^{d_{\text{model}} \times l}$.

- Produce the final logits by applying the unembedding matrix $\mathbf{W}_U \in \mathbb{R}^{d_{\text{vocab}} \times d_{\text{model}}}$:

$$\mathbf{T}(\mathbf{t}) = \mathbf{W}_U \cdot \text{LayerNorm}(\mathbf{x}_{n-1}),$$

  where $\mathbf{T}(\mathbf{t}) \in \mathbb{R}^{d_{\text{vocab}} \times l}$.

- Produce the final output probabilities by taking a softmax over the logits:

$$\mathbf{P}(\mathbf{t}) = \text{softmax}(\mathbf{T}(\mathbf{t})),$$

  where again $\mathbf{P}(\mathbf{t}) \in \mathbb{R}^{d_{\text{vocab}} \times l}$. The final column of $P(t)$ implicitly corresponds to the model's predicted probability distribution over which token follows the input string.

This process is visualised in Figure 2.1.

2.2.1 Terminology

We will define some terminology which will be used throughout the thesis. It will draw from the terminology developed by Elhage et al. [13].

- *Activations* are any vectors which are computed during the forward pass of the model on some input.
- In particular, each $\mathbf{x}_i$ and $\mathbf{x}'_i$ are *Residual Stream activations*, or the *Residual Stream tensors* at the layer i. These tensors can be described as being located in the *Residual Stream*.
- Each $\mathbf{x}_i$ can be expressed as $\mathbf{x}_i = (\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,l})$, where each $\mathbf{x}_{i,j} \in \mathbb{R}^{d_{\text{model}}}$ corresponds to the activations associated with the j'th token. In some contexts, these will just be referred to as *tokens*.
- Individual vectors $\mathbf{x}_{i,j}$ are referred to as *activations of a token* for the i'th layer of the Residual Stream.
- The *final activations* in a layer are considered activations corresponding to the final token at some layer l of the transformer, i.e. $\mathbf{x}_{i,l}$ for some string s_k.

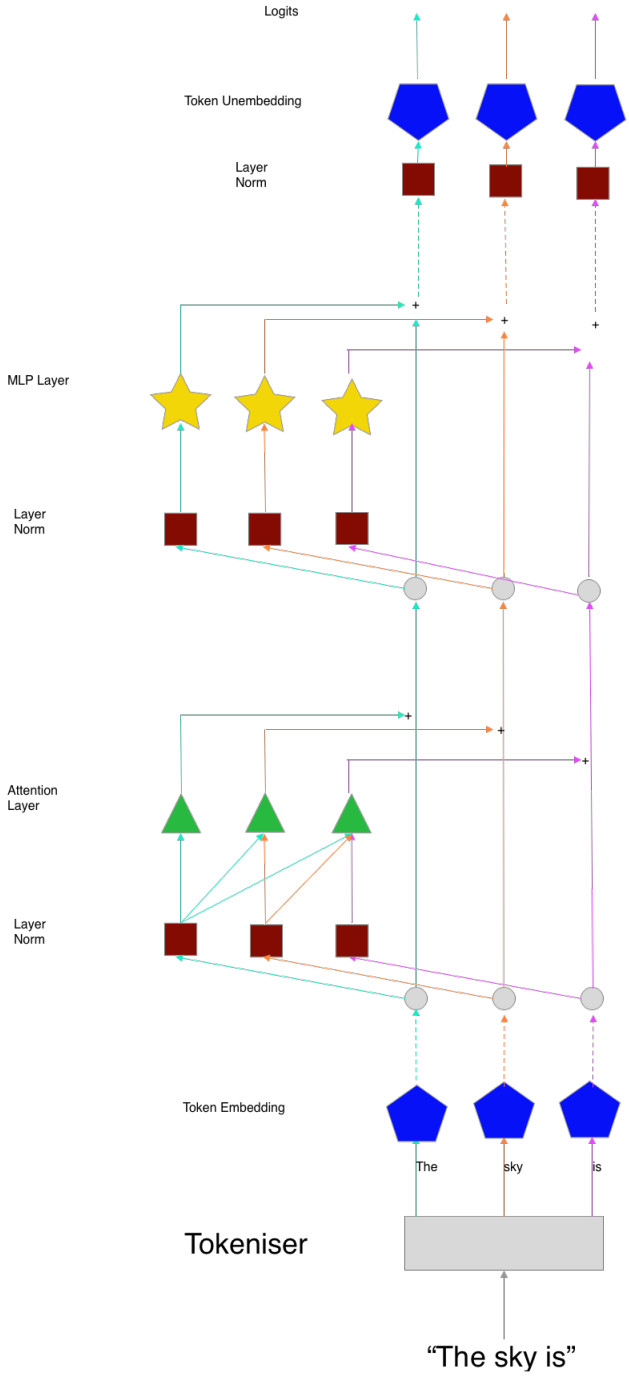

Figure 2.1: A diagram illustrating how GPT-2 style transformers work. The string "The sky is" is first split into three tokens, which are then embedded into the Residual Stream. The dimension of the Residual Stream is thus 3 x d_{model}. The Residual Stream then passes through n_{layers} transformer blocks, before finally passing through Layer Norm and the Unembedding Layer. The dimension of the final output is 3 x d_{vocab}.

2.2.2 Residual Stream

In GPT-2 style transformers, information can only be shared between different sections of the transformer through the Residual Stream. The input to each Attention Layer and MLP Layer is a LayerNormed copy of the Residual Stream, and the output of each of these layers is added back to the Residual Stream. Eventually, the Residual Stream is unembedded to produce the final predictions of the model.

Since information can only be passed through layers of the transformer via the Residual Stream, Residual Stream activations are very important to understand when trying to interpret transformers. These activations will provide the main source of our analysis in this thesis.

2.2.3 Layer Norm

As demonstrated in Figure 2.1, LayerNorm is applied to the Residual Stream every time it is used by other parts of the transformer. Hence, it is important to understand the impact of LayerNorm to understand how the model uses the Residual Stream. As seen above, LayerNorm is applied before every intermediate calculation occurs in GPT-2 models. Hence, we can simplify the space we need to consider by only considering what happens after LayerNorm has been applied.

Note that LayerNorm is applied to each Residual Stream vector in parallel. Thus, it can be understood by considering its behaviour on each Residual Stream token individually. Hence, throughout this section $\mathbf{x} \in \mathbb{R}^{d_{\text{model}}}$ is a vector corresponding to a single token in the Residual Stream.

The equation for LayerNorm is typically defined as follows:

Definition 1. *If $\mathbf{x} \in \mathbb{R}^{d_{model}}$, let LayerNorm be defined as*

$$\textit{LayerNorm}(\mathbf{x}) := \frac{\mathbf{x} - \mathbb{E}[\mathbf{x}]}{\sqrt{\mathbb{V}[\mathbf{x}] + \epsilon}} \cdot \gamma + \beta,$$

where

$\mathbb{E}[\mathbf{x}]$ *is the empirical mean over the components of* $\mathbf{x} = (x_i)_{i=1}^{d_{model}}$,

$$\textit{i.e., } \mathbb{E}[\mathbf{x}] := \frac{1}{d_{model}} \sum_{i=1}^{d_{model}} x_i;$$

$\mathbb{V}[\mathbf{x}]$ *is the biased empirical variance over the components of* $\mathbf{x}$,

$$\textit{i.e., } \mathbb{V}[\mathbf{x}] := \frac{1}{d_{model}} \sum_{i=1}^{d_{model}} (x_i - \mathbb{E}[\mathbf{x}])^2;$$

γ *is a diagonal matrix of dimension* $d_{model} \times d_{model}$,

β *is a vector of dimension* d_{model}.

We can also define the following, which is a simplification that removes the affine map post-normalisation.

Definition 2. *If $\mathbf{x} \in \mathbb{R}^{d_{model}}$, let PreLayerNorm be defined as*

$$\textit{PreLayerNorm}[\mathbf{x}] := \frac{\mathbf{x} - \mathbb{E}[\mathbf{x}]}{\sqrt{\mathbb{V}[\mathbf{x}] + \epsilon}},$$

Because the immediate operations following LayerNorm are always linear in GPT-2 style transformers, we can reduce this to instead considering PreLayerNorm, by "folding" the affine transformation after normalisation into the next operations.

Theorem 1. *If $f : \mathbb{R}^n \to \mathbb{R}^m$ is an affine transformation, then there exists an affine transformation $g : \mathbb{R}^n \to \mathbb{R}^m$ such that $f(LayerNorm[\mathbf{x}]) = g(PreLayerNorm[\mathbf{x}])$ for all $\mathbf{x} \in \mathbb{R}^m$*

Proof. The proof is straightforward. Given

$$f(\text{LayerNorm}(\mathbf{x})) = f\left(\text{PreLayerNorm}(\mathbf{x}) \cdot \gamma + \beta\right),$$

we can expand this as

$$f(\text{PreLayerNorm}(\mathbf{x})) \cdot \gamma + \beta,$$

since f is affine. Then, defining

$$g(\mathbf{y}) := f(\mathbf{y}) \cdot \gamma + \beta,$$

we find that g is the required affine map.

□

Thus, we only need to consider the simplification PreLayerNorm. We will make the simplifying assumption that this is approximately equivalent to normalising without the ϵ term. This is reasonable since this term is often in the region of 10^{-5}, and is only included for numerical stability.

Now if we consider the image of PreLayerNorm on $\mathbb{R}^n$, we have the following result.

Theorem 2. *PreLayerNorm[$\mathbb{R}^d_{model}$] corresponds to a sphere on the hyperplane orthogonal to the unit vector $\mathbf{1} = (1, ..., 1) \in \mathbb{R}^{d_{model}}$, centred at the origin with radius $\sqrt{d_{model}}$.*

Adapted from [14]. Given the operation $\mathbf{x} - \mathbb{E}[\mathbf{x}]$, it serves as a projection onto the orthogonal complement of the vector $\mathbf{1}$. This is evident since, defining

$$p_1(\mathbf{x}) := \frac{\langle \mathbf{x}, \mathbf{1} \rangle \mathbf{1}}{||\mathbf{1}||^2},$$

the mapping $\mathbf{x} - p_1(\mathbf{x})$ also projects onto the orthogonal complement of $\mathbf{1}$. Thus, we have:

$$\begin{aligned}
\mathbf{x} - \mathbb{E}[\mathbf{x}] &= (x_i - \frac{1}{d_{\text{model}}} \sum_j x_j)_i \\
&= \mathbf{x} - \left(\frac{1}{d_{\text{model}}} \sum_j x_j \right) \mathbf{1} \\
&= \mathbf{x} - \frac{\langle \mathbf{x}, \mathbf{1} \rangle \mathbf{1}}{||\mathbf{1}||^2} \\
&= \mathbf{x} - p_1(\mathbf{x}).
\end{aligned}$$

Further, discarding the factor ϵ (added typically for numerical stability and being insignificantly small), the expression for PreLayerNorm becomes:

$$\frac{\mathbf{x} - \mathbb{E}[\mathbf{x}]}{\sqrt{\mathbb{V}[\mathbf{x}]}} = \frac{p_1(\mathbf{x})}{\sqrt{\mathbb{E}[(\mathbf{x} - \mathbb{E}[\mathbf{x}])^2]}}$$

$$= \frac{p_1(\mathbf{x})}{\sqrt{\frac{1}{d_{\text{model}}} \sum_i p_1(\mathbf{x})_i^2}}$$
$$= \sqrt{d_{\text{model}}}\, p_s(p_1(\mathbf{x})),$$

where p_s denotes the projection onto the unit sphere.

In conclusion, we can express:

$$\text{PreLayerNorm}(\mathbf{x}) = \sqrt{d_{\text{model}}}\, p_s(p_1(\mathbf{x})),$$

representing a projection onto the unit sphere of the hyperplane orthogonal to $\mathbf{1}$, subsequently scaled by d_{model}.

□

Together, Theorems 1 and 2 provide a new way of understanding LayerNorm: it projects all activations onto a scaled unit sphere, on some hyperplane. This can meaningfully change the behaviour of successive functions in the transformer block, so it will need to be considered when trying to understand transformer blocks.

Note also that intuitively, the key aspect here is the projection onto a sphere. Since we typically operate in high dimensional vector spaces, the projection to the hyperplane typically has little impact on the vectors (as measured by cosine similarity). Again due to Roger [14], if the coordinates of $\mathbf{x}$ are expressed in some basis with the normalised $\mathbf{1}$ normalised as the first vector in the basis, the projection p_1 only changes one coordinate of $\mathbf{x}$. Since d_{model} is generally in the range of $800 - 12,000$ for LLMs, the projection p_1 has minimal impact on the vector $\mathbf{x}$.

The implication of this is that **when understanding the behaviour of Attention and MLP Layers on the Residual Stream, it is only necessary to understand their behaviour on a sphere centred at the origin**.

2.2.4 Attention Layers

Recall that an Attention Layer is the summation of multiple Attention Heads. Each head operates in parallel on copies of Residual Stream tensors $\mathbf{x} \in \mathbb{R}^{l \times d_{\text{model}}}$. Specifically, the Attention Layer can be expressed as:

$$\sum_{h \in H_i} h(\text{LayerNorm}(\mathbf{x})).$$

This expression implies that to understand the behaviour of the entire Attention Layer, we only need to consider each Attention Head in isolation.

Using the notation provided in Elhage et al. [13], consider an Attention Head h defined as $h : \mathbb{R}^{l \times d_{\text{model}}} \to \mathbb{R}^{l \times d_{\text{model}}}$. The output of this Attention Head, when acting on a Residual Stream tensor $\mathbf{x} \in \mathbb{R}^{l \times d_{\text{model}}}$, can be examined by focusing on the output corresponding to the i-th position of the Residual Stream.

$$h(\mathbf{x})_i = \sum_{1 \le j \le i} \mathbf{A}^h_{i,j}(\mathbf{x}) * \mathbf{W}^h_{OV}(\mathbf{x}_j),$$

In this context, $\mathbf{x}_i$ is termed the *destination token*, as it corresponds to the token to which the output will be added. Conversely, the other tokens are referred to as *source tokens* when considering their impact on the output of the destination token.

Before defining the terms in this formula thoroughly, we will provide an intuition for what is happening here. The matrix $\mathbf{A}^h(\mathbf{x}) \in \mathbb{R}^{l\text{x}l}$ represents attention scores, where each row sums to 1. This means that the output is constructed by weighting each $\mathbf{W}^h_{OV}(\mathbf{x}_j)$ vector by an attention scalar. Notice how $\mathbf{A}^h$ and $\mathbf{W}^h_{OV}$ are independent functions, so they can be understood separately.

Firstly unpacking $\mathbf{A}^h$, given the Residual Stream tensor (after LayerNorm has been applied) $\mathbf{x} \in \mathbb{R}^{l \times d_{model}}$, the unmasked attention scores are determined through the weights of the Attention Head:

$$\mathbf{I} = \mathbf{x}^T \mathbf{W}^h_{QK} \mathbf{x} \in \mathbb{R}^{l \times l},$$

where $\mathbf{W}^h_{QK} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ is a matrix.

Note a potential issue with using unmasked attention scores is that tokens should only be able to attend to previous tokens: they cannot be allowed to use future tokens to make predictions, since this would make predicting future tokens trivial, and prevent the network from learning anything useful. Preventing tokens from attending to future tokens is known as *masking*. One method of masking is applying a mask $\mathbf{M} \in \mathbb{R}^{l \times l}$ to the unmasked attention scores. This is zero in the lower triangular values (including the diagonal) and $-\infty$ in the upper triangular values. The resulting attention scores are then $\mathbf{z} = \mathbf{I} + \mathbf{M}$. This has the effect of preventing destination tokens from attending to source tokens after the source token.

Finally, the attention pattern is determined by applying softmax, in order to ensure that the attention scores sum to 1. We also divide by $\sqrt{d_{\text{model}}}$, which has the impact of making the attention scores more uniform.

$$\mathbf{z}_i = (\mathbf{x}^T \cdot \mathbf{W}^h_{QK} \cdot \mathbf{x} + \mathbf{M})_i \in \mathbb{R}^l$$
$$\mathbf{A^h}(\mathbf{x})_i = \text{softmax}\left(\frac{\mathbf{z}_i}{\sqrt{d_{\text{model}}}}\right) \in \mathbb{R}^l$$

These are the attention scores given to each source token when considering the i'th token as the destination token. Looking at all destination tokens allows this to be visualised as a heatmap: the i'th row corresponds to the attention pattern associated with $\mathbf{A^h}(\mathbf{x})_i$, i.e. the attention scores for the i'th destination token. An example is demonstrated in Figure 2.2. Notice that it is lower-triangular, due to the causal mask $\mathbf{M}$.

and $\mathbf{W}^h_{OV}, \mathbf{W}^h_{QK} \in \mathbb{R}^{l \times d_{\text{model}}}$ are matrices.

Once the attention pattern $\mathbf{A}^h$ has been computed, the final output is simple to compute since $\mathbf{W}^h_{OV} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ is a matrix.

Important Points

When trying to understand Attention Heads, the most important thing to consider is that they are in many ways similar to linear maps. Indeed, if attention scores are held fixed, they are simply linear maps. Attention scores also involve bilinear maps, before softmax is applied. This will be explored further in Section 5.1 when trying to better understand the structure of activations in the Residual Stream.

Note also that Attention Layers are the only place in the transformer where information is moved between different tokens. MLP Layers, LayerNorm, and Unembedding layers operate on each token in the Residual Stream in parallel, and so no information is moved between tokens in these layers.

Figure 2.2: An attention heatmap on an induction head for a transformer performing a translation task. Darker colours correspond to larger scores, with the scores in each row summing to 1 due to softmax. Here we can see that for the destination token 'grand' (the highlighted row), the token being most strongly attended to is 'temple' (the bright red square in this row). We can hypothesise this is because the transformer has learned to attend to this token since it is the next word that needs to be translated. Taken from [15].

2.2.5 MLP Layers

MLP Layers are much simpler to describe than Attention Layers since they are simply one-layer feed-forward neural networks.

MLP Layers operate on each token in parallel (just as LayerNorm does). We can thus see MLP Layers as operating as $m(x) = (m(x_1), ..., m(x_l))$. Then, the operation $m : \mathbb{R}^{d_{\text{model}}} \to \mathbb{R}^{d_{\text{model}}}$ is defined as

Definition 3. *Given a single Residual Stream activation input $x \in \mathbb{R}^{d_{model}}$, define the function $m(x)$ as the composition*

$$m(x) = g(\sigma(f(x)))$$

where

$$\begin{aligned} f(x) &= \mathbf{W}_{in}^m x + \mathbf{b}_{in} && \text{for } \mathbf{W}_{in}^m \in \mathbb{R}^{d_{mlp} \times d_{model}}, \mathbf{b}_{in}^m \in \mathbb{R}^{d_{mlp}} \\ \sigma &= \textit{GeLU} && \textit{(activation function)} \\ g(z) &= \mathbf{W}_{out}^m z + \mathbf{b}_{out} && \text{for } \mathbf{W}_{out}^m \in \mathbb{R}^{d_{model} \times d_{mlp}}, \mathbf{b}_{out}^m \in \mathbb{R}^{d_{model}} \end{aligned}$$

Typically, $d_{\text{mlp}} = 4d_{\text{model}}$, but this isn't necessary and is just convention.

Chapter 3

Literature Review

This chapter presents a survey of literature relevant to this thesis. It briefly discusses current approaches to controlling language models, before examining how exploiting the feature representations used by generative image networks can be used to better control these models. We then consider how these techniques can be applied to language models, with an emphasis on existing work understanding the activation space of language models.

3.1 Transformers and Large Language Models

Most of the most capable systems developed in Machine Learning today are based on neural network architectures, which are trained on data with a loss function using a combination of methods such as backpropagation and stochastic gradient descent. One of the early successes of gradient-based approaches to training neural networks came in Lecun et al. [16], where convolutional neural networks were trained to achieve state-of-the-art performance for recognising handwritten digits. The capabilities of neural networks on computer vision tasks advanced rapidly throughout the early 2010s [17], in large part due to the continued development of GPUs.

Besides computer vision, another domain that emerged was that of sequence tasks: given a sequence of inputs, a model would be tasked with predicting the next element of the sequence. Language generation can also be seen as an instance of this problem: given text as input, text that a model predicts as output can be used to extend the text input.

The state-of-the-art architectures for language neural networks until 2017 were sequence models such as RNNs [18] and LSTMs [19]. However, these architectures had some issues, such as vanishing gradients and difficulty capturing long-range dependencies in text. The computations needed to train these models were also not easy to parallelise, meaning that these models didn't benefit as much from the use of GPUs as vision models like CNNs.

The introduction of the transformer architecture [20] contributed to solving these issues of vanishing gradients and inability to parallelise, leading to great advancements in language generation. There are several variations of the transformer architecture which have been used for different tasks:

- One architecture is the Masked Language Model (MLM), including the BERT family of models [21] which were designed to be fine-tuned to be used for various other tasks such as sentiment analysis. These are sometimes referred to as encoder-only models.

- Another class of models are Machine Translation Models (MT), which take in both a source sentence and the start of a translation as the destination before iteratively producing the translation. This was the architecture originally introduced by Vaswani et al. [20]. These models are sometimes referred to as encoder-decoder models.
- The final major class of models are Language Models (LM), which take in a single string as input and iteratively add to it. These models are sometimes referred to as decoder-only models. This thesis will focus primarily on decoder-only models, with more detail on this architecture presented in Section 2.2.

OpenAI has developed some of the most capable decoder-only language models, using increasingly large architectures for the GPT-1 [22], GPT-2 [23], GPT-3 [24] and GPT-4 [3] models.

One important fact about these models is that, empirically, it has been observed that training larger models on larger datasets leads to increasingly more capable models. This trend was formalised in Kaplan et al. [25], which demonstrated a power-law relationship between the loss of a language model and the total amount of compute used to train the model. This result was refined by Hoffmann et al. [26], presenting a compute-optimal ratio between dataset size and model parameters when optimising for model performance. Language models with sufficiently many parameters are often referred to as Large Language Models (LLMs).

3.2 Controlling Language Models

Despite the increase in capabilities of state-of-the-art ML systems, there are still many open questions surrounding how to control these systems. These issues pertain to both longer-term concerns, as well as pragmatic concerns around how to control LLMs to prevent them from being misused. A brief overview of some of these issues is presented in Amodei et al. [27].

These issues also apply to Large Language Models. Controlling the output of these models has proved difficult, although several techniques have been developed to address these shortcomings. One example is reinforcement learning from human feedback (RLHF) [11], which uses human feedback-based reinforcement learning to shape model behaviour. RLHF has proved to be practically useful for better controlling LLMs, although it suffers from various issues such as the difficulty of providing accurate feedback to models Casper et al. [28].

Relatedly, RLHF is a computationally intensive process, requiring many iterations of backpropagation as well as a large amount of feedback from humans (which can be costly to obtain). To make these methods less costly to implement, it is important to develop methods for controlling LLMs which might be less computationally expensive.

Extensions to RLHF have also been proposed, although they suffer from the same fundamental issues [29].

A different approach is known as prompt engineering, where variations of the input prompt are designed to better control the model. One prominent example of this is chain of thought (CoT) prompting [30], where models are prompted to explain their reasoning to improve the quality of their responses. Although this has been demonstrated to be practically useful, this method essentially only papers over the issues of controlling language models without dealing with them properly. Indeed, CoT prompting often elicits unfaithful explanations from models Turpin et al. [31]. Relying on prompt engineering also relies on users to use models as intended, which introduces the potential for misuse from prompt injections and jailbreaking.

3.3 Neural Network Interpretability

Neural network interpretability is a growing sub-field of machine learning research, focused on trying to describe the computations of neural networks in human understandable terms. Better understanding neural networks, beyond merely viewing them as black boxes, can also enhance our control over systems like LLMs.

3.3.1 Feature Representations

One promising finding in early neural network interpretability was a growing body of evidence that neural networks learned to represent features of inputs, using human-interpretable representations. The working definition of features used throughout this thesis is based on the definition provided by Ilyas et al. [32]:

Definition 4. *A feature is a a function mapping from the input space $\mathcal{X}$ to the real numbers $\mathbb{R}$, with the set of all features thus being $\mathcal{F} = f : \mathcal{X} \to \mathbb{R}$.*

One example of a feature of some text s might be a function that maps to 1 if an animal is described in the text, and 0 otherwise. This description is broad enough that any attribute of text can be described formally as a feature this way.

When trying to understand how neural networks utilise features of inputs, it is often desirable to try to create *feature representations*: subsets of activation space in a neural network which correspond to single, somewhat indivisible concepts. Being able to find the feature representations used by a network would allow you to understand what features of the input are important in determining the outputs of a network, which would be highly significant for a range of interpretability efforts.

Olah et al. [5] made valuable early contributions to understanding feature representations, locating and analysing neurons in CNNs which appeared to correspond to networks representing features internally. Another example was the existence of a neuron in a CNN corresponding to photos, drawings, and text describing Spiderman Goh et al. [6]. There is also empirical evidence for this in RNN-style models Radford et al. [7], where an LSTM trained to generate reviews was found to contain a single neuron which corresponded to the sentiment of the review.

Another significant example of networks learning to represent features is Mikolov et al. [33], where the authors trained a simple Recurrent Neural Network using word2vec embeddings before investigating the properties of the word embeddings. Mikolov et al. [33] showed that the embedding vectors often satisfied algebraic properties that suggested certain features were being represented in embedding space (such as $embed(queen) \approx embed(king) + embed(woman) - embed(man)$ suggesting that a woman vector exists).

Despite the above examples of successfully finding feature representations in neural networks, finding these feature representations was often resource-intensive, and no general method for finding all the features represented internally by networks has been proposed.

3.3.2 Stucture of Feature Representations

One of the major issues faced when trying to find feature representations in neural networks is trying to understand the kinds of mathematical objects which could best be used to describe

them. Naively, one might hope that features will always be represented by single neurons in networks. However in practise many neurons in CNNs were shown to be polysemantic, meaning that they correspond to multiple features of the input [5].

One potential explanation for polysemanticity is the phenomenon of superposition. The basic idea is that a model is often incentivised to store more features than it has dimensions in its latent representations. If the extra features are important enough then the model will be incentivised to try to store more features than it has neurons, causing it to compress its representation of these features. Elhage et al. [34] were the first to refer to this phenomenon as superposition, and were able to construct several examples of this occurring in toy models. The authors also conjecture that superposition occurs in large language models, due to the huge number of features it is useful for a model to be able to represent. This poses significant challenges when trying to understand large language models, since if superposition occurs then it will be impossible to construct a basis with each feature corresponding to a different basis vector.

Even assuming that superposition is a factor in how features are represented in transformers, it is not clear exactly how they are being represented. Elhage et al. [34] show that in simple linear models with a single ReLU activation function, features may be represented as directions in activation space. An alternative, proposed by Black et al. [35], is that convex regions of space might be used to represent features instead. There are some theoretical reasons to consider convex regions of activation space instead of directions: networks which use only ReLU functions for their non-linear activations can be understood as partitioning the input space into convex polytopes, and performing some simple linear function on each polytope [36]. Recent work has focused on generalising these results to other activation functions through an abstraction known as max-affine spline operators [37]. However, these generalisations and formalisms are based on classic feed-forward networks, with similar formalisms yet to be made for the transformer architecture. Indeed Section 5.1 will present the claim that features in transformer models are represented as directions, not convex regions of space.

3.3.3 Using Feature Vectors in Latent Space

One benefit of locating feature representations in neural networks is that it introduces the possibility of controlling the behaviour of models, by increasing or reducing the presence of features in the activations of a model.

The most significant examples of this line of work exist in the domain of generative image models. Radford et al. [38] demonstrated that the latent space of DCGANs is often semantically meaningful, showing that you could perform similar arithmetic to vectors as done by Mikolov et al. [33] and produce meaningful alterations to images. A similar phenomenon was investigated by Larsen et al. [39] in the context of VAE models.

These papers were extended further by White [8], who suggested performing the interpolation using spherical coordinates as opposed to just linearly. One thing to note here is that the mean of the activations can be assumed to be zero since the mean of the Gaussian prior is generally taken to be zero for these models. As noted by Cai et al. [40], this assumption breaks in the case of language models, so we would need to take into account the bias when considering latent space manipulations of language models.

White [8] also introduced a method of using feature representations to classify images, if the

feature is assumed to be represented as a direction. The algorithm, AtDot, uses the inner product between some feature vector and the activation associated with an image to be classified. This inner product can then form the basis of a classifier. In the domain of image classification, at the time this achieved performance often comparable or better than CNNs. This greatly informs the algorithm we propose for classification using language models, AtDotLLM, in Section 5.5.

The final useful point from White [8] is that labels can often be correlated, which can cause bias in the feature vector. For example, they found that smiling was often correlated with being female in the CelebA dataset, so the smiling vector also caused images to appear more feminine. In order to prevent this, the dataset was balanced. When performing analogous investigations for transformer models, unintended correlations in the data used to produce steering vectors could cause similar unintended biases, which it will be important to be aware of.

3.4 Language Model Interpretability

Although early work in interpretability largely focused on architectures such as CNNs and RNNs, there is an increasing amount of work focusing on transformer interpretability. [13] was one of the earliest works here, providing a foundation for future interpretability work by offering clear mathematical insights into transformer operations.

A challenge in this domain is identifying phenomena in smaller networks that can be generalised to larger networks. One success here is the identification of *induction heads* in language models Olsson et al. [15], which search for the previous occurrence of a token before attending to the following token. For example, when given a chapter from Harry Potter which finishes with the token "Harry", induction heads would use previous occurrences of "Harry Potter" to predict that the correct continuation is "Potter". Induction heads were originally observed in smaller toy models, and have since been found in several larger language models.

3.4.1 Feature Representations in Language Models

In Chapter 5.1, we will claim that features are represented as directions in language models. Assuming this is true, two important sub-questions might arise.

Open Question 1. *To what extent do GPT-2 style transformers represent features using the same direction across layers?*

There is some evidence to suggest that the same direction corresponds to the same feature across different layers of the transformer: nostalgebrist [41] demonstrates that applying the unembedding matrix directly to latent activations often leads to predicting the final output token before the final layer itself does.

However, other work [42] has demonstrated that improvements to [41] can be made by learning affine transformations from each layer to the output layer. This suggests that there is some degree of representation shift across layers, i.e. that feature directions are not entirely consistent across layers.

In light of this, in general it shouldn't be assumed that feature directions are perfectly consistent across different layers of the transformer. This means that activations from one layer will

not be directly compared with activations in subsequent layers, as these comparisons will not be meaningful if there is representation shift.

The partial success of [41] means that representations might remain constant enough that directly unembedding activations at intermediate layers can provide useful information. This is the implicit assumption made by the method described in Section 4.1.4, although it is important to recognise the shortcomings of this method.

The second sub-question is how many features a transformer might be capable of representing if features are represented as directions in activation space.

Hypothesis 1. *GPT-2 style transformers with a Residual Stream of dimension d_{model} could represent exponentially many features, if features are represented as directions.*

As demonstrated by Elhage et al. [34], it is unlikely that neural networks will learn to represent single interpretable features through single neurons in a network. This is because of superposition: networks can learn more features than they have neurons, at the cost of having some interference between different features.

Although directly answering how many features a network might learn to represent is an open question, it is possible to gather evidence about this by answering a related question: how many almost orthogonal vectors can be fit into an n-dimensional vector space? Almost orthogonal vectors are defined to be vectors with an angle smaller than some small angle θ.

Wyner [43] demonstrates that you can fit at least $\exp(n\log(\frac{1}{\sin\theta})+o(n))$ spherical caps of angle θ onto a sphere, such that no two spherical caps overlap. This is a sufficient (but not necessary) condition for finding a set of almost orthogonal vectors in n dimensional space, where all pairs of vectors have an angle of at least θ between them.

As described by Roger [14], one can apply this to the size of the Residual Stream of different transformer architectures to estimate how many feature vectors can fit into the Residual Stream space with a cosine similarity less than some threshold. Note also that Layer Norm makes little impact on this, since projecting to a $(n-1)$-dimensional plane has little impact for large n with respect to cosine similarity, and projecting to a sphere has no impact on cosine similarity.

If $n = 1600$, as is the case for GPT-2 XL's Residual Stream, then it is possible to fit at least 3000 vectors with pairwise cosine similarity less than 0.1 into the Residual Stream. If the pairwise cosine similarity is relaxed to only requiring smaller than 0.2, then 10^{14} vectors can be fit in the Residual Stream.

If $n = 12288$, as is the case for GPT-3's Residual Stream, then it is possible to fit at least 10^6 vectors with pairwise cosine similarity less than 0.05 into the Residual Stream. If the pairwise cosine similarity is relaxed to only requiring smaller than 0.1, then 10^{26} vectors can be fit in the Residual Stream.

This helps to provide an intuition for how many features might be able to be represented by a transformer, if there is an allowance for any two vectors to have a small cosine similarity: potentially exponentially many, compared to the size of the Residual Stream.

It also suggests that there might be a larger incentive to represent a feature using the same direction across different inputs for smaller models when compared to larger models. This is because larger models can represent many more features for the same amount of interference, so there may be less pressure to compress their representations.

3.5 Visualising Data

Since transformers often operate on vector spaces with dimensions in the range of thousands or tens of thousands, it is often necessary to be able to visualise high-dimensional data when interpreting transformer networks. This section introduces various mathematical and empirical techniques to aid in visualising and interpreting high-dimensional data.

3.5.1 t-SNE

One of the most popular approaches to producing low-dimensional plots of high-dimensional data is t-SNE [44]. t-SNE works by describing the distribution of similarities between points in their original high-dimensional space, P, and another distribution of similarities between points as described in low-dimensional space, Q. The KL divergence between these distributions is given by ($KL(P||Q)$). This divergence can be minimised using gradient descent.

It is important to understand some of the details of this algorithm to better understand its uses and limitations. Firstly, the nature of KL divergence means that a relatively large penalty is applied to plotting points far away from each other if they occur close together in the high-dimensional space. The penalty is less severe for plotting points near each other which are far away in the high-dimensional space. This may lead to points being positioned near each other even if they are relatively far away in the high-dimensional space, if this helps other points to be positioned better. This is worth noting when interpreting KL divergence.

Another important aspect is that since the distributions P and Q only incorporate information about the local similarities between points, t-SNE plots will only preserve local information about data points. Because of this, it is often not meaningful to interpret more global information about data points from the t-SNE plots.

A practical point is that, since t-SNE optimises the KL divergence using gradient descent, the low-dimensional description can get stuck in a local minimum. Fortunately, this can often be alleviated through using different initialisations alongside methods such as early exaggeration. Other practical issues arise when choosing how to measure similarities, with choices for both the high-dimensional similarity measure (perplexity, Gaussian or uniform kernels) and the low-dimensional similarity measure (how heavy-tailed to make the distance function).

t-SNE plots will be used extensively in Section 4 to perform exploratory analyses of language model activations. An alternative method could have been the UMAP algorithm, which has similar benefits and drawbacks to t-SNE McInnes et al. [45].

Another popular method for visualising high-dimensional data is Principal Component Analysis, or PCA (see Shlens [46] for a review). PCA identifies the basis vectors that account for the most variance in a set of vectors, before projecting the data points onto this basis and visualising the results.

3.5.2 AttentionViz

Some techniques have used these methods to develop methods specifically for visualising the activations of transformer models. One prominent example is AttentionViz, developed in Yeh et al. [47] to help self-attention at arbitrary layers of the transformer, for arbitrary inputs. This tool is useful for trying to understand which Attention Heads are relevant for certain

computations since it provides some insight into the target tokens being focused on by different source tokens for an Attention Head. This tool won't be used directly in the project, although it inspired the use of t-SNE plots to visualise activation space in the empirical investigations in Chapter 4.

3.6 Language Model Activations

This thesis will focus primarily on analysing the activations of language models, and analysing the similarities and differences between activations from different inputs. There is existing work which looks at similar questions, which will be useful both in providing potential avenues of exploration as well as object-level evidence about the activations of these models.

3.6.1 Contextual Word Representations

One important question when analysing the activations of a transformer on a sequence of text is how dependent these activations are on the specific context in which they are found, as opposed to being well predicted by just the token itself. Ethayarajh [48] examines the relationship between tokens and their associated Residual Stream activations, and interprets these as *contextual representations* of the token.

Some of the interpretations made by this paper do not seem applicable to language models specifically. For example, the interpretation of the Residual Stream as contextual representations of words is somewhat unfounded - although the Residual Stream can be initially seen as a static representation of the word vectors with their positions, the Residual Stream is eventually used to make predictions about the next token in the sequence, so it doesn't make sense to interpret it as providing a contextual representation of the token, especially for later layers.

One significant finding is that activations of the Residual Stream, across randomly sampled words, seemed to be highly *anisotropic*. This means that the directions are not uniformly distributed throughout the latent space of a given layer, and instead lie predominantly inside a narrow cone about the origin. This is significant because it suggests that there is a great deal of redundancy in how these models are processing information: it suggests that many directions are being underutilised, compared to what is optimal. Ethayarajh [48] measure this by computing the average cosine similarity between vectors in the Residual Stream for randomly sampled words, across all layers of the model, and find that for GPT-2 the average cosine similarity is non-zero.

3.6.2 Evolution of Activations through Layers

Voita et al. [49] formalises the shortcomings of Ethayarajh [48] well, emphasising that the Residual Stream needs to both preserve information needed to build representations for other tokens, whilst also predicting the output label. This is different to other model architectures such as Machine Translation and Masked Language Modelling, and is important to consider when analysing the Residual Stream and activations for language models such as GPT-2.

This is demonstrated empirically by estimating the mutual information between the initial source token, and the vector in the Residual Stream corresponding to that token at other layers

in the model. For GPT-1 this decreased throughout deeper layers, which is consistent with the hypothesis that the model gradually forgets the input token throughout the forward pass. A symmetric analysis was carried out for the output token and found that the mutual information monotonically increased.

Voita et al. [49] also creates t-SNE plots for the Residual Stream activations of different tokens across layers in a model. By producing these plots for each layer of the original GPT-1 model, one can see that the representations start by forming discrete clusters in early layers based on input tokens, which merge in later layers.

If Voita et al. [49] demonstrate that there are clusters in early layers based on the input token, then many of the empirical investigations in Chapter 4 can be seen as trying to understand the conditions under which clusters will form in later layers. The results provided by [49] serve as a useful proof of concept for this line of investigation.

3.6.3 Describing GPT-2's Activations

After accounting for the misinterpretation of Residual Stream activations, there are two major open questions posed by Ethayarajh [48]: why does anisotropy arise; and can a fuller geometric description of the activations be provided besides simply noticing the anisotropy? Gao et al. [50] provide one potential explanation for the anisotropy, suggesting that it can arise in embeddings for transformer models due to a combination of layer-normalisation and weight tying, alongside the fact that when training on language tasks there will inevitably be tokens with a low frequency relative to the entire training corpus. This suggests that the phenomenon of anisotropy isn't just a quirk of some of the smaller models which have been investigated, but could also arise in larger decoder-only language models. That being said, there are some limitations of the explanation provided: it makes some assumptions about the frequency of tokens in the corpus used to train models which might not be realistic; and it only applies to initial embeddings, not explaining why anisotropy can persist and even increase across layers. These questions will be investigated further in Section 4.2.

Cai et al. [40] provide a further description of anisotropy and carry out more investigations of the Residual Stream of GPT-2 across a large dataset of inputs. They first measure how well the space can be described using PCA, and look at the singular value distribution of the Residual Stream to do this. Separately, the authors perform K-means clustering alongside using a measure of clustering quality known as Maximum-Mean-Silhouette to describe how the activation clusters are distributed in the space. Both the PCA and K-means results suggest that the activations of GPT-2 small are well described by two clusters, complicating the picture of a single cone.

This also helps to further explain the seemingly anomalous result from Ethayarajh [48]: if you shift the mean from each cluster to the origin, and then look at the cosine similarities between tokens in the same cluster, the anisotropy disappears. This mirrors the word representation post-processing technique introduced by Mu et al. [51], and the resulting clusters demonstrate approximately 0 average cosine similarity between word representations for most layers in GPT-2 small. This means that the Residual Stream is almost perfectly isotropic within each cluster.

Cai et al. [40] make further investigations into the nature of these activation clusters. Some important results with respect to GPT-2 small were:

- Several layers seemed to arrange points into a "swiss roll" structure after PCA is performed

on the activations. Tokens with smaller positional encodings mapped closer to the centre of the roll.

- The frequency of the token influences its embedding position.
- The two clusters did not seem to correspond to different sets of tokens: a token in the smaller cluster would often appear in the larger cluster, dependent on the context.
- The approximate local intrinsic dimension is 7. This increases throughout the layers of GPT-2 and is significantly smaller than that given by static word embeddings.

This paper makes a good deal of progress towards some of the goals of this thesis. However, several directions remain unexplored here.

- Although different models are explored, the authors do not test whether the results found in GPT-2 small replicate to other decoder-only transformers. This makes it hard to know whether these are properties of the specific model they tested, or whether they are preserved as scale increases.
- Cai et al. [40] focus on similarities between activations for the same input tokens, across different contexts. They do not focus on activations between different tasks which produce similar outputs.
- The Residual Stream is the only space examined: they do not consider the outputs of the Attention Heads, or MLP Layers, directly.

This paper significantly informs this thesis. Many of the investigations in Section 4 seek to address the directions for future work suggested by this paper, and the thorough account of anisotropy directly informs the development of mean-centred activation steering in Section 5.2.2.

3.7 Manipulating LLM Activations

A major aim of this thesis is to better understand the structure of language model activations, to better control the behaviour of language models through intervening on the activations of language models. There have been some recent developments in this area, which can be seen as extending the analogous methods for image models from Section 3.3.3.

3.7.1 Activation Steering

One method for changing the behaviour of LLMs by intervening on their activations is developed by Turner et al. [52]. Their method works by modifying activations at inference time, by adding a *steering vector* to the Residual Stream activations at some layer. These steering vectors are generated by storing the activations associated with an arbitrary prompt, at some layer of the model.

This method is promising, since it has much lower computational overheads than other methods such as RLHF and finetuning. It also potentially offers a more principled method of control

than prompt engineering. However, there are a number of free parameters which need to be optimised for the method to work correctly, and there is currently no principled method of finding these outside of grid search. The method of using single text prompts to generate steering vectors is also limited.

3.7.2 Detecting and Improving Truthfulness

Some initial explorations in applying these ideas to language models have indeed been fruitful, although they have been somewhat limited in scope. One prominent example of this kind of work is Burns et al. [53], where the authors train a linear network to accurately answer yes-no questions, using only the activations of a language model. Surprisingly, this method often outperformed the zero-shot accuracy of the language models themselves.

There are several interesting ideas to take from this. Firstly, the paper demonstrates that tractable progress can be made in improving our understanding of language models using existing techniques. There is also the object-level result itself: since the model they trained on the activations of the language model was a simple linear probe, it suggests that something corresponding to a truth feature is encoded as a direction in the final layer of the model, which provides some evidence that features in language models occurring in multiple different contexts are represented as single directions. This suggests that feature detection is indeed tractable.

The obvious extension of this work is to find directions which correspond to truthful outputs to improve the truthfulness of language models. Li et al. [54] do this by adding outputs to certain Attention Heads which correspond the most to truth during inference, and demonstrate that this can improve truthfulness without degrading performance too drastically.

3.8 Summary

Additional work is required to be able to control LLMs (Section 3.2). There has been some progress in controlling other kinds of neural networks by exploiting how they represent features of inputs (Section 3.3). There has been some initial progress in understanding the structure of language model activations (Section 3.6), before using this knowledge to better control language models (Section 3.7).

The rest of this thesis will seek to make further contributions to these final two areas. Chapter 4 will provide an empirical investigation into the structure of language model activations, focusing specifically on when and how they represent features of text. Chapter 5 will begin with a further theoretical investigation of the structure of language model activations, before using this improved understanding of language model activations to develop new methods for controlling language models.

Chapter 4

Empirical Investigation

The purpose of this chapter is to empirically investigate the structure of the activations of language models. In particular, we will investigate how and when features of text are represented in the activations of transformer models. Some sub-questions to help answer this will include:

- For which layers are activations of language models zero-centred, if any?
- Do transformer models represent features of text via activations?
- If a transformer represents a feature, in what layer is it first represented?
- Which features predict similarities between activations the best? Does this change across layers?
- Do these findings differ across models?

The general method used to investigate this will be to create datasets of strings S such that the strings in S share some feature. The activations associated with these strings for some model can then be analysed, to better understand which features of text are salient for changing how models process different inputs.

For the models studied here (GPT-2 small, medium, large and XL), the activations lie in high-dimensional vector spaces (ranging from 768 to 1600), meaning that it is not trivial to compare the activations. To aid with this, we present several different methods which can be used to infer information about the activation space in Section 4.1.

We then use these empirical methods to look at how some specific features are represented in these models. We will then summarise some broader inferences that can be drawn about how these models represent certain features.

4.1 Methods

This section will discuss four experiments for empirically analysing the activations of transformer models.

4.1.1 Singular Value Distribution of Datasets

This method was designed to help determine which features of text could best predict similarities between activations. The goal was to develop a measure to gauge the variety of activations for a given dataset at a specific layer: large variety in activations means that the feature is a bad predictor for the activations at the layer; lower variety in activations means that the feature is a good predictor.

The general setup for the experiment was as follows:

1. Create a dataset S of size N, where every element of the dataset is composed of k tokens.
2. Feed each of the strings in S into a GPT-2 model and store the activations X_i at some layer i of the network for either the outputs of the MLP Layer, the Attention Layer, or the Residual Stream.
3. Normalise X_i, and look at the matrix resulting from $Y_i = X_i^T X_i$.
4. Calculate the singular value decomposition of Y_i. Plot the distribution of the singular values of this matrix.

Repeating this for different datasets allows us to compare the singular values of the activation matrix for different datasets. The dataset size N and number of tokens k were held constant for each dataset, to make the comparison as meaningful as possible.

This can be interpreted as follows: a more uniform distribution of singular values means there is more variety in the activations, since it takes a larger subspace to accurately approximate the activations. Conversely, a less uniform distribution (with singular values decreasing more quickly) means that a smaller subspace can accurately approximate the activations, suggesting that the feature is a better predictor of the activations.

4.1.2 PCA Reconstruction Errors

The experiments in 4.1.1 provide evidence that different datasets inputted to GPT-2 can lead to variations in the number of dimensions that their corresponding activations can be described with. However, singular value decomposition also generates singular vectors, which can be used to better understand which subspaces the activations typically lie in. By comparing the singular vectors of the activations for two different datasets, it is possible to measure how similar different datasets are.

One method we may wish to do this through is by comparing how well the top k principal vectors from one dataset can be used to approximate the activations from a different dataset.

1. Create a target dataset S of size N_1.
2. Feed each of the strings in S into a GPT-2 model and store the activations X_i at some layer i of the network for either the outputs of the MLP Layer, the Attention Layer, or the Residual Stream.
3. Calculate the singular value decomposition of Y. Take the top k singular vectors, which correspond to the k largest singular values.

4. Create a comparison dataset S', of size N_2.
5. Feed each of the strings in S' into a GPT-2 model, and store the activations X'_i at the layer i of the network for either the outputs of the MLP Layer, the Attention Layer, or the Residual Stream.
6. Project the rows of X'_i onto the subspace generated by the top k singular vectors from X_i.
7. Look at the L_2 norm of the errors. Calculate this as a fraction of the L_2 norm of the original row vectors in X'_i.
8. Take this fraction as the reconstruction accuracy.

We can repeat this process for multiple target datasets, as well as for multiple comparison datasets. The reconstruction accuracy can then be interpreted as a measure of how similar two different datasets are: if the reconstruction accuracy for a dataset Z_1 on a target dataset X is higher than the reconstruction accuracy on a second dataset Z_2, then we might interpret this as Z_1 being processed more similarly to X than Z_2, at this location in the model.

Note that there are various parameters in the above that we could vary. We can consider what would be learned about the models if these parameters were to be varied.

Vary Number of Principal Components

One parameter that could be varied is the number of principal components, denoted by k, that activations are projected onto.

This can be particularly useful if the activations of each dataset are projected onto its own top k principal components. This provides a metric for how well different datasets can be approximated via low-rank approximations.

Vary Layers

We might also keep the number of principal components k fixed, and instead vary i, the layer we look at the activations of. This will allow us to see whether different datasets are processed similarly throughout the layers of a model, or if instead there are differences based on the location. This might help us to characterise which layers of models are most relevant for representing different features of text.

4.1.3 t-SNE Plots

Another approach to differentiate between datasets is to generate low-level descriptions of their activations and observe their distributions.

We utilised t-SNE to produce these low-dimensional descriptions. This has some disadvantages, as noted in Section 3.5.1, but it should be able to determine whether datasets can be separated based on local distances between points.

1. Create multiple datasets $X_1, X_2, ..., X_n$ of size N_1.
2. Feed each of the strings in each of the datasets into a GPT-2 model, and store the activations from either the outputs of Attention Layers, MLP Layers, or the Residual Stream for some layer i of the transformer.
3. Create a 2-D t-SNE plot from these activations.

The existence of clusters in these t-SNE plots will suggest that there are similarities between the activations for the different inputs in the studied datasets. This can help us to determine which features of text are being represented by the transformer, as well as which layers they are first represented in.

4.1.4 Comparing Feature Vectors to Embedding Vectors

To better understand potential feature vectors, it is useful to find ways to interpret them. One method of doing this, as demonstrated by Beren Millidge [55], is to compare the inner product of the vector with each token embedding. We can then find the top k tokens which generate the highest dot product, and the bottom k tokens which generate the smallest dot product.

The intuition behind this is that it should provide information about what the chosen vector represents: if it is most similar to words associated with fantasy stories, for example, then we might assume that the vector represents a feature associated with fantasy stories.

It should be noted that this is not a rigorous comparison, since we are making the implicit assumption that feature representations are consistent across layers of the transformer (Section 3.4.1). Despite these caveats, empirically this represents a useful method for better understanding and interpreting vectors in the Residual Stream.

In Table 4.1 we show one example of performing the above analysis.

Table 4.1: Tokens with Highest and Lowest Cosine Similarity to the vector from the mean of activations from a subset of the training dataset, to the mean of activations of the fantasy dataset. Using GPT-2 XL, analysing the Residual Stream after layer 30.

Tokens with Highest Cosine Similarity	Tokens with Lowest Cosine Similarity
enchanted	itto
mystical	endors
magical	pez
awakened	referen
sorce	zzi
enchant	Anyway
mystic	NYT
arcane	crap
awakening	Tank
destiny	BUS
enchantment	pson
magic	Reps
awaken	Jerry
treasures	Asked
secrets	Corrections

4.2 Understanding Bias in Activations

The following is the first experiment conducted on the activations of GPT-2 models. The purpose of this experiment is to determine the extent to which there is bias and clustering in the activations of GPT-2 models, helping to understand whether activations are centred around the origin or not. This is important to determine when developing approaches for activation additions, which we will discuss in Section 5.4. We will follow a similar analysis to that carried out by [40].

4.2.1 Does a bias exist?

One method of seeing the extent to which there is clustering in activation space is to look at the average cosine similarity between different vectors in the Residual Stream. Vectors which are uniformly distributed around the origin should produce an average cosine similarity of 0; all vectors lying along a single direction would give an average cosine similarity of 1.

Here, we take the average cosine similarity between all pairs of activation vectors from either the Residual Stream, output of Attention Layers, and output of MLP Layers separately.

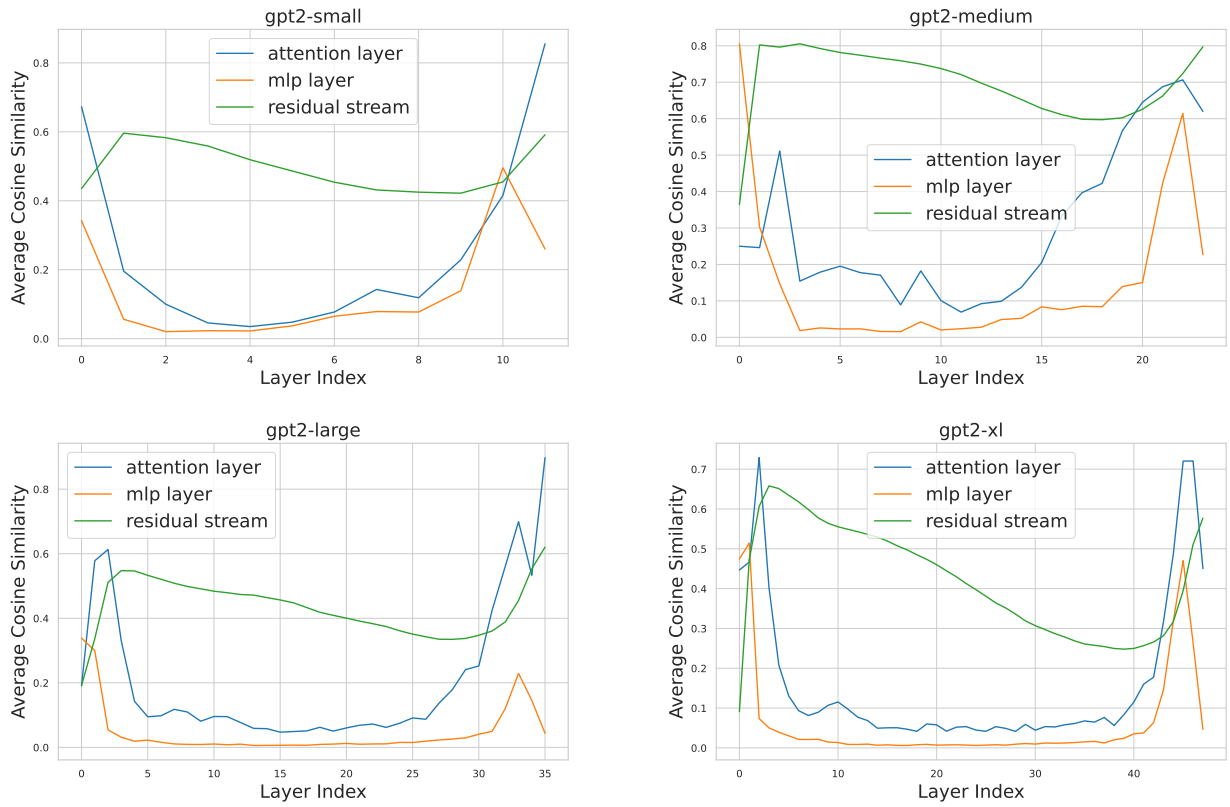

Figure 4.1: Average cosine similarity across pairs of activations in either the Residual Stream, the output of an Attention Layer, or the output of an MLP layer. Results are for GPT-2 small, medium, large and XL respectively. Activations were generated from a subset of training data [56].

Figure 4.1 demonstrates that for the GPT-2 models (small, medium, large, XL), there exists a bias in the Residual Stream across all layers. It is interesting to note that in GPT-2 large and XL

there is initially little bias, before it grows over the first few layers. All models also exhibit an increase in bias in the final few layers.

All models also exhibit non-zero bias in the output of the Attention Layer across all layers, although this is lower than the Residual Stream bias. The MLP Layers seem to exhibit some bias in the penultimate layer of each model.

4.2.2 Implications for Investigations

The existence of bias in the Residual Stream activations has several implications for interpreting results in the ensuing investigations of GPT-2 models. First, note that PCA reconstruction accuracies may be expected to be higher than they would be without bias in activations, since bias in the activation space implies larger than 0 average cosine similarity between PCA directions.

Also, note that since there are significant changes in the average magnitude of bias in the activation space across layers, we may expect to see trends in the PCA reconstruction accuracies across layers. Hence when analysing trends in PCA reconstruction accuracises across layers, the trends in bias size should be taken into account.

4.3 Model Output

The rest of this Chapter will consist of investigations into the structure of activations for specific datasets, using the methods in Section 4.1. Each of these datasets corresponds to different features of text, so these investigations can be interpreted as investigations into how GPT-2 models represent different features.

The first investigation aims to see whether t-SNE plots on the activations can produce clusters with respect to the expected model output. This is motivated by the monotonic increase in mutual information between activations and model output observed in Voita et al. [49].

This can be tested by choosing a simple task which the model can perform, determined by evaluating it on different inputs. One such task is simply repeating a string, corresponding to the prompt shown in Figure 4.2.

> Repeat the string '{x} {y} f'. {x} {y}

Figure 4.2: Template for the text in the repetition dataset. Text in {} are variables, which can stand for any letter a-z. All the GPT-2 models correctly outputted the token ' f' for these inputs.

Figure 4.3 demonstrates how clustering in the final position of the activations changes throughout the layers of the transformer (in this case GPT-2 large). Initially we see 26 clusters, which corresponds to the 26 different ways the input prompt can end. Then we see these clusters begin to separate based on the output label, before the outputs are separated into global clusters based on the output. Interestingly each global cluster is still composed of 26 smaller clusters, which correspond to the final tokens of the input prompts (i.e. the variable y in the template).

This provides strong evidence that low-level features of input data can be lead to geometric similarities in activations of GPT-2 style models. This suggests that trying to analyse features geometrically could be promising across more complex and interesting datasets.

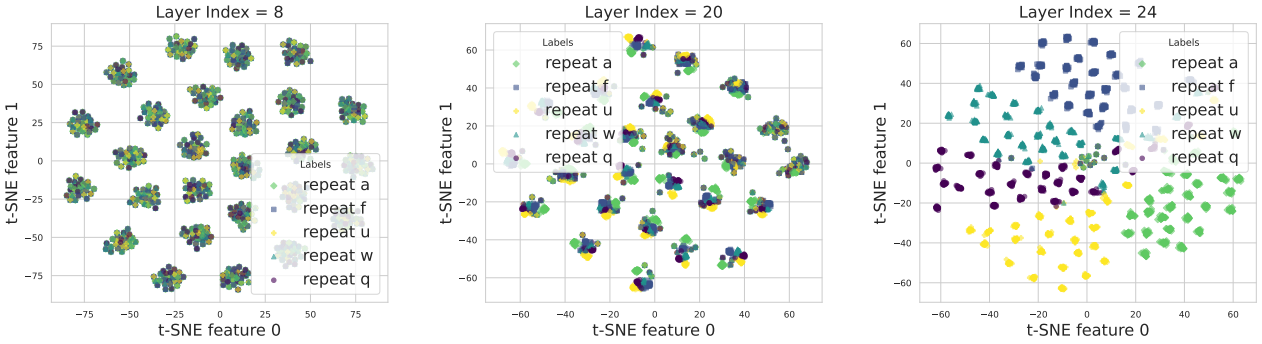

Figure 4.3: t-SNE plots for Residual Stream activations across various layers in GPT-2 large, for six different repetition datasets.

4.4 Numerical Datasets

Following the clear demonstration of clustering based on outputs in Section 4.3, a natural extension is to conduct a wider investigation on other tightly controlled datasets. We investigate different numerical datasets, to better understand the similarities and differences between how their activations are structured.

It is important to note that neither GPT-2 small nor GPT-2 XL perform well on any of these basic arithmetic datasets. This means that we shouldn't expect to see clustering emerge based on the outputs of these models, but the models' treatment of different datasets can still be analysed. This might be useful in better understanding how models perform even when they are failing on certain tasks.

4.4.1 Datasets

Seven different datasets were compared for this experiment.

1. *Addition dataset*. This dataset consisted of all strings of the form "x + y =", where x, y were integers between 0 and 24. These were then tokenised using the GPT-2 tokeniser, giving a dataset of 5 tokens (also including the EOS token).

2. *Multiplication dataset*. This dataset was identical to the addition dataset, except that "+" was replaced by "*".

3. *Shuffled Addition dataset*. This dataset was created by randomly shuffling the characters in the addition dataset.

4. *Random Addition dataset*. This dataset was similar to the Addition dataset, but instead of choosing x and y as integers between 0 and 24, 24 tokens were randomly chosen from GPT-2's tokeniser. The dataset then consisted of all strings of additions between these.

5. *Addition and Multiplication dataset*. This dataset was created by choosing inputs from both the Addition and Multiplication datasets.

6. *Training dataset*. Although OpenAI did not open source the dataset used to train GPT-2, they did publicise how the dataset was collected. This has since been used to create an open-source replication [56]. This was used to create a dataset by taking a subset of sentences from the training dataset, and then taking the first 5 tokens from these sentences.

7. *Random token dataset*. This dataset was created by randomly sampling from tokens in GPT-2's tokeniser, and creating 5 token long inputs using this.

One important point to note about the above datasets is that the precise makeup of the datasets is important here if we want to make meaningful comparisons between datasets. The main reason behind this is the sensitivity of the tokeniser: it will tokenise "x + y =" differently to "x + y = ", for example, due to the trailing whitespace after "=" in the second string. This can also lead to inadvertently creating sequences which require different numbers of tokens to describe, which would make comparisons between datasets less meaningful.

4.4.2 Results

The singular value distributions (Section 4.1.1) for the activations of GPT-2 Small (Section 4.1.1) on the datasets described above, can be found in Figure 4.4.

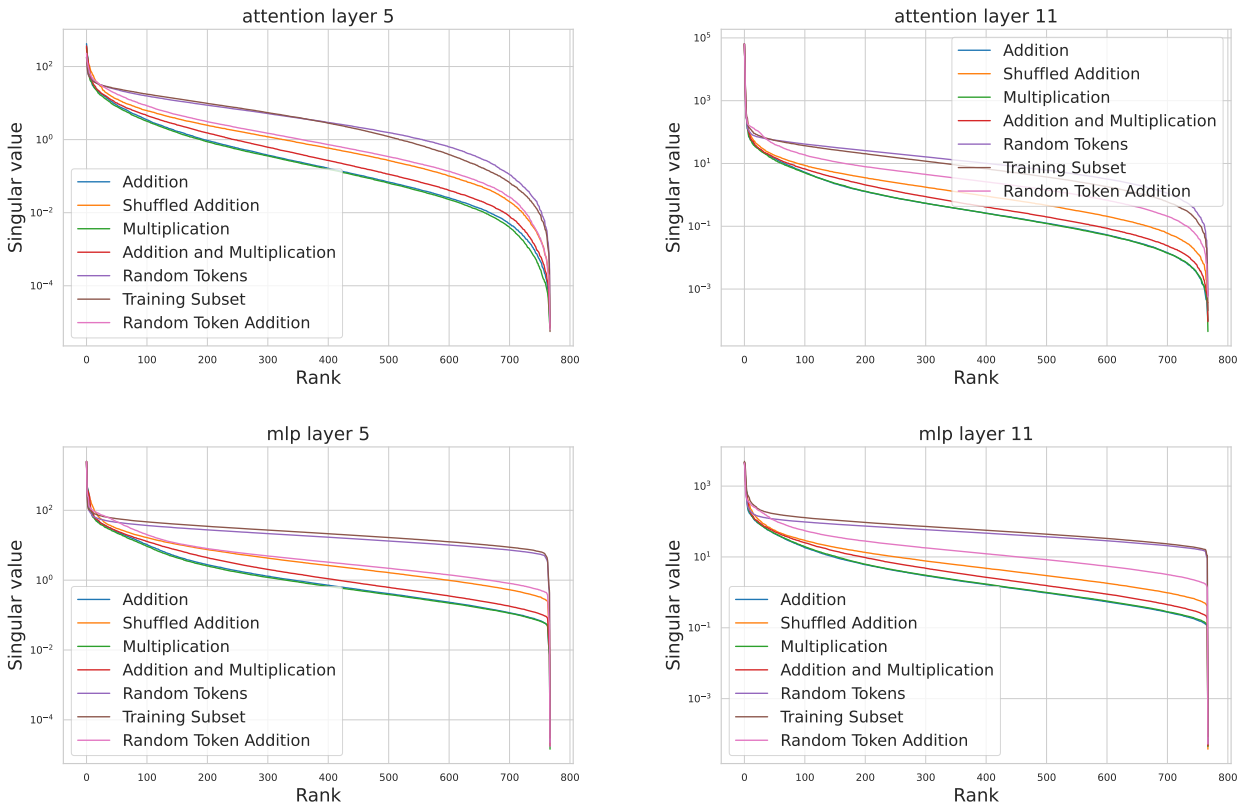

Figure 4.4: Singular value distributions for the activations corresponding to various datasets in GPT-2 Small.

The first point to notice is that there are meaningful differences in the singular value distributions of these datasets, which suggests that different features of datasets can produce differences in the structure of activations.

Furthermore, the datasets which one might expect to be the "simplest", i.e. the Addition and Multiplication datasets, have the least uniform associated singular values. Note that this is not explained simply by the syntactic regularity of the datasets, which can be seen by comparing these to the larger singular values associated with the Random Token Addition dataset. This suggests that the model has learned to reuse similar activations across different integer addition prompts to a greater than it has for random strings of the form "x + y =". This might indicate that the model is representing a feature corresponding to integer addition in the same way across multiple different inputs.

The structure of the activations associated with the numeric datasets can be investigated further by producing t-SNE plots (Section 4.1.3). The results are provided in Figure 4.5.

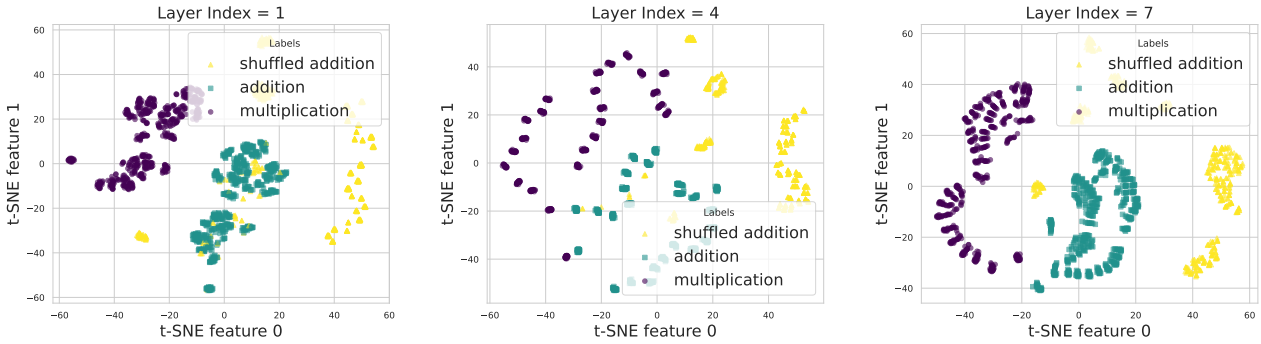

Figure 4.5: t-SNE plots of Residual Stream activations of GPT-2 small for Addition, Multiplication, and Shuffled Addition datasets.

Looking at the Residual Stream across layers of GPT-2 small, we see that there is clear clustering for the Addition and Multiplication datasets before layers 1 and 7. Note that some points in the Shuffled Addition dataset are originally plotted between Addition dataset activations in layers 1 and 4, before being moved out after layer 7. This might suggest that in early layers the model doesn't distinguish between the strings "2 + 2 =" and "+ 2 2 =", but that this distinction is then made in the later layers.

Additionally, there is less clear clustering for the Shuffled Addition dataset in general, and one might hypothesise that the distinct clusters in layer 7 are based on the order the "+" and "=" tokens arise in.

4.4.3 Analysis

Overall, this investigation provided evidence that singular value distribution plots could help determine the variety of a model's behaviour on a dataset. Different datasets produced different singular value distributions, and this seemed to correspond loosely to the diversity of the tasks in the dataset. The addition and multiplication of integers seem equally complex, and they seem to have similar singular value distributions. Likewise, the most complex datasets (the training dataset and the random token dataset) showed the most uniformity in their corresponding singular values.

However, looking at the singular value distribution was insufficient to distinguish between all of our different datasets. For example, addition and multiplication produced almost identical plots, and it was not obvious how to interpret the results of the random token dataset as opposed to the training dataset. Figure 4.5 provided evidence that t-SNE plots could provide a useful tool for doing this, since they were able to demonstrate separation between the addition and multiplication datasets.

This suggests the following distinction in uses between methods: singular value distribution plots demonstrate the complexity of datasets; t-SNE plots demonstrate which individual data points are being treated similarly or differently to one another.

4.5 Language

Although being able to distinguish between the activations for the highly structured numeric datasets suggested that this might be possible for a wider range of datasets, one limitation of these datasets was that they were highly syntactically structured. To address this limitation, datasets written in different languages are considered (specifically French, German, English, Spanish, and Portuguese).

Figure 4.6 demonstrates the results of making t-SNE plots from these datasets. They demonstrate clear separation between different languages, with some small portions of overlap. This suggests that GPT-2 small recognises the language of the inputs, and represents this in its activations. One exception to this is that after the final transformer block, there is little separation between Spanish, Portuguese and French. This could be explained by the similarities in the tokens used by these languages.

One interesting thing to note here is that the separation isn't "clustering" in a traditional sense, but instead the clusters often look more one-dimensional. It is unclear why this is the case.

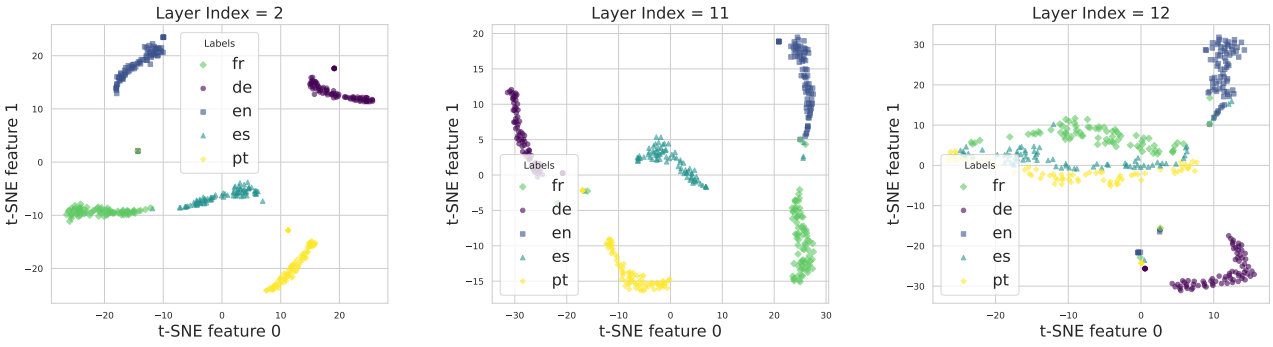

Figure 4.6: t-SNE plots of Residual Stream activations of GPT-2 small, for French (fr), German (de), English (en), Spanish (es), and Portuguese (pt) sentences.

4.6 Reading Comprehension

Given this initial evidence that some features are represented by GPT-2 models in the activation space, it would be useful to understand which features affect the outputs of a model, and

which features are most relevant in certain layers of a model's computation. Model behaviour on reading comprehension datasets was studied to aid with this since GPT-2 XL can perform well on basic reading comprehension, and the questions can be carefully controlled to pinpoint exactly which changes have the biggest impact on the output.

4.6.1 Datasets

Each story in the reading comprehension datasets followed this template:

> In the quaint village of Meadowbrook, the Thompson family was renowned for their artistry. {patriarch}, the patriarch, was an accomplished painter. His wife, {matriarch} Thompson, was a talented sculptor. They had two children, {child1} and {child2}. {matriarch}'s brother, {brother}, often visited them and was known for his comical anecdotes.
>
> The name of {matriarch}'s {relation} is

Figure 4.7: Template for the text in the datasets. Names in brackets are variables, which can stand for any one of ten possible names.

There are several variables in this template, which correspond to the question being asked (the relation), and the names of different characters.

Datasets were produced by specifying some aspect of the template to be kept fixed across the dataset, with all non-fixed aspects being randomly sampled to generate each story in the dataset. Note also that **only the activations associated with the final token are analysed for this Section**, since they correspond to the activations used by GPT-2 XL to make its final prediction.

Three datasets were produced with this method:

- 'fixed name' where the correct answer to the final question was kept constant, Taylor;
- 'fixed relation' where the question about the relation was kept constant, brother;
- and 'fixed name and relation' where both the correct answer and relation were kept constant, i.e. Taylor and brother.

Note that these definitions give some possible overlap between the 'fixed name' and 'fixed name and relation' datasets, so the 'fixed name' dataset was changed to remove all instances where the relation was brother.

The purpose of this setup was that it seemed likely that both the relation and answer would be important for the model in answering the question. This setup allowed us to see the layers of the model where these features were being represented and used by the models.

4.6.2 Results

T-SNE plots were created for the 'fixed name', 'fixed relation' and 'fixed name and relation' datasets, at various layers of the Residual Stream of GPT-2 XL. The results are shown in Figure 4.8.

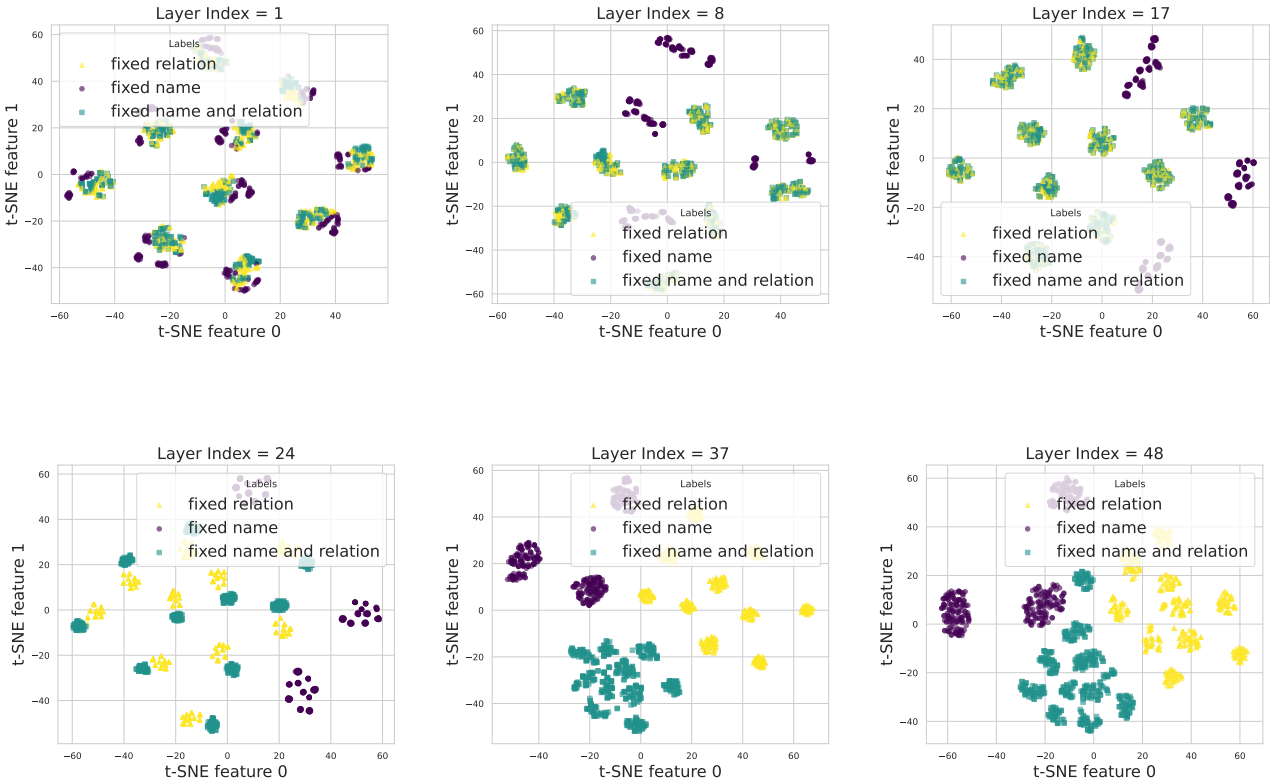

Figure 4.8: t-SNE plots for various layers in the Residual Stream of GPT-2 XL, for the three reading comprehension datasets.

After the first layer clusters have emerged, but not based on characteristics which we fixed. As we look through deeper layers of the model, three clusters begin to emerge for the fixed name dataset. We then see the 'fixed name' and the 'fixed name and relation' datasets splitting, before they eventually form separate clusters. However, it is not the case that each dataset corresponds to a single cluster in the final layers, as they determine large clusters which are composed of smaller sub-clusters.

By inspecting the data points in different clusters, we can hypothesise what characterises them.

Layer 0 Clusters

For the first layer, we find that the **clusters correspond to the name of the matriarch in the story**. This might initially seem surprising, but upon further inspection this is the variable that appears the most in the stories. Hence, perhaps we can hypothesise that this early layer corresponds to just moving information from all tokens to the final token somewhat uniformly, and thus we should expect the largest differences to be due to the names which vary the most.

Layer 47 Fixed Relation Clusters

Inspection reveals that these clusters are determined by the correct output name. This is consistent with what we would expect, since these names are output by GPT-2 XL when we generate completions with temperature 0.

Layer 47 Fixed Name Clusters

Inspection reveals that these clusters are determined by the relation variable in the prompt. It makes sense that this clustering would occur in intermediate layers of the model, but, interestingly, it remains even in the final layer (just before the Unembedding Layer is applied). This suggests that the model is not removing instrumentally, but not terminally, useful information from the final token, even though it is ultimately unimportant for the final answer.

Layer 47 Fixed Name and Relation Clusters

Given that both name and relation are fixed here, it is not obvious why clusters should emerge here.

Analysis of these clusters suggests that they are based on the name of the cousin in the story. This is surprising because, for this dataset, the final question is only ever being asked about the name of the brother.

Interestingly, when we put these questions into the GPT-3 playground we see that the weakest models sometimes fail by answering with the name of the cousin instead of the brother. For the larger GPT-3 models, the second largest logprobs are again ascribed to the cousin. This seems to suggest a possible failure mode of these models, which was interesting to find through this kind of analysis.

PCA Reconstruction Results

Following this experiment, the PCA reconstruction error experiment (Section 4.1.2) was performed, using the 'fixed name and relation' dataset as the target dataset. The results of this can be seen in 4.9.

Given the bias inherent in the Residual Stream of GPT-2 XL (Section 4.2.2), the general downward trend of these curves down and then up for the final few layers is not surprising. However, the relative position of the 'fixed name' and 'fixed relation' curves across layers is meaningful. Initially the 'fixed relation' curve is higher for all three plots, before it eventually drops below the 'fixed name' curve.

This suggests that in early layers the relation is more important in determining the acti-

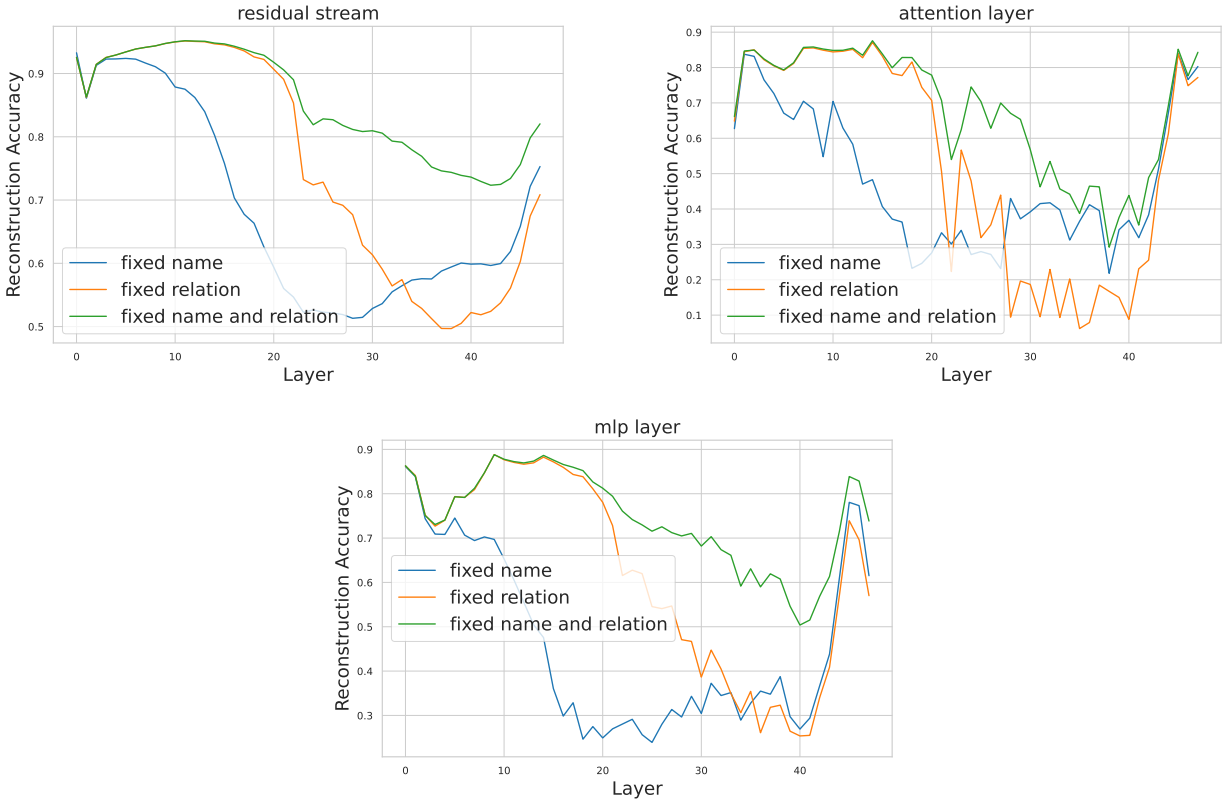

Figure 4.9: PCA reconstruction errors across layers of GPT-2 XL for the Residual Stream, Attention Layer, and MLP Layer.

vations, before the final name to be output becomes more important. This is similar to the findings in the t-SNE plots (Figure 4.8) that the activations associated with the 'fixed name' dataset formed distinct clusters in earlier layers, before the 'fixed relation' data points also formed clusters distinct from the 'fixed name and relation' data points.

4.6.3 Analysis

One broad takeaway from this investigation is that clusters in activation space can often be semantically meaningful, since it was possible to describe each cluster accurately. However, although some aspects of clustering could be predicted in advance (such as some clustering based on the output), this investigation demonstrated that sometimes unpredictable clusters may emerge (such as clusters based on the name of the matriarch and the cousin). This could be used in different contexts to help predict whether models will fail in other contexts.

Another broad takeaway is that PCA reconstruction errors can be used to separate different datasets, to make better comparisons between how models process datasets differently throughout different layers. The consistency with the t-SNE plots indicated that they are both measuring similar phenomena, which suggests it is reasonable to use the PCA reconstruction errors as a more formal measurement of similarity than the t-SNE plots.

Finally, notice that the PCA reconstruction error plots for the Residual Stream, Attention Layer output and MLP Layer output all demonstrate similar trends in general, specifically in the relative error of the 'fixed relation' dataset compared to the 'fixed name' dataset. This suggests that both the MLP and Attention Layers differentiate between these features in this instance.

4.7 Story Genres

Another extension of the finding that there are differences in the structure of the activations of GPT-2 models on different data points (Sections 4.4, 4.5) is to consider other datasets which are relatively unstructured. To investigate this, datasets of stories of different genres are constructed. Genre was chosen because it seems less obvious that a model would represent it than other features such as language, yet given the relative proficiency of GPT-2 models at writing stories one might expect the models to represent the genre in activation space.

Stories were generated using GPT-3.5 through the OpenAI API, generating 200 stories using a prompt such as "Write a fantasy story. It should have 8 lines.". To ensure there was variation in these stories, a temperature of 1 was used. This was chosen empirically, as it demonstrated sufficient variation whilst still producing coherent stories. Fantasy, sci-fi, and sports stories were generated.

Note that because all of these stories were generated in the same way, one should expect there to be other similarities or confounding variables besides just the genre and length of the stories. It is worth noting this as a limitation of these datasets.

Figure 4.10 gives the results of creating t-SNE plots for the final activations in the Residual Stream of GPT-2 small. The emergence of clusters based on story genre demonstrates that after some intermediate layer GPT-2 small represents genre in activation space.

Given the clear clustering observed here, these datasets could provide useful for testing different methods of extracting feature vectors. We will return to this in Section 5.3.

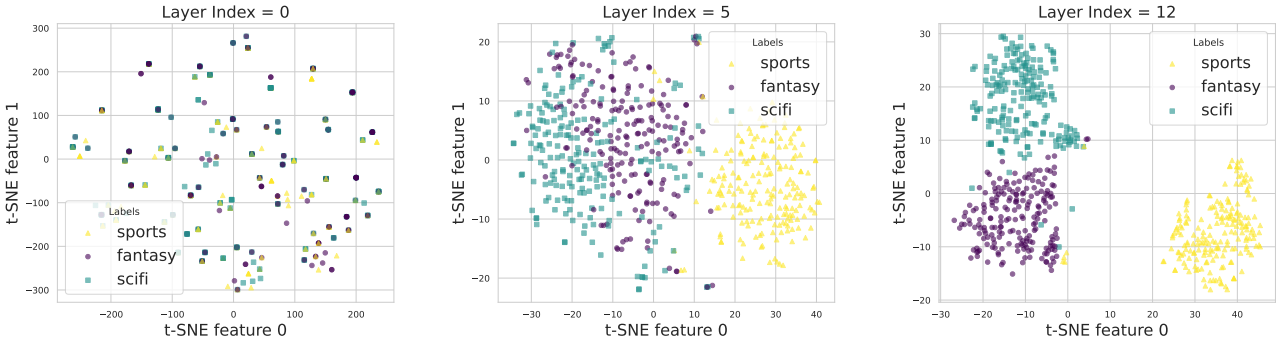

Figure 4.10: t-SNE plots of Residual Stream activations of GPT-2 small, for sports, fantasy, and scifi stories generated by GPT-3.5

4.8 Genres vs. Tasks

The analysis in Section 4.7 can be extended by adding different questions to the stories to quantify the relative impact of different features on the activations of the GPT-2 models. To do this, we will conduct analyses similar to Section 4.6.

4.8.1 Datasets

The datasets used to investigate this were created by using the fantasy, sci-fi and sports stories from Section 4.7, and adding different questions to each of them. The rationale behind this was to understand better how the task given to the language model changed its behaviour, and how this differed from just changing the genres of the stories.

Q: What is the genre of this story? Answer in one word A:
Q: How likely is it that this story could occur? Answer in one word A:
Q: Continue this story. A:

Figure 4.11: Different questions added to the stories generated in Section 4.7. These will be referred to as the genre question, likelihood question, and continuation question respectively.

4.8.2 Results

Figure 4.12 demonstrates t-SNE plots when varying both the genre of the story, as well as the question asked after the story. The clear separation in clusters after the first layer suggests that GPT-2 XL represents both the genre and the question in activation space after the first transformer block.

To further understand the differences in how the models treat these two differences (the difference in genre of story vs. the difference in question asked), the PCA reconstruction experiment (section 4.1.2) can be performed, using the fantasy and continuation dataset as the baseline. The results of this are demonstrated in Figure 4.13.

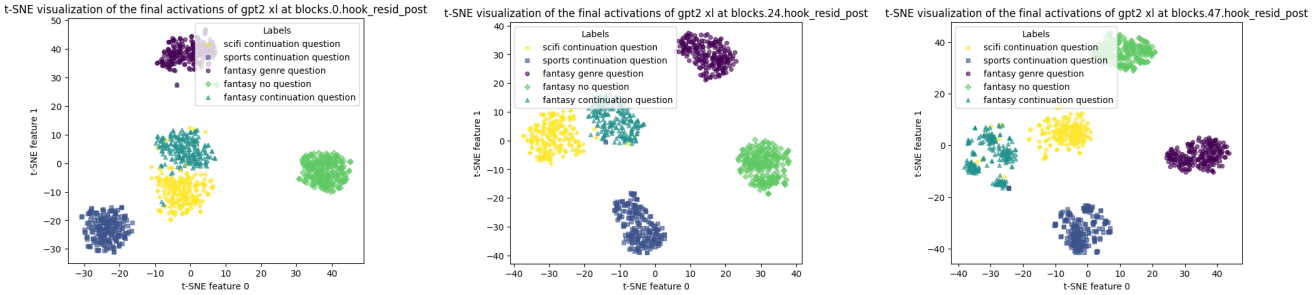

Figure 4.12: t-SNE plots of Residual Stream activations of GPT-2 XL, for stories of different genres followed by different questions.

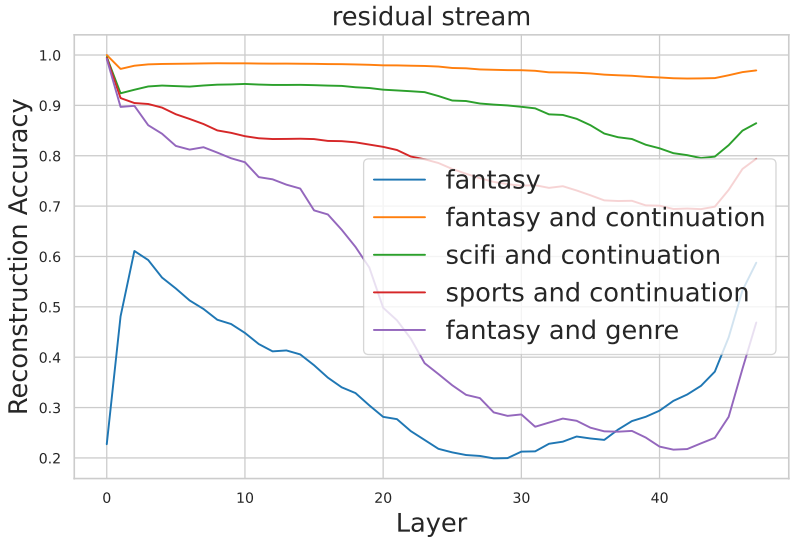

Figure 4.13: PCA reconstruction accuracies for GPT-2 XL, using fantasy stories followed by continuation questions as the comparison to project onto.

4.8.3 Analysis

The higher reconstruction accuracies of sci-fi and sports continuation questions when compared to fantasy stories with different questions indicates that the question is more important when determining the computation performed by the model, when compared to the genre of the stories.

Furthermore, it is interesting that (by the PCA reconstruction metric) the fantasy and genre dataset is initially more similar to the fantasy and continuation dataset than the fantasy dataset without any question, but is less similar by about layer 35 onwards.

One explanation for the initial similarity might be that there are superficial similarities between the fantasy and genre question dataset and the fantasy and continuation dataset (both end with the same token, and have a question at the end). However, as the model starts computing the output the computations being done are similar between the fantasy and continuation dataset and the fantasy and no question dataset. This might be because, in the training data for these models, tokens that are likely to follow from a fantasy story might involve simply continuing the story.

4.9 Similarity across prompts

A further extension of Section 4.7 is determining how changing the wording of semantically similar questions impacts the activations of GPT-2 models.

4.9.1 Datasets

To investigate this, a similar experiment to 4.8 is performed, although the genre of the story is kept fixed. Instead, the questions after the stories are varied, to determine how similarly different questions are processed.

(Genre Question 1) Q: What is the genre of this story? Answer in one word A:
(Genre Question 2) Q: For this story, what is the genre? Answer in one word A:
(Genre Question 3) The genre of this story is:
(Genre Question 4) Q: What is the genre of this story? A:

Figure 4.14: Different questions used for the text in the story questions datasets. In this Section, they are always added to fantasy stories.

Alongside the likelihood and continuation questions used in Section 4.8, variations of questions related to the genre of the model are investigated. The purpose of this is to investigate the extent to which model activations are robust to small syntactic changes to the task, or whether a small change in the phrasing of the task leads to completely different activations.

4.9.2 Results

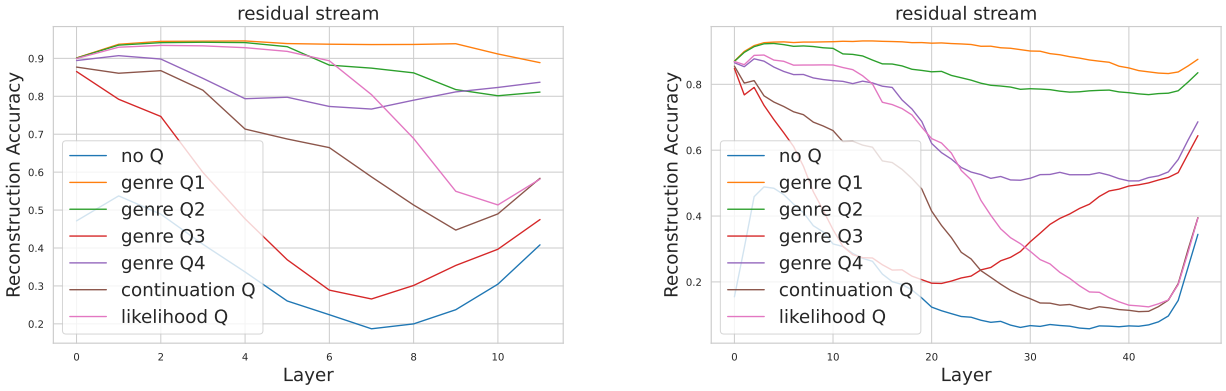

Figure 4.15: PCA reconstruction accuracies for GPT-2 Small and XL, using fantasy stories followed by Genre Question 1 as the comparison to project onto. A single principal component is used.

4.9.3 Analysis

The PCA reconstruction accuracies (Section 4.1.2) for GPT-2 Small and GPT-2 XL, using Genre Question 1 as the baseline dataset, are demonstrated in Figure 4.15.

Looking first at GPT-2 Small, we can see that the Residual Stream activations corresponding to Genre Question 2 are well approximated by the first principal component approximating the activations of Genre Question 1. This is true across all layers of the model.

However, the Genre Question 3 dataset activations are less similar to Genre Question 1 than both the Likelihood and Continuation Questions. Since Question 3 is the most syntactically different, this may suggest that the model relies on superficial similarities to produce outputs, as opposed to understanding deeper semantic similarities.

By contrast, the PCA reconstruction activations for the final layer of GPT-2 XL suggest that the final activations associated with Genre Question 2 are the most similar to Genre Question 1, followed by Genre Questions 3 and 4, and finally by the completely different questions. This aligns with how similar the questions are semantically: Genre Question 2 is identical to Genre Question 1, just rephrased; Genre Questions 3 and 4 ask the same question, but without specifying a single-word response is required; and the Continuation and Likelihood Questions require entirely separate responses.

This suggests that at least in the domain of reading comprehension, GPT-2 XL has learned to use similar activations across a wider range of prompts than GPT-2 Small has. This is similar to work by Grosse et al. [57] which demonstrated that larger language models demonstrated more abstract generalisation patterns than smaller ones across training examples. Figure 4.15 suggests a somewhat converse result: as models become larger, they may become better able to recognise when superficially different tasks require similar computations.

4.10 Summary

Recall that in Section 3.3.1 we defined features as functions of the input space $f : \mathcal{X} \to \mathbb{R}$. Based on the empirical results in this Chapter, where we investigated whether text sharing some human-interpretable feature (i.e. being part of the same dataset) leads to similarities in the activations of GPT-2 models, we make the following claim.

Claim 1. *In GPT-2 style transformers, there exist similarities across different inputs in how some human-interpretable features are represented in activation space.*

The evidence for this can be summarised as follows:

- Clustering was observed based on syntactic features of numeric datasets, despite models answering questions incorrectly (Section 4.4). This demonstrates that even for tasks which are too difficult for models to perform, they are able to learn which features of the task are most pertinent and represent these in the Residual Stream.
- Clustering in the Residual Stream was observed in terms of the output of a transformer in multiple settings (Sections 4.3, 4.6). This confirms that features which are important for the model to represent in order to produce outputs can lead to clustering in the Residual Stream.
- Clustering was also observed in the final position of the Residual Stream for other features of inputs, such as the relation of someone in a reading comprehension task (Section 4.6). This is significant because in this example the relation is useful instrumentally for producing the output, even though the output cannot be predicted from the relation alone.

- Clustering was observed based on the language of input text (Section 4.5). Significantly, this feature is represented in the Residual Stream of these models whilst corresponding to a less structured dataset than the previous examples.
- Clustering in the Residual Stream was observed both based on genres of stories (Section 4.7) and on the question posed after a story (Section 4.8). This is significant as another example of a high-level feature that is represented in the same way across an entire dataset.

Our findings corroborate empirical evidence that neural networks often learn to represent human-interpretable features through specific regions of latent space activations, as was outlined in Section 3.3.

There are several other high-level takeaways from the empirical evidence collected in this Chapter:

- For GPT-2 small, medium, large and XL, the activations of the residual stream are not zero-centred at any layer. This bias in the Residual Stream activations is not explained solely by the Embedding Layer. The Attention Layer outputs also exhibit notable bias throughout the layers of all the GPT-2 models (Section 4.2).
- Different features are first represented by GPT-2 models in different layers. The extent to which the presence of a feature impacts the activations of a model changes between layers (Sections 4.6, 4.9).
- There is some limited evidence that larger models are better at recognising semantic features of text than smaller models, and continue to represent the same feature found in different contexts in the same way (Section 4.9).

Chapter 5

Controlling Transformer Activations

In Chapter 4, we saw evidence that many kinds of high-level features of inputs were being represented in the activations of transformer models. However, the empirical investigation did not demonstrate how these features were being represented. This Chapter will begin with a brief argument that **features are represented as directions in activation space** (Section 5.1). This will provide the basis for creating new techniques for developing a group of methods for extracting feature vectors from datasets (Section 5.2), alongside a brief empirical investigation into their efficacy (Section 5.3). This will be followed by descriptions of how feature vectors can be used to detect features in the processing of text (Section 5.5), as well as how to influence the behaviour of language models (Section 5.4). The successful application of these methods (Section 5.6) provides further empirical evidence for the argument that features are represented as directions, as well as offering new methods of interacting with language models in their own right.

5.1 Features as Directions in Activation Space

Currently, there does not exist a consensus in the literature on how features are represented in transformer models. However, the following hypothesis is arguably the most widely accepted answer. We will provide a brief justification for it in this Section.

Hypothesis 2. *For GPT-2 style transformers, human-interpretable features are represented by directions in the Residual Stream.*

There is a mixture of both empirical and theoretical reasons we might expect this to be true. Empirically, note that the majority of existing methods for manipulating LLM Activations (Section 3.7) make this assumption, since they often involve training a linear probe on the activations of LLMs [53] [54]. Training a linear probe to predict the existence of a feature f is equivalent to learning some direction θ in activation space such that high cosine similarity with θ leads to confident predictions that the input contains the feature f. Hence, the direction θ can be interpreted as representing the feature f. The success of these methods suggests that at least some features are represented as directions.

There are also theoretical reasons to expect transformers to represent features as directions in activation space. Some of these are provided by Elhage et al. [34] in the case of neural networks in general, but the argument is even stronger in GPT-2 style models because of the

similarities between Attention Heads and linear maps. Recall that the output of an Attention Head on a Residual Stream tensor $\mathbf{x} \in \mathbb{R}^{l \times d_{model}}$ is given by

$$h(\mathbf{x})_i = \sum_{1 \leq j \leq i} \mathbf{A}^h_{i,j}(\mathbf{x}) * \mathbf{W}^h_{OV}(\mathbf{x}_j),$$

Since $\mathbf{W}^h_{OV}$ is a matrix, this means that if attention scores are treated as fixed then Attention Heads are indeed linear maps. Although attention scores are non-linear, they only impact the relative degree to which different tokens are attended, not how vectors are processed once attended to. The structure of linear maps means that the most natural way to represent features is as directions, because linear maps act consistently on vectors in the same direction (up to scaling). Thus, we might also expect this to be true for Attention Heads.

Although the arguments for features being represented as directions MLP networks are significantly weaker, since the same Residual Stream is used by both Attention Layers and MLP Layers and we expect Attention Heads to require features to be represented as directions, we might expect MLP Layers to in a sense be forced to also represent features as directions.

Although LayerNorm is applied to the residual stream before the Attention and MLP Layers, it doesn't complicate this argument because LayerNorm has minimal impact on the direction of Residual Stream activations (Section 2.2.3).

Together, these arguments provide evidence that features will be represented as directions in the Residual Stream of GPT-2 style language models. This assumption will be made throughout the remainder of this thesis.

5.2 Feature Vectors from Datasets

Given the assumption that features are represented as directions in the Residual Stream, we will look at methods for creating *feature vectors*: directions in Residual Stream space which correspond to how language models represent certain features, at a given layer. Existing work has already investigated several approaches to creating feature vectors. There are various ways one might try to do this, such as training a linear probe on a supervised [54] or unsupervised [53] task, or using the activations on specific prompts [52].

A similar but alternative approach is to modify the approach used by White [8], i.e. taking the mean across all Residual Stream activations at a layer of the transformer across a dataset which shares some feature.

5.2.1 Mean Activations

The most naive possible implementation of this approach is to simply take the mean of Residual Stream activations at some layer of the transformer model, $\mathbf{x}_1, \ldots, \mathbf{x}_k$. These vectors are produced by running the model on some dataset, where inputs share some attribute f. For example, they may all contain the word "woof". Assuming that this feature is represented as a single direction $\mathbf{v}_f \in \mathbb{R}^n$ across all these activations, one might describe the activations as

$$\mathbf{x}_1 = \mathbf{v}_f + \mathbf{e}_1,$$

$$\vdots$$
$$\mathbf{x}_k = \mathbf{v}_f + \mathbf{e}_k.$$

Assuming that the $\mathbf{e}$ vectors are independently distributed about the $\mathbf{0} \in \mathbb{R}^n$ vector, the mean of these vectors is given by

$$\overline{\mathbf{x}} = \mathbf{v}_f + \frac{1}{k}\sum_{i=1}^{k} \mathbf{e}_i$$

which tends towards

$$\overline{\mathbf{x}} \to \mathbf{v}_f$$

as $k \to \infty$ (i.e., as we grow our dataset of uncorrelated examples with the attribute f).

5.2.2 Mean-Centring

A core assumption in 5.2.1 is that the $\mathbf{e}$ vectors for some dataset are independently distributed about the origin. However, as demonstrated both in Cai et al. [40] and Section 4.2, the Residual Stream activations are not uniformly distributed about the origin.

Hence, a more accurate description of the activations might be:

$$\mathbf{x}_1 = \mathbf{v}_f + \mathbf{v}_b + \mathbf{e}_1,$$
$$\vdots$$
$$\mathbf{x}_k = \mathbf{v}_f + \mathbf{v}_b + \mathbf{e}_k.$$

Assuming that the $\mathbf{e}$ vectors are independently distributed about the $\mathbf{0} \in \mathbb{R}^n$ vector, the mean of these vectors is given by:

$$\overline{\mathbf{x}} = \mathbf{v}_f + \mathbf{v}_b + \frac{1}{k}\sum_{i=1}^{k} \mathbf{e}_i$$

which tends towards:

$$\overline{\mathbf{x}} \to \mathbf{v}_f + \mathbf{v}_b$$

as $k \to \infty$ (i.e., as we grow our dataset of uncorrelated examples with the attribute f).

To isolate the vector $\mathbf{v}_f$ without the bias vector $\mathbf{v}_b$, the activations can be *mean-centred* by approximating the bias vector $\mathbf{v}_b$. For some set of Residual Stream activations at some layer of the transformer model $\mathbf{x'}_1, \ldots, \mathbf{x'}_{k'}$, where the $\mathbf{x'}_i$ have no common attribute (ideally drawn from a large subset of the training dataset), the activations can be described as:

$$\mathbf{x'}_1 = \mathbf{v}_b + \mathbf{e'}_1,$$
$$\vdots$$

$$\mathbf{x'}_{k'} = \mathbf{v}_b + \mathbf{e'}_{k'}.$$

Assuming the $\mathbf{e}'$ vectors are independently distributed about the $\mathbf{0} \in \mathbb{R}^n$ vector, the mean of these vectors is:

$$\overline{\mathbf{x}'} = \mathbf{v}_b + \frac{1}{k'} \sum_{i=1}^{k'} \mathbf{e'}_i$$

which tends towards:

$$\overline{\mathbf{x}'} \to \mathbf{v}_b.$$

The value $\overline{\mathbf{x}'}$ can then be used to mean-centre the activations. This allows the extraction of the feature vector $\mathbf{v}_f$:

$$\overline{\mathbf{x}} - \overline{\mathbf{x}'} \approx \mathbf{v}_b + \mathbf{v}_f - \mathbf{v}_b = \mathbf{v}_f.$$

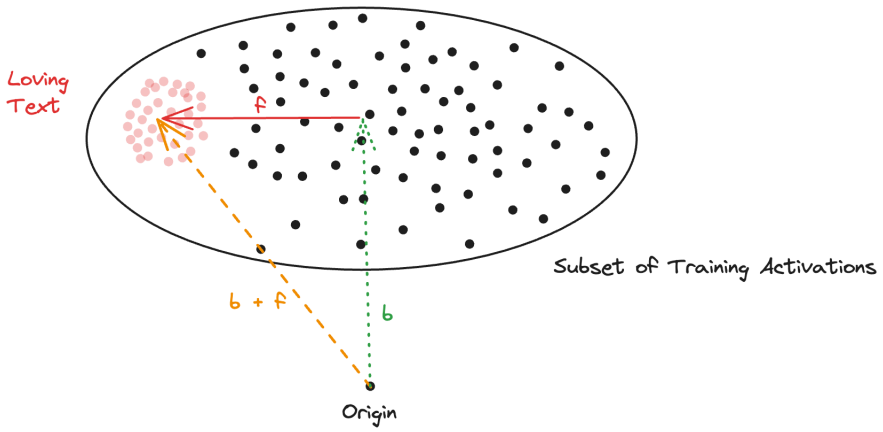

Figure 5.1: An example of mean-centring. If all activations are distributed about bias vector b, and there exist some activations associated with loving text, then just taking the mean of the loving activations will produce $b + f$ vector. The vector f can be isolated by subtracting the bias vector b.

5.2.3 Isolating Specific Features

The approach in Section 5.2.2 might suffer if our dataset has an undesired correlated feature, besides the feature we are looking to extract. This mirrors the issue identified by White [8], which demonstrated that in the CelebA dataset, there was an unexpected negative correlation between the celebrity being male and smiling. However, the nuance of text could mean that this issue is even more pronounced here, as it might be more difficult to identify the unwanted correlates in text due to its complexity. This might mean that simply removing unwanted correlates via resampling might not be possible in this context.

An alternative approach to remove unwanted correlations to isolate specific features is to use the same idea as mean-centring, but subtracting a vector which accounts for both the bias and the unwanted correlation.

For example, suppose there exists a dataset of sports stories, and the corresponding activations can be described as

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{v}_{\text{sports}} + \mathbf{v}_{\text{stories}} + \mathbf{v}_b + \mathbf{e}_1 \\ &\vdots \\ \mathbf{x}_k &= \mathbf{v}_{\text{sports}} + \mathbf{v}_{\text{stories}} + \mathbf{v}_b + \mathbf{e}_k \end{aligned}$$

with the mean given by:

$$\bar{\mathbf{x}} \approx \mathbf{v}_{\text{sports}} + \mathbf{v}_{\text{stories}} + \mathbf{v}_b$$

To extract just $\mathbf{v}_{\text{sports}}$, the activations corresponding to a broader range of stories can be utilized:

$$\begin{aligned} \mathbf{x}''_1 &= \mathbf{v}_{\text{stories}} + \mathbf{v}_b + \mathbf{e}''_1 \\ &\vdots \\ \mathbf{x}''_{k''} &= \mathbf{v}_{\text{stories}} + \mathbf{v}_b + \mathbf{e}''_{k''} \end{aligned}$$

with the mean being:

$$\bar{\mathbf{x}''} \approx \mathbf{v}_{\text{stories}} + \mathbf{v}_b$$

Thus, the difference is:

$$\bar{\mathbf{x}} - \bar{\mathbf{x}''} \approx \mathbf{v}_{\text{sports}}$$

This approach potentially allows for a clearer extraction of the sports feature.

5.3 Feature Vector Experiment

To ascertain empirically how effective these different methods of extracting feature vectors are, we perform a simple experiment. We can unembed feature vectors associated with the different story datasets from Section 4.7, and try to interpret these by finding the token embeddings with the highest and lowest cosine similarity. This utilises the Method described in Section 4.1.4.

5.3.1 Mean Activations

Firstly, we may try to interpret simply the mean of all the Residual Stream activations corresponding to either the fantasy, sports, or sci-fi datasets, at some layer of the model, as outlined in Section 5.2.1. The results in layer 1 are given in Table 5.1.

These tokens do not seem easily interpretable, and do not seem to correspond to words associated with the respective genres (although some for fantasy and sci-fi seem interpretable, such as “magical”, “shining”, and perhaps “destroyed”).

Fantasy Positive	Fantasy Negative	Sci-fi Positive	Sci-fi Negative	Sports Positive	Sports Negative
mine	\x12	rive	\x12	?	rive
mad	?	crumbling	?	\x12	white
Ard	?	ruined	\x16	?	shining
ruined	\x16	mine	?	\x16	ruined
maiden	\x02	Gat	\x02	\x02	sand
shining	?	destroyed	?	\r	gr
Garland	?	Garland	?	?	scra
bra	\r	shattered	?	?	—
sand	?	mag	?	\x0c	struck
grim	InstoreAndOnline	mad	?	InstoreAndOnline	right
crumbling	?	charred	\x0c	?	ground
mag	\x0c	bra	\x1a	\x1a	bree
magical	\x1a	destroy	InstoreAndOnline	?	tall
shattered	rawdownload	Drill	?	?	night
ha	?	war	rawdownload	?	pressed

Table 5.1: The top and bottom 15 tokens by inner product size, after averaging the Residual Stream activations corresponding to the fantasy, sci-fi, or sports story activations in layer 1 of GPT-2 XL. ? Refers to unicode characters.

Using the same method to produce feature vectors in later layers leads to a large degree of similarity between the tokens produced by the different feature vectors. The results are provided in Table 5.2.

This might be evidence that just taking the mean of Residual Stream activations to produce the feature vector is not effective. The convergence to the same tokens across the different examples might be explained by the presence of the bias in the Residual Stream activations (Section 4.2).

Fantasy Positive	Fantasy Negative	Sci-fi Positive	Sci-fi Negative	Sports Positive	Sports Negative
unthinkable	://	unthinkable	://	unthinkable	://
enormous	clips	enormous	clips	enormous	clips
massive	wcsstore	massive	wcsstore	massive	wcsstore
immense	: {	immense	: {	immense	: {
fateful	'/	fateful	'/	fateful	'/
vast	addr	vast	addr	vast	addr
larger	TEXTURE	larger	TEXTURE	larger	TEXTURE
formidable	actionGroup	colossal	actionGroup	obvious	actionGroup
Rebellion	76561	Rebellion	76561	undeniable	76561
colossal	office	secretive	office	intense	office
secretive	Versions	formidable	{	formidable	Versions
obvious	?	unimaginable	76561	Rebellion	${
ill	${	obvious	isEnabled	colossal	?
unimaginable	isEnabled	undeniable	.(	sole	dates
intense	.(	intense	?	unimaginable	.(

Table 5.2: The top and bottom 15 tokens by inner product size, after averaging the Residual Stream activations corresponding to the fantasy, sci-fi, or sports story activations in layer 30 of GPT-2 XL.

5.3.2 Mean-Centring

If we instead produce the feature vector by mean-centring (Section 5.2.2) with a random subset of the training dataset, we get the results as described in Table 5.3.

Fantasy Positive	Fantasy Negative	Sci-fi Positive	Sci-fi Negative	Sports Positive	Sports Negative
Elven	sit	cosmic	Plays	swirling	sit
warrior	BUS	interstellar	USD	clenched	oS
jewel	KK	asteroid	Opinion	sto	?
enchantment	ipes	disemb	KK	pounding	CM
realms	USD	dimensional	ippi	flames	ETA
magical	Reply	Celestial	oS	longing	iz
Celestial	rep	wasteland	TA	clasp	Sit
Primordial	Bye	explorer	votes	grit	tics
elf	oS	fireball	yd	gripping	ancies
elemental	National	beings	Reply	trembling	cop
enchanted	Advertisement	adventurer	News	euph	tz
magically	News	teleportation	Reuters	fists	nai
celestial	Plays	loneliness	eret	towering	chool
warriors	Reps	Primordial	ork	loving	Panel
Realm	UP	Artifact	National	roaring	anti

Table 5.3: The top and bottom 15 tokens by inner product size, after mean-centring the Residual Stream activations corresponding to the fantasy, sci-fi, and sports story datasets. Results are for layer 1 of GPT-2 XL.

The corresponding tokens immediately appear to be much more interpretable, and this remains true for deeper layers: we no longer have degeneration to the same collection of words for all genres, regardless of layer. This is demonstrated in Table 5.4.

Fantasy Positive	Fantasy Negative	Sci-fi Positive	Sci-fi Negative	Sports Positive	Sports Negative
enchanted	itto	humankind	itto	exhilar	Anyway
mystical	Anyway	humanity	ETF	victorious	Anyway
magical	Tank	mankind	endors	triumph	ALSO
awakened	endors	millennia	Tweet	cheering	Otherwise
sorce	NYT	interstellar	Marketable	triumphant	unspecified
enchant	zzi	galactic	nodd	adrenaline	uggest
runes	referen	civilization	confir	glory	listings
mystic	crap	cosmic	Asked	trembling	FY
enchantment	pez	awakened	vc	victory	Asked
arcane	Stuff	starship	referen	chants	zzi
treasures	nodd	teleportation	Stuff	crimson	separately
awakening	guys	ensl	nice	celebration	Basically
shaman	Jerry	sentient	adder	cheers	Sources
goddess	ETF	destiny	Asked	clenched	Basically
visions	Asked	enslaved	pretty	soaring	downgrade

Table 5.4: The top and bottom 15 tokens by inner product size, after mean-centring the Residual Stream activations corresponding to the fantasy, sci-fi, and sports story datasets. Results are for layer 29 of GPT-2 XL.

Overall, this provides evidence that mean-centring offers an improvement over simply taking the mean of activations, as our theoretical analysis suggested.

5.3.3 Isolating Specific Features

Despite the general success of mean-centring in this example, notice also that the words associated with sports have an emphasis on words we might associate with sports stories, and not just sports themselves. This is likely because our dataset isn't just about sports, but about sports stories.

We can try to correct this by using the method of isolating specific feature vectors proposed in Section 5.2.3, approximating the $\mathbf{v}_{\text{stories}}$ as the mean of the fantasy, sci-fi, and sports Residual Stream activations at any given layer.

This leads to the results we observe in Table 5.5.

Highest Inner Product Tokens	Lowest Inner Product Tokens
applause	habitable
clinch	sentient
cheering	Cosmic
cheers	colonization
spectators	interstellar
scoreboard	civilization
clin	planetary
Wembley	inhabited
announcer	cosmic
scorer	humankind
hoop	planet
referee	extrater
coaches	quarantine
drib	technologically
whistle	Humanity

Table 5.5: The top and bottom 15 tokens by inner product size, after isolating the sports vector as detailed in Section 5.2.3. Results are for layer 29 of GPT-2 XL.

5.3.4 Summary

This brief investigation provides some empirical evidence that features are represented as directions in Residual Stream activations of GPT-2 models, since we were able to find directions in activation space which represented a specific feature.

It also helps to confirm our theoretical prediction that mean-centring (Section 5.2.2) might be a superior method for producing feature vectors than simply taking the mean of activations. It also demonstrated that our hypothesised method for isolating specific features (Section 5.2.3) works as desired. These insights will be useful when refining methods for Activation Additions in the next Section.

5.4 Activation Additions

The methods used for extracting feature vectors in Section 5.2 can then be used to manipulate the behaviour of language models. Following from the work of Turner et al. [52] and Li et al. [54], this Section will propose a method of adding feature vectors to the activations of a single layer of the model at inference time. The details of this algorithm can be found in Algorithm 1. This algorithm mirrors that used by Turner et al. [52], but incorporates the more general methods of generating feature vectors developed in Section 5.2.

The function f here warrants further explanation. Given a feature vector $\mathbf{v}$ of shape $\mathbb{R}^{1\times d_{\text{model}}}$, and Residual Stream activations $\mathbf{x}_l$ of dimension $\mathbb{R}^{|S|\times d_{\text{model}}}$ (where $|S|$ is the number of tokens in S), there are various ways that $\mathbf{v}$ could be added to $\mathbf{x}_l$. Examples include padding with zero vectors to the left or right, or repeating the vector across all $|S|$ positions of the Residual Stream.

Note the number of hyper-parameters here: even once the model, prompt, feature and baseline datasets have been determined, optimal values for the injection coefficient, target layer,

Algorithm 1 Activation Additions

Input:
M = language model
p = user prompt
n = maximum output length
$(\mathbf{x}_i)_{i \leq N}$ = feature dataset
$(\mathbf{x}'_i)_{i \leq N'}$ = baseline dataset
c = injection coefficient
l = target layer
f = feature vector extender
Output:
S = steered output text

Calculate the mean of the feature activations:
for $1 \leq i \leq N$ **do**
  $\mathbf{h}_i \leftarrow M$.forward($x_i$).activations[$l$]
  $\bar{\mathbf{x}} \leftarrow \bar{\mathbf{x}} * \frac{i-1}{i} + \mathbf{h}_i * \frac{1}{i}$
end for

Calculate the mean of the baseline activations
for $1 \leq i \leq N'$ **do**
  $\mathbf{h'}_i \leftarrow M$.forward($x'_i$).activations[$l$]
  $\bar{\mathbf{x}}' \leftarrow \bar{\mathbf{x}}' * \frac{i-1}{i} + \mathbf{h'}_i * \frac{1}{i}$
end for

Calculate the steering vector
$\mathbf{v} \leftarrow \bar{\mathbf{x}} - \bar{\mathbf{x}}'$

Autoregressively produce output using the steering vector
for $i \leq n$ **do**
  $\mathbf{v}' = f(\mathbf{v}, S)$
  $\mathbf{x}_l \leftarrow M$.forward($S$).activations[$l$]
  $t \leftarrow M$.continueForward($c\mathbf{v}' + \mathbf{x}_l$).text
  $S \leftarrow S$.append(t)
end for

and feature vector extender have to be found. Currently the best methods for doing this involve using a grid search and keeping the feature vector extender fixed as padding to the right.

5.5 Feature Classification

Closely related to the problem of trying to control the output of LLMs is detecting the presence of certain features during the forward pass of the model. One potential method for doing this is outlined by [8] in the domain of generative image models, using the dot product of latent activations with some feature direction to classify images. Given the method of extracting feature vectors in arbitrary layers of an LLM in Section 5.2, a similar process can be used to classify how LLMs process text prompts. Intuitively, it works on the basis that a large dot product with a feature vector should indicate the presence of that feature, and smaller dot products should indicate its absence. The details of this algorithm can be found in Algorithm 2.

Again, the function f warrants further explanation. It is in some ways the opposite of f in Section 5.4, in that it specifies how to produce a vector $\mathbf{g}_i \in \mathbb{R}^{1 \times d_{\text{model}}}$ from the Residual Stream activations of a single layer $\mathbf{x}_l$ of dimension $\mathbb{R}^{|\mathbf{z}_i| \times d_{\text{model}}}$ (where $|\mathbf{z}_i|$ is the number of tokens in $\mathbf{z}_i$). This could be done by taking the average across all token activations, or just taking the final token activation. This reduction is necessary to compute the dot product with $\mathbf{v}$.

Also, note the number of hyper-parameters here: once the model, prompt, feature and baseline datasets have been determined, optimal values for the target layer, feature vector extender, and threshold have to be found. Currently the best methods for doing this again involve using a grid search.

Algorithm 2 AtDotLLM

Input:
M = language model
$(\mathbf{x}_i)_{i \leq N}$ = feature dataset
$(\mathbf{x}'_i)_{i \leq N'}$ = baseline dataset
$(\mathbf{z}_i)_{i \leq K}$ = evaluation dataset
l = target layer
f = vector extractor
t = threshold

Output:
$(\hat{\mathbf{z}}_i)_{i \leq K}$ = feature prediction

$(\hat{\mathbf{z}}_i)_{i \leq K} \leftarrow 0$

Calculate the mean of the feature activations
for $1 \leq i \leq N$ **do**
  $\mathbf{h}_i \leftarrow M$.forward($x_i$).activations[$l$]
  $\bar{\mathbf{x}} \leftarrow \bar{\mathbf{x}} * \frac{i-1}{i} + \mathbf{h}_i * \frac{1}{i}$
end for

Calculate the mean of the baseline activations
for $i = 1$ **to** N' **do**
  $\mathbf{h'}_i \leftarrow M$.forward($x'_i$).activations[$l$]
  $\bar{\mathbf{x}}' \leftarrow \bar{\mathbf{x}}' * \frac{i-1}{i} + \mathbf{h'}_i * \frac{1}{i}$
end for

Calculate the feature vector
$\mathbf{v} \leftarrow \bar{\mathbf{x}} - \bar{\mathbf{x}}'$

Classify each input to be evaluated based on dot product with the feature vector
for $1 \leq i \leq K$ **do**
  $\mathbf{g}_i \leftarrow f(M$.forward($z_i$).activations[$l$]$)$
  $y_i \leftarrow \langle \mathbf{g}_i, \mathbf{v} \rangle$
  if $y_i \geq t$ **then**
    $\hat{z}_i \leftarrow 1$
  else
    $\hat{z}_i \leftarrow 0$
  end if
end for

5.6 Experiments

This section will present some initial examples of using the algorithms presented in Sections 5.4 and 5.5 to perform output manipulation and feature classification.

5.6.1 Activation Addition Examples

Some results from implementing Algorithm 1 are presented below. The text in the left-hand columns represents unsteered model completions; the text in the right-hand columns results from applying the Activation Addition Algorithm. The hyper-parameters used to produce each example are provided, and the datasets are as described in Section 4.7. Right padding is used as the feature vector extender for all examples.

Here is a story: A man walks into a bar. He orders a drink. He takes his drink and goes to the bathroom. When he comes out, he is wearing a hat with an American flag on it. The bartender says, "You know what? I think you should take that off." The man says, "I'm not going to take that off." The bartender says, "Why not?"	**Here is a story:** A man who wants to be a part of the world's first time-traveling football game has discovered that his plans are being thwarted by the rules of an ancient and mysterious game. The referee is called upon to help, but his own knowledge of the rules will likely leave him stranded in an interstellar war between two sides trying to find out what it is they're fighting about. humanity vs

Table 5.6: An example activation addition with the following hyper-parameters: Model = GPT-2 XL, coefficient = 180, layer = 25, feature dataset = scifi stories, baseline dataset = sports stories.

Yesterday, my son was out kicking a football. Then, he went to the gym and I saw him. He was on his way to work. He had no shoes on and he was just playing with his feet." My son's father said that when I told him about the incident, he didn't know what to say or do. "He said 'I'm sorry,' but it wasn't something that I could say," says my dad.	**Yesterday, my son was out kicking a football. Then, he** went to the lake and was awakened by a strange creature. It seemed to be a magical creature that appeared in her home, and she found it there with an enchanted enchantment on it. She took it from the lake and told her that if she could find some kind of magical power in this enchanted land of the Lost Princesses (or perhaps even better) she could restore them back to life.

Table 5.7: An example activation addition with the following hyper-parameters: Model = GPT-2 Small, coefficient = 60, layer = 6, feature dataset = fantasy stories, baseline dataset = sports stories.

A girl once had a goose egg. She was not the only one to have it. And now, as we enter the fourth quarter of the season, I've got to say that my favorite moment in football is when I'm on the field and watching the Dallas Cowboys take on their best team in seven years. It's like a ghost town where you can't even get out of bed until your phone rings	**A girl once had a goose**'s tail, but alas, her wild side has wane. A little beauty once brightened the sky with a maiden shine, but now she withers in the dark...and never shines again. A beautiful maiden once longed to dance for her fair queen, but now she is forgotten...and never ever shines again. She's all fair, I'll sing thee sweet sleep's

Table 5.8: An example activation addition with the following hyper-parameters: Model = GPT-2 Small, coefficient = 150, layer = 6, feature dataset = shakespeare like text, baseline dataset = training subset.

5.6.2 Genre Classification Examples

The story datasets of varying genres (Section 4.7) are used here to develop a classifier based on the genre of a piece of text, using Algorithm 2.

The feature dataset is a subset of the fantasy stories; the baseline dataset is a subset of the training dataset; and the evaluation dataset consists of a separate subset of fantasy, sci-fi and sports stories, and a subset of the training dataset.

Instead of setting a threshold, we vary this hyper-parameter to calculate ROC-AUC metrics. Results from running the AtDotTwo algorithm for GPT-2 small and calculating the ROC-AUC score for each layer on each of the Residual Stream activations, Attention Layer outputs, and MLP Layer outputs, are shown in Figure 5.2. The best layer to use for classification is the output of the 10th Attention Layer, as this has an AUC score of close to 0.95.

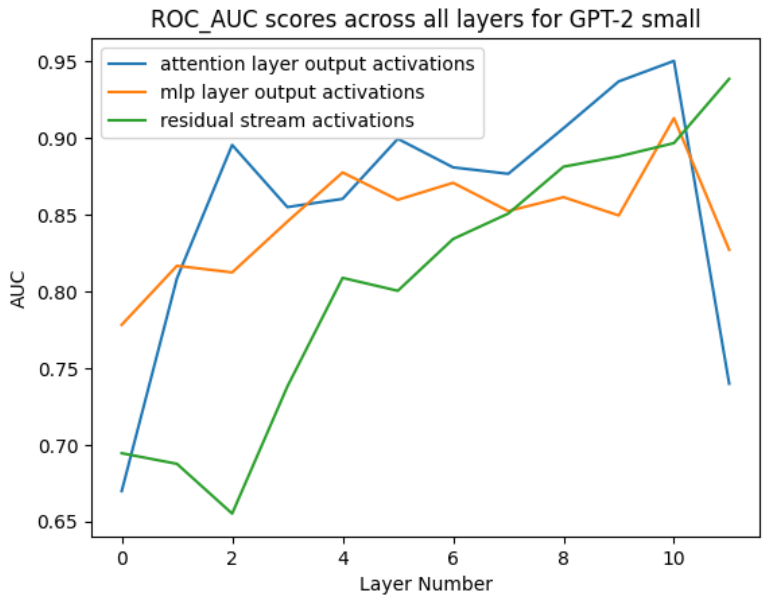

Figure 5.2: AUC scores for the AtDotTwo experiment at each layer of GPT-2 small, for Residual Stream, mlp, and Attention Layers.

This can be repeated for GPT-2 XL. The results are shown in Figure 5.3. Note that on average performing the experiment with the output of the Attention Layers leads to higher AUC scores, but using the Residual Stream at intermediate layers has lower variance. This might represent a general tradeoff that may exist across other models.

5.7 Summary

This Chapter described new methods for finding feature representations of GPT-2 style language models and used these to propose alterations to methods for better understanding and controlling these models. Examples of successful applications of these algorithms provided further empirical evidence that these models represent features as directions.

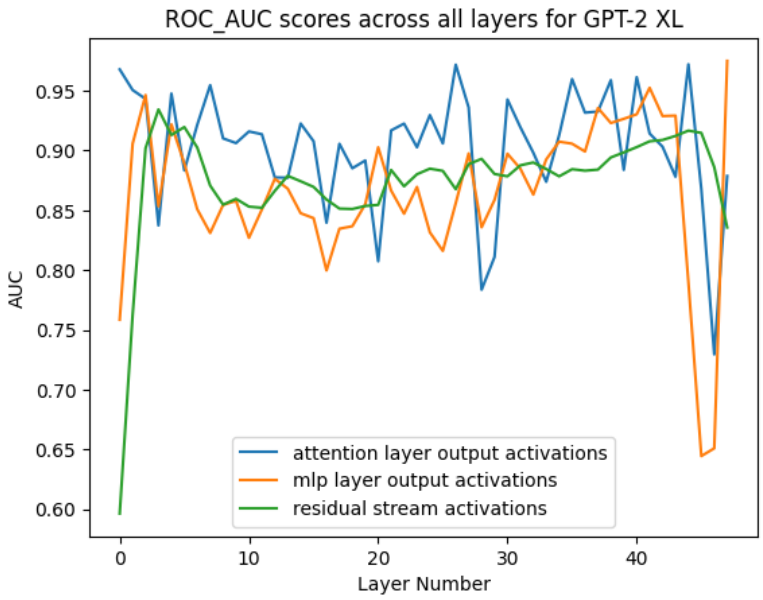

Figure 5.3: AUC scores for the AtDotTwo experiment at each layer of GPT-2 XL, for Residual Stream Activations, MLP Layer outputs, and Attention Layer outputs.

Chapter 6

Evaluation

To properly judge the significance of the results provided throughout this thesis, serious consideration must be taken to the limitations of this work. It is also worth considering its significance to wider debates surrounding LLMs.

6.1 Limitations of Empirical Investigations

Although several methods were developed and applied to several different datasets and features, the results are by no means exhaustive enough to claim an understanding of how features are represented in GPT-2 style models. The first issue is that we only analyse a small handful of features, and thus it is difficult to draw strong conclusions about when features of text will be represented in the same way in language models.

The second is that although some of our methods provide evidence of certain kinds of representations, they do not rule out other classes of representations. This is most clearly demonstrated by our use of t-SNE plots: clustering here suggests that there is some kind of distance-based clustering occurring, but it does not provide any evidence about the kind of clustering besides this.

6.2 Feature Vector Creation

Recall the method used to create feature vectors highlighted in Section 5.2. Whilst feature vectors created with this method were used to successfully perform activation steering (Section 5.6), there exist various other methods of forming feature vectors which have been used successfully in other contexts [54] [53] [52]. It is worth considering the benefits and drawbacks of using the method highlighted in 5.2, as opposed to these other methods.

The most similar method comes from Turner et al. [52], which uses the activations from single inputs to produce feature vectors. This method is simpler and computationally cheaper than those presented here, since it requires running inference on only a single pair of inputs as opposed to an entire dataset. However, a more principled way of performing counterbalanced subtractions is presented here, and using a dataset should allow for the representation of features which cannot be captured as easily in a single input.

Li et al. [54] finds feature directions by training a linear probe on the individual Attention Head values, finding the direction which best predicts whether the output has some desired feature using gradient descent. This has two immediate advantages over our method, as looking at individual Attention Heads allows for more fine-grained control of the feature vector, and learning a probe means that unwanted correlates won't impact the final feature vector. An advantage of using the mean activations from a dataset is that it is not limited to the supervised setting.

Burns et al. [53] provides another method for finding feature directions by training linear probes in some unsupervised settings, through a method called Contrast-Consistent Search (CCS). Although promising, it is not immediately clear how to apply CCS across all unsupervised domains, since it replies on consistency properties about how a network might represent truth.

There are some other general limitations with the methods provided here. One limitation is that the method of mean-centring provided here relied on a somewhat incomplete picture of the structure of the activations in GPT-2 style models, since it was assumed that mean-centring implied the existence of a single cluster away from the origin. However, Cai et al. [40] found that 2 clusters were sometimes necessary to describe the activations. This might suggest that a different approach to mean-centring might be required to produce more useful feature vectors.

Another is that although all the methods implicitly assume Hypothesis 2, taking the mean of the activations associated with some dataset requires some slightly stronger version of this, i.e. that a feature is represented with the same direction across different inputs. Although Section 4 suggested this was true for the GPT-2 models studied here, if it doesn't hold for larger models then this method will cease to be effective.

6.3 Activation Additions Limitations

Even once a feature vector has been produced, there are some key limitations with the Activation Addition Algorithm (Algorithm 1).

One of these is the difficulty in evaluating the outputs of a language model, especially in unsupervised contexts. Ideally, activation additions can be used to change some features of the outputs, with as little detriment to the capabilities of the model as possible. Metrics should be developed to measure whether the desired feature is more present in the outputs of the model; as well as metrics to ensure the overall capabilities of the model are preserved. Turner et al. [52] look at the increase in the frequency of certain desired words to detect the presence of a desired feature in the output of the model, and use perplexity on training data as a proxy measure for the model's capabilities. Although useful, these are still imperfect proxies and could be developed further.

Another difficulty is in optimising the hyper-parameters associated with activation additions, such as the injection layer and the injection coefficient. Given proxy measures this can be done automatically through hyper-parameter optimisation procedures such as grid search, but this was not conducted in this thesis.

Another limitation is that features might be represented in multiple directions instead of just one, and these features might need to be added with varying injection layers and injection coefficients. The method as currently presented only allows for activation additions in a single layer, in a single direction.

These difficulties limited the applications of Activation Additions to a small number of examples in this project. This means that a systematic analysis of whether mean-centring can lead to better performance than that exhibited by Turner et al. [52] was not performed, limiting the significance of our results.

6.4 AtDotLLM Limitations

Although it hasn't been developed as a method of classifying features of input text, many of the ideas used by AtDotLLM (Algorithm 2) are used implicitly in other investigations in this area. Furthermore, a rigorous analysis of this Algorithm on relevant benchmarks, such as Sentiment Analysis, was not completed. It seems unlikely to surpass state-of-the-art algorithms in this area, due to the strong performance of LLMs such as GPT-4 without further fine-tuning.

The significance of this Algorithm is thus best understood as complementing the other methods developed in Chapter 5, since it can be used to evaluate the quality of feature representations.

6.5 Implications of Feature Representation

In the wider discussion around language models, one point of intense discussion is the extent to which language models are genuinely intelligent, in the sense that they understand the meaning of the language they are processing (past just being able to generate convincing language). A related question is the extent to which these models work by memorisation, which could involve using very different heuristics for different inputs which superficially appear very similar; or whether they are learning by forming robust generalisations. Some influential works such as Bender et al. [10] have taken the position that despite the meaninglessness of LLM outputs, "humans mistake LM output for meaningful text".

Although speculative, Section 4 could be interpreted as providing some evidence against this claim. It demonstrated that across a range of tasks, GPT-2 style models represented features of datasets in the same way in the Residual Stream. Although a lot of the datasets we have looked at often shared syntactic similarities, this does suggest that language models do in some sense understand high-level features of text, and are not working entirely based on the memorisation of training data.

Chapter 7

Conclusions and Future Work

There are several directions in which this project could be extended, which might be categorised into three broad directions of future study: collecting further empirical evidence on the structure of language model activations; applying insights to detect distribution shift; and further developing methods similar to Activation Additions.

7.1 Further Empirical Investigations

Chapter 4 could be extended to further understand the structure of language model activations by addressing the shortcomings noted in 6.1.

- Further methods could be developed to investigate the geometric representation of different datasets. Methods which make causal predictions based on the activations would be particularly significant.
- A greater range of features could be investigated. It would be particularly useful to see investigations of features which are important for better controlling LLMs such as deception, truthfulness, and sycophancy.
- More kinds of clustering algorithms could be used to study Residual Stream activations, such as PCA and UMAP.

The empirical investigations could also be expanded to consider a wider range of models. In particular, there could be further investigations into:

- GPT-2 style models with more parameters.
- Transformer models with different architectures. The recently open-sourced llama-2 models [58] represent a great candidate for this, although the hardware required to run these models makes it substantially more difficult.

7.2 Detecting Distribution Shift

One aim of interpretability research is to understand when a network is about to use a fundamentally different kind of reasoning in order to produce an output compared to what we might

have wanted or expected. This would be particularly useful for detecting unwanted behaviour in the presence of distribution shift.

The AtDotLLM Algorithm described in Section 5.5 presents one possible approach to predicting robust behaviour under distribution shift, since it could be used to detect the presence of features in the computations of a model. This could be used to detect the representation of desired features, as well as to check unwanted features are not being represented.

More robust versions of AtDotLLM could be developed to help with this by considering more than just inner products. This could perhaps be through quantifying the extent of change in MLP Layers that are induced by small changes in the input (since MLP Layers are not linear, and so their outputs can vary greatly with a small amount of change to inputs).

7.3 Better Activation Additions

The Activation Addition algorithm described in Section 5.4 offers a potential improvement to an exciting new method for interacting with language models. It has the potential to offer a much cheaper alternative to fine-tuning, since it only requires performing inference on a handful of inputs to form feature vectors (as opposed to performing further training to a model). It could also be used to improve various safety-related features of models, such as by reducing the likelihood of models outputting harmful text.

However, there are several areas of further development required before Activation Additions can be applied to state-of-the-art language models. One area is in addressing the shortcomings listed in Section 6.3, particularly with respect to conducting a systematic analysis of the modification to the construction of steering vectors proposed in this project.

Another area of development suggested by Li et al. [54] is to look not only for single directions, but subspaces constituting multiple basis vectors. This might be more robust than looking for single directions, as a feature may be represented across multiple directions instead of just one. Indeed, this is found by Li et al. [54] when investigating truthfulness, where at least two directions which correlate with truthfulness are found. It would be interesting to explore activation additions using multiple feature vectors, instead of just one.

7.4 Conclusion

In conclusion, this thesis presents evidence that the activations of GPT-2 style language models are highly structured, and uses this evidence to improve upon new methods for controlling language models by developing new ways to create feature vectors. These new perspectives on understanding and controlling LLMs offer the potential to help address some of their shortcomings in the future.

Bibliography

[1] Richard Ngo, Lawrence Chan, and Sören Mindermann. The alignment problem from a deep learning perspective, 2023. URL https://arxiv.org/abs/2209.00626. [Accessed: 4th May 2023]. pages 3

[2] Tilman Räuker, Anson Ho, Stephen Casper, and Dylan Hadfield-Menell. Toward transparent ai: A survey on interpreting the inner structures of deep neural networks, 2023. URL https://arxiv.org/abs/2207.13243. [Accessed: 2nd August 2023]. pages 3

[3] OpenAI. Gpt-4 technical report, 2023. URL https://arxiv.org/abs/2303.08774. [Accessed: 22nd July 2023]. pages 3, 16

[4] Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. Palm 2 technical report, 2023. URL https://arxiv.org/abs/2305.10403. [Accessed: 20th May 2023]. pages 3

[5] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan

Carter. Zoom in: An introduction to circuits, 2020. URL https://distill.pub/2020/circuits/zoom-in. [Accessed: 2nd August 2023]. pages 3, 17, 18

[6] Gabriel Goh, Nick Cammarata †, Chelsea Voss †, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 2021. doi: 10.23915/distill.00030. https://distill.pub/2021/multimodal-neurons. pages 3, 17

[7] Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment, 2017. URL https://arxiv.org/abs/1704.01444. [Accessed: 3rd June 2023]. pages 3, 17

[8] Tom White. Sampling generative networks, 2016. URL https://arxiv.org/abs/1609.04468. [Accessed: 13nd June 2023]. pages 3, 4, 18, 19, 47, 49, 57

[9] Neel Nanda. Transformerlens, 2022. URL https://github.com/neelnanda-io/TransformerLens. [Accessed: 2nd August 2023]. pages 3

[10] 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*, FAccT '21, page 610–623, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383097. doi: 10.1145/3442188.3445922. URL https://doi.org/10.1145/3442188.3445922. pages 5, 65

[11] Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2023. URL https://arxiv.org/abs/1706.03741. [Accessed: 20th May 2023]. pages 7, 16

[12] Abubakar Abid, Maheen Farooqi, and James Zou. Persistent anti-muslim bias in large language models. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '21, page 298–306, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384735. doi: 10.1145/3461702.3462624. URL https://doi.org/10.1145/3461702.3462624. pages 7

[13] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html. [Accessed: 2nd August 2023]. pages 7, 8, 12, 19

[14] Fabien Roger. Some ml-related math i now understand better. https://www.lesswrong.com/posts/Tu8DYNCg63F4HYbXE/some-ml-related-math-i-now-understand-better, 2023. [Accessed: 29th August 2023]. pages 11, 12, 20

[15] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn

Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html. [Accessed: 2nd August 2023]. pages 14, 19

[16] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791. pages 15

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf. pages 15

[18] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. 1986. URL https://api.semanticscholar.org/CorpusID:62245742. pages 15

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. pages 15

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf. [Accessed: 20th May 2023]. pages 15, 16

[21] 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, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423. pages 15

[22] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018. URL https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf. [Accessed: 20th May 2023]. pages 16

[23] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. URL https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf. [Accessed: 20th May 2023]. pages 16

[24] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165. [Accessed: 20th May 2023]. pages 16

[25] 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, 2020. URL https://arxiv.org/abs/2001.08361. [Accessed: 20th May 2023]. pages 16

[26] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=iBBcRUlOAPR. pages 16

[27] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety, 2016. URL https://arxiv.org/abs/1606.06565. [Accessed: 3rd June 2023]. pages 16

[28] Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, Tony Wang, Samuel Marks, Charbel-Raphaël Segerie, Micah Carroll, Andi Peng, Phillip Christoffersen, Mehul Damani, Stewart Slocum, Usman Anwar, Anand Siththaranjan, Max Nadeau, Eric J. Michaud, Jacob Pfau, Dmitrii Krasheninnikov, Xin Chen, Lauro Langosco, Peter Hase, Erdem Bıyık, Anca Dragan, David Krueger, Dorsa Sadigh, and Dylan Hadfield-Menell. Open problems and fundamental limitations of reinforcement learning from human feedback, 2023. URL https://arxiv.org/abs/2307.15217. [Accessed: 1st September 2023]. pages 16

[29] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional ai: Harmlessness from ai feedback, 2022. URL https://arxiv.org/abs/2212.08073. [Accessed: 3rd June 2023]. pages 16

[30] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf. pages 16

[31] 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, 2023. URL https://arxiv.org/abs/2305.04388. [Accessed: 3rd June 2023]. pages 16

[32] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. pages 17

[33] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751, 2013. pages 17, 18

[34] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/toy_model/index.html. [Accessed: 2nd August 2023]. pages 18, 20, 46

[35] Sid Black, Lee Sharkey, Leo Grinsztajn, Eric Winsor, Dan Braun, Jacob Merizian, Kip Parker, Carlos Ramón Guevara, Beren Millidge, Gabriel Alfour, and Connor Leahy. Interpreting neural networks through the polytope lens, 2022. URL https://arxiv.org/abs/2211.12312. [Accessed: 2nd August 2023]. pages 18

[36] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper_files/paper/2014/file/109d2dd3608f669ca17920c511c2a41e-Paper.pdf. pages 18

[37] Randall Balestriero and Richard G. Baraniuk. Mad max: Affine spline insights into deep learning. *Proceedings of the IEEE*, 109(5):704–727, 2021. doi: 10.1109/JPROC.2020.3042100. pages 18

[38] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016. URL https://arxiv.org/abs/1511.06434. [Accessed: 13nd June 2023]. pages 18

[39] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1558–1566, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/larsen16.html. pages 18

[40] Xingyu Cai, Jiaji Huang, Yuchen Bian, and Kenneth Church. Isotropy in the contextual embedding space: Clusters and manifolds. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=xYGNO86OWDH. pages 18, 23, 24, 30, 48, 64

[41] nostalgebrist. interpreting gpt: the logit lens. https://www.alignmentforum.org/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens, 2020. [Accessed: 29th August 2023]. pages 19, 20

[42] 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, 2023. URL https://arxiv.org/abs/2303.08112. [Accessed: 20th May 2023]. pages 19

[43] A. D. Wyner. Random packings and coverings of the unit n-sphere. *The Bell System Technical Journal*, 46(9):2111–2118, 1967. doi: 10.1002/j.1538-7305.1967.tb04246.x. pages 20

[44] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL http://www.jmlr.org/papers/v9/vandermaaten08a.html. pages 21

[45] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020. URL https://arxiv.org/abs/1802.03426. [Accessed: 20th May 2023]. pages 21

[46] Jonathon Shlens. A tutorial on principal component analysis, 2014. URL https://arxiv.org/abs/1404.1100. [Accessed: 20th May 2023]. pages 21

[47] Catherine Yeh, Yida Chen, Aoyu Wu, Cynthia Chen, Fernanda Viégas, and Martin Wattenberg. Attentionviz: A global view of transformer attention, 2023. URL https://arxiv.org/abs/2305.03210. [Accessed: 2nd August 2023]. pages 21

[48] Kawin Ethayarajh. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1006. URL https://aclanthology.org/D19-1006. pages 22, 23

[49] Elena Voita, Rico Sennrich, and Ivan Titov. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4396–4406, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1448. URL https://aclanthology.org/D19-1448. pages 22, 23, 31

[50] Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. Representation degeneration problem in training natural language generation models, 2019. URL https://arxiv.org/abs/1907.12009. [Accessed: 13nd June 2023]. pages 23

[51] Jiaqi Mu, Suma Bhat, and Pramod Viswanath. All-but-the-top: Simple and effective post-processing for word representations, 2018. URL https://arxiv.org/abs/1702.01417. [Accessed: 13nd June 2023]. pages 23

[52] Alex Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte MacDiarmid. Activation addition: Steering language models without optimization, 2023. URL https://arxiv.org/abs/2308.10248. [Accessed: 29th August 2023]. pages 24, 47, 55, 63, 64, 65

[53] Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision, 2022. URL https://arxiv.org/abs/2212.03827. [Accessed: 2nd August 2023]. pages 25, 46, 47, 63, 64

[54] Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model, 2023. URL https://arxiv.org/abs/2306.03341. [Accessed: 13nd June 2023]. pages 25, 46, 47, 55, 63, 64, 67

[55] Sid Black Beren Millidge. The singular value decompositions of transformer weight matrices are highly interpretable, 2022. URL https://www.alignmentforum.org/posts/mkbGjzxD8d8XqKHzA/the-singular-value-decompositions-of-transformer-weight. [Accessed: 29th August 2023]. pages 29

[56] Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019. [Accessed: 16th May 2023]. pages 30, 33

[57] Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamilė Lukošiūtė, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Samuel R. Bowman. Studying large language model generalization with influence functions, 2023. URL https://arxiv.org/abs/2308.03296. [Accessed: 29th August 2023]. pages 44

[58] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor

Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288. [Accessed: 3rd August 2023]. pages 66