ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Understanding Features in Superposition in Transformer Language Models

Master Thesis

Kasper Munk Rasmussen

November 13, 2023

Advisors: Dr. Mor Geva, Google Research, Tel Aviv University and Prof. Mrinmaya Sachan, ETH Zürich

Department of Computer Science, ETH Zürich

Abstract

It has been proposed that a hidden state space of dimensionality d in a neural network can, in some cases, linearly represent and make use of much more than d one-dimensional features in non-orthogonal one-dimensional subspaces. This is described as superposition. However, there has been limited work on investigating this in realistic pre-trained Transformer language models [16]. Although the notion of a feature is contested, we argue that the pattern of a token-bigram AB qualifies as a feature. We study sets of bigrams with cardinality greater than d, and their representation and usage in the spaces of hidden states of pre-trained models, using linear probing and causal interventions.

We find evidence that there exist sets of bigram features with cardinality larger than d, for which linear probes can identify the presence of the features with good recall and precision. Building upon ideas about the aggregate structure of the representations in superposition in a space, we find that the distribution of feature vectors may be related to their importance for prediction, and that the bigram features do not accord with what has, in recent work, been called isotropic superposition.

Using causal interventions, we study how the linear representations may be used by the model. Some bigrams have likely token continuations, and causal investigations suggest a relationship between feature directions and model predictions of such tokens. Linearly encoded information about a feature, erased from one hidden state, will in some cases be available in later states. The effects of concept erasure on overall model performance vary greatly depending on the combination of the bigram and the hidden state for which the erasure is performed. Although the exact requirements for what qualifies as superposition may have some ambiguity, we consider the results as evidence supporting at least a weak formulation of superposition in Transformer language models.

Contents

Chapter 1

Introduction

Transformer language models have shown impressive results in a wide variety of settings [39] [11] [36]. Despite their success, as is the case with many other types of neural networks, the inner workings of pre-trained Transformer language models are not well understood [22] [15].

One line of work, *mechanistic interpretability*, aims to understand such models by decomposing the computation into smaller circuits that operate on linearly encodings of features [35] [16] [18]. In [16], the authors investigate simple non-Transformer models and propose the term 'superposition': "*In this paper, we use toy models — small ReLU networks trained on synthetic data with sparse input features — to investigate how and when models represent more features than they have dimensions. We call this phenomenon superposition.*" They propose the term *superposition hypothesis* for the idea that the fact that specifically neuron activation spaces are not easily interpretable is due to such superposition: "*Concretely, in the superposition hypothesis, features are represented as almost-orthogonal directions in the vector space of neuron outputs. Since the features are only almost-orthogonal, one feature activating looks like other features slightly activating.*" From this perspective, a preliminary goal of understanding the inner workings of models in depth involves the identification of such feature directions and showing how they are used in the computation of the model.

The idea of superposition bears resemblance to ideas in the field of hyperdimensional computing, which also relies on properties of high-dimensional spaces [30]. In Appendix B, we perform simple simulations of some aspects of the behaviors of high-dimensional spaces, such as the behavior of almost-orthogonal vectors. Understanding models in terms of linear representations has been explored in many works, such as [24] and [41], but with limited focus on the number of features represented.

In [25], which is closely related to the work of this thesis[1], the authors investigate the existence of a variety of features in the vector spaces of MLP activations of pre-trained Transformer language models. Using a sparse probing methodology, the work aims to find a limited set of neurons that encode the feature. While it has been pointed out that the MLP entry-wise activation functions give privilege to the standard basis of MLP activation space and that there are therefore reasons for attempting to interpret individual or small sets of neurons [15] [21] [8], the idea of a small subset is not inherent to the idea of superposition. As the Transformer language models are residual networks, information is transmitted through the residual stream. Since it is of interest to determine feature representations and their usage downstream, we focus on representations in the residual stream and consider MLPs mainly in terms of their effects on representations in the residual stream.

The idea of superposition, as described in [16], focuses primarily on one-dimensional or binary features that can be represented in one-dimensional subspaces. Such subspaces can potentially be identified with linear probing. Linear probing is a methodology widely used to interpret deep learning models, in which a linear model is trained on the hidden states of a model to predict a variable of interest. This provides insight into what kind of information is encoded in the hidden state [1] [3]. However, the aspect of how this information is used by downstream circuitry is not directly addressed by linear probing. Without additional techniques or modifications, it is primarily a correlational methodology [41] [6] [5]. In [47], the authors introduce a theory of usable information called predictive V-information, and linear probing can be understood within this framework as investigating aspects of linear information. Further work on *concept scrubbing* [6] considers the possibility of intervening on hidden states in a principled way to remove the information about a variable of interest that is accessible by a linear model. In principle, linear probing, V-information, and concept scrubbing should allow one to describe more concretely what features are represented linearly in superposition and aspects of how the model uses such information.

Concurrent work [48] [44] [45] proposes methods for investigating the causality and usage of features that are linearly encoded in one-dimensional subspaces. Here, causal interventions are used to alter the hidden state in ways that involve a candidate feature direction vector, such as adding a vector or projecting out a specific direction.

While the term 'feature' is widely used in many lines of work, there is no direct consensus on the definition [12] [26], and original work on superpo-

[1]The work was published early in the process when we had already started focusing on linear probing as an avenue, and the part of the work that focuses on bigram features inspired us to consider large sets of bigrams to show superposition.

sition [16] [15] keeps the definition partly open, with an emphasis that features should generally be understood as human-interpretable, though this property is hard to define. [29] opts for a more general description of features as functions of the input and defines additional properties on top of such a general definition. [4] argues with [12] for *representational pragmatism*, where features are relative to the goals of the researcher. This stands in contrast to some parts of [16] and [35], which more or less implicitly assume that features can be inherent to a model and that mechanistic interpretability involves identifying features which are, in some sense, real or natural. In this sense, we might call such a perspective *representational realism*. [33] proposes a theory explaining the neural scaling laws observed for large language models, in which models learn specific *quanta* that correspond to prediction rules. Such a theory is arguably related to representational realism in its assertion that there are discrete patterns native to the data distribution that are learned by the model. We draw on these different perspectives in our definition and selection of features of interest.

Through the writings in [35], [16], [18], and [25], we get the impression that the idea of superposition is best understood as being situated within a package of ideas. We will try to describe this package view while acknowledging and emphasizing that most likely no researchers will claim to hold such a view. However, we believe it is worth articulating as a reference for discussions, and we might call it the *'strong view'*: Large trained models have a finite set of inherent native features, in the sense of representational realism. These features are represented linearly in subspaces. This set of features is very large because non-linear processing in the model allows for a way of solving the problem that not all the subspaces can be mutually orthogonal. These subspaces not only contain information about the features but are also native to the model in a sense that modules that act on these spaces perform specific computations by accessing information in these subspaces and write to new subspaces in a localized way. Such localized computations form circuits in the model. It is these features that are in superposition in a model. Most of the model's ability to perform well can be explained through reference to such features, their subspaces, and the circuits formed. Computations performed by such circuits are, in some sense, symbolic, and there might exist a mapping between a computer program that operates on variables and control flow that has a form of causal fidelity to the computations performed by the model. Being in possession of such a mapping to a program would be the supreme version of understanding a model.

The main goal of this work is to understand if and how superposition occurs in the hidden spaces of realistic pre-trained Transformer language models. Investigating the form of superposition implied by the strong view is difficult. We will investigate a weaker view, which we will describe in terms of linear accessibility and the usage of large sets of features that are not nec-

essarily native to the model. We will use methods of probing and causal interventions, taking into account the many conceptual considerations we have described so far. In short, we will argue for an approach to such an investigation: The approach will be to train a large number of probes and validate their performance on a distribution in which one is interested in understanding the model's behavior, and validate specific feature directions as being causally involved in the model's computation using causal interventions on hidden states.

In Chapter 2, we will further discuss the notion of features and aim to make the language for features more precise for our use case. We will then argue that bigram patterns fit both this definition of a feature and other considerations for the meaning of a feature, and that they form a useful way for studying the linear encoding of many features in the residual stream of Transformer language models. We will also describe the *probing data sets* we associate with such bigram features. In Chapter 3, we consider the correlational aspect of probing under the terminology of feature accessibility. We investigate the accessibility of bigram features in the residual stream of a medium-sized Transformer language model and how many such bigram features are accessible. We further look into how the feature vectors identified by the linear probes relate to each other in terms of cosine similarity and perform an investigation based on the hypothesis that features that are more "important" for good predictions should have feature vectors that are more distant from the other feature vectors. In Chapter 4, we move on to consider aspects of feature representations that are not only correlational and do so by discussing what one might mean by saying that a model is *using* a feature and linear representations of such features. We conduct experiments involving causal interventions on hidden states in order to assess whether the feature directions for bigram features identified by linear probes are being used and get mixed results. We also conduct experiments using concept scrubbing in order to investigate if such methods can be used to understand how models use linear representations of features.

In Appendix A, we provide a description of the relevant parts of the architecture of the Transformer language models that we are analyzing, and the notation for the different hidden states that are being investigated and intervened upon.

Chapter 2

Feature Types and Data

2.1 Defining Feature Types

[16] contains a section describing some aspects of what could be meant by the term 'features,' arguing that leaving it open might be useful. For our use case, we will need language that allows us to talk more precisely about what we mean. While [16] partly argues that features are generally human-interpretable, even if they might not remain so for very advanced models, the human-interpretable aspect of features is obviously very difficult to make precise. However, the bigram features we work with are arguably mostly considered human-interpretable. We follow [29] in thinking of features as deterministic functions of the data and believe that it could be useful to let the human-interpretable aspect be something built on top of a more generic definition like the one we use. As part of an attempt to talk clearly about features, we also use the type-token distinction [37] in this context, which allows us to distinguish it from the general word 'feature' with its various conceptions. Thus, in the following, 'feature type' is supposed to be understood as just a 'feature,' the idea being that specific inputs will contain specific instances of this general type.

- Let $\mathcal{X}$ be a set that forms the domain of the input data. In our case, this would be the set of possible sequences for which a language model is then to predict the next token.
- A *feature type set* $\mathcal{Z}$ on $\mathcal{X}$ is a set of feature types.
- A *feature type* $z \in \mathcal{Z}$ is a function $z : \mathcal{X} \to S_z$ mapping input x to an element in the *feature type domain* S_z (a set) of the feature type z.
- A feature type $z \in \mathcal{Z}$ is *binary* if its feature type domain is $S_z = 0, 1$.
- A feature type set $\mathcal{Z}$ is binary if all feature types $z \in \mathcal{Z}$ are binary feature types.

2.2 A Bigram Feature Type Set

We will consider a specific binary feature type set corresponding to bigrams that are pairs of tokens.

As mentioned, it might be desirable to understand models as representing features that are 'interpretable to humans'. It is not clear what exactly this would mean. For a binary feature type z, this could mean that when presented with examples x_i, a human being could learn to perform well as a classifier according to target labels $z(x_i)$ in such a way that the human's classification performance is good according to a certain metric. Bigrams are patterns that can easily be detected by a human, though depending on the tokenizer, some bigrams might be ambiguous,[1] but on the other hand, not all bigrams are intuitively thought of as a 'natural' pattern: (for example) is a pattern that many humans would view as more 'natural' than the bigram (., The), even if this latter bigram is just as common.

Another question about the feature type set is what we in the introduction called representational realism: It could be argued that with the terminology we have now, one is completely free to select any feature type set one wishes, and then study how the processing within the model relates to the feature types. Some may argue instead that in some way or another, some feature type sets are more 'native', 'natural', or 'inherent' to the model, so that there is something that would resemble the 'true feature type set' or at least a spectrum of how true this feature type set is.

If models have native features, we believe the following simple consideration to be important: For a model which is optimized to minimize cross-entropy on the next-token prediction task, it is beneficial to consider a feature type set with feature types that relate to patterns that are relevant for next-token prediction. In this way, if there is a native feature type set, it seems reasonable to think that such feature types are related to prediction. While many complex patterns in text might be relevant for next-token prediction, the simplest n-gram statistics of the training data set contain patterns which are relevant for next-token prediction.

This last argument for viewing bigrams as features is related to the proposal in [33] that models learn quanta, which are prediction rules that have a certain probability of being useful for next-token prediction, and that power-law neural scaling might be linked to a power-law distribution of the probabilities of these quanta. As shown in Section 2.2.2 below, bigrams also follow such a power-law and can, in many cases, be linked to a kind of prediction rule given the trigram statistics of data. An argument against this, how-

[1] For example, byte pair encoding can give rise to tokens consisting of different numbers of whitespaces, in which case it would require more work for a human to decide.

ever, is that trigram statistics do not form deterministic quanta as the theory mostly implies.

Focusing on n-grams has additional benefits that relate to the above concerns: Both the presence and the location of a specific n-gram in a sequence of tokens can be easily determined algorithmically and by humans. Thus, these considerations, in conjunction with the fact that interpreting models is generally difficult, lead us to focus on some of the simplest patterns we can imagine: n-grams and their statistics in the dataset. Denoting the vocabulary set by $\mathcal{V}$, an n-gram is a contiguous sequence of n tokens $(v_1, ..., v_n) \in \mathcal{V}^n$. When $n = 1$, we call it a unigram; when $n = 2$, we call it a bigram. It is clear that for predicting the next token, the previous word is in many cases relevant, just as the previous two words are relevant, and we will focus on bigrams. For our vocabulary $\mathcal{V}$ with a vocabulary size of $V \approx 50,000$, the number of possible bigrams is $|\mathcal{V}^2| = 2.5$ billion.

Given our focus on bigrams, we will use the symbol $\kappa \in \mathcal{V}^2$ to refer to a bigram in the abstract as the tuple of two tokens. To avoid confusion, we will not use the type-token distinction for bigrams and instead refer to a specific instance of a bigram in a sequence of tokens as an 'instance of a bigram' [37].

We now have two main options when defining the feature type set. Given an input sequence x, we can consider either whether the bigram occurs at some place in the sequence or if it occurs as the last two tokens in the sequence. We take the latter approach, and thus for a bigram κ, we define the binary feature type such that given a sequence $x = (x_1, ..., x_{T-1}, x_T) \in \mathcal{V}^T$ of length T, we have

$$z_\kappa(x) = \begin{cases} 1 & \text{if } (x_{T-1}, x_T) = \kappa, \\ 0 & \text{otherwise.} \end{cases}$$

meaning it is 1 if the sequence ends with the bigram. In principle, our feature type set is then $\mathcal{Z} = \{z_\kappa | \kappa \in \mathcal{V}^2\}$, though in practice, we can only consider a subset.

2.2.1 The Pile

The Pile [19] is a large webscale text dataset used for training language models, including all the Pythia models considered in this work. The datset includes news articles, computer code, academic articles, forum discussions and more. In this work we will use a subset of The Pile: T322K is approximately 322 thousand sequences of 600 tokens from The Pile Train selected from a randomly chosen part of The Pile Train. Documents from the set were only used if they were at least 600 tokens long and in this case only the initial 600 tokens were used. T10K and T77K refers to respectively 10 thousand and 77 thousand sequences of 600 tokens from sequences from

The Pile Validation Set that were capped in the same way. The motivation behind having sequences of such length is that it makes experiments easy by storing the data sets as a matrix.

The fact that the data used is limited in size as well as the restriction to the beginning of sequences of certain length can in some sense introduce bias into both probe training and evaluations. Futhermore [9] argues for using multiple varied data sets when interpreting models, showing that one will otherwise fall under an "interpretability illusion" as to how a direction encodes a feature.

As with the discussion about the notion of a feature, some aspects apart from sampling error, are here partly conceptual. It might be argued that unless one subscribes to a very strong form of representational realism, the aim of understanding a model will mostly be bound to a specific data distribution, in this case the distribution induced by selecting sequences from The Pile as described. We call this distribution the interpretation distribution. In our case, apart the biases mentioned, the interpretation distribution agrees with the training distribution and as argued in the previous section this has some advantages for the representational realism concern, but in other cases one could imagine that the interpretation distribution and training distribution do not agree.

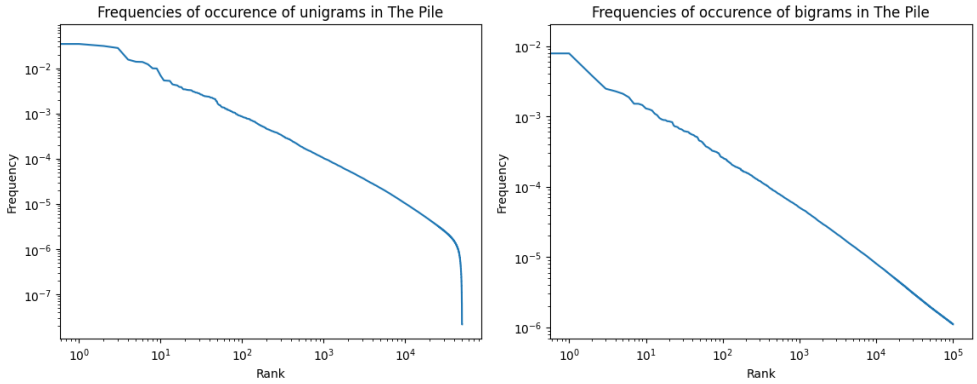
2.2.2 Feature Frequency

For a binary feature type z, it is natural to consider the associated feature frequency: Letting $X \sim P_{\mathcal{X}}$ be a random input from the input domain, with distribution according to the interpretation distribution, then $z(X)$ is then a random variable and the feature frequency for z is $P(z(X) = 1) = \mathbb{E}_{P_X}[z(X)]$.

Since a model given an input sequence x gives predictions for the next-token for each prefix of the sequence (see Appendix A.7), it seems reasonable to consider $P_{\mathcal{X}}$ to be the empirical distribution induced by the set of all subsequences of the sequences in our dataset, which would also be helpful because we can then estimate the feature frequency for each bigram by counting occurences of bigrams on a sample of the interpretation distribution, in our case we use T77K.

In Figure 2.1 we show rank-frequency plot for unigram and bigrams in T77K. Here the x-axis denotes that bigrams are sorted according to the number of occurences in the data in descending order. The most frequent bigrams obviously has higher probability of occuring and as we move to less and less frequent bigrams, the probability falls in a way that can be approximated as a power-law where the frequency $p(\kappa)$ of a bigram κ with rank r_κ is approximated as $p(\kappa) \propto r^{-\alpha}$, which forms a straight line on a log-log plot.

This means that most bigram feature types are very rare which puts limitations on our ability to investigate such feature types because of the large amount of data needed to find occurrences. However this problem is not unique to our bigram feature type set. Especially for stronger forms of representational realism this is also a concern given that many patterns one might consider meaningful features are bound to be very infrequent [25] [16].



(a) Unigrams **(b)** Bigrams

Figure 2.1: Rank-frequency plots for occurrence frequencies for bigrams and unigrams in The Pile (T77K)

2.3 Probing Data Sets for Bigram Feature Types

In the investigations to follow, we will associate with a bigram $\kappa = (v_1, v_2)$ a data set $\mathcal{S}_\kappa$ of contiguous subsequences from T322K called the *probing dataset* for κ. The construction follows [25], which considers a dozen bigrams as part of a more general investigation of various features in Transformer models. The probing data set for a bigram has a structure and some properties:

- Training probing dataset is the collection of 3 set $S^{train}_{pos}, S^{train}_{1neg}, S^{train}_{2neg} \subset \mathcal{V}^T$ of sequences where all sequences in each set is of length $T = 22$.
  - All sequences in the training probing data set exists as contiguous subsequences in T322K.
  - For a sequence x in the training probing dataset there is a ground truth *next-token* for the sequence, which is the token found immediately after the sequence in T77K from which the sequence is derived.
  - All sequences $x \in S^{train}_{pos}$ end with the bigram κ, that is, the last two tokens of x are v_1 and then v_2. In other words $\forall x \in S^{train}_{pos}.z_\kappa(x) =$

1. This set is called the *positive training sequences for the bigram* κ, an element is a "positive sequence".

- All sequences $x \in S^{train}_{1neg}$ have the property that the second-last token $x_{T-1} = v_1$ and $x_T \neq v_2$. That is, the ending of the sequence agrees only with the first token in κ. This is called the 1-negative training sequences for κ.
- All sequences $x \in S^{train}_{2neg}$ have the property: $x_{T-1} \neq v_1$ and $x_T = v_2$. That is, the ending of the sequence agrees only with the second token in κ. This is called the 2-negative training sequences for κ.
- The sets have sizes $|S^{train}_{pos}| = 800, |S^{train}_{1neg}| = 1600, |S^{train}_{2neg}| = 1600$.

- The validation probing dataset $S^{val}_{pos}, S^{val}_{1neg}, S^{val}_{2neg} \subset \mathcal{V}^T$ are 3 sets with the same properties, except that
  - $|S^{train}_{pos}| = 100, |S^{train}_{1neg}| = 200, |S^{train}_{2neg}| = 200$
- The sequences in the probing dataset are derived from distinct locations in T322K.
- It is worth emphasizing that $\mathcal{S}_\kappa$ is off-distribution in the sense that sequences in $\mathcal{S}_\kappa$ do not represent uniformly randomly sampled sequences or subsequences from to T322K or The Pile in general. We emphasize this because it becomes relevant below.

Since some bigrams are very infrequent, it is not possible to construct training probing data sets efficiently for all bigrams, so we consider bigrams that are more frequent.

The way the probing dataset $\mathcal{S}_\kappa$ is used, is that wrt. a model M such as Pythia-410m, we are interested in how M processes sequences in S_κ. For a sequence x, we will primarily be interested in the residual stream $r^s_T(x)$ at the last position T at some depth s (see hidden space notation in A.8). Thus we denote

$$\mathcal{X}^{train}_{pos}(r^s_T) = \{r^s_T(x) | x \in S^{train}_{pos}\}$$

as the positives at depth s (where position T is assumed), define the other sets for validation, 1-negatives and 2-negatives, etc. similarly.

2.4 The Problem of Probing Data Sets

While we can easily determine whether or not a specific sequence ends with a bigram, and there is therefore no label-noise in the probing datasets, a major problem remains: Because for all bigram feature types the feature frequency is very low (which is the case for many other features one can consider), in most cases we would be required to construct such an off-distribution probing dataset as described above. The way such a probing

dataset is constructed introduces a kind of bias or noise in downstream analyses of the associated bigram feature type: While the positives are always positives and can be said to be an unbiased sample of positives that are distributed as $P(X|z(X) = 1)$, the negatives could be selected in many other ways. This will impact the resulting model parameters of the probe classifier and the identified feature direction derived from these model parameters.

It seems that one should therefore distinguish between, on one hand, the bigram feature type for which the ideal probing dataset is simply a very large on-distribution sample, and on the other hand, the feature that is "implicitly" being probed for by the details of the probing dataset construction. The problem is, however, that such a feature is not describable with our terminology for feature types: The function that determines whether a sequence ends with a bigram κ is the same as the function that determines whether a sequence ends with bigram κ *and* does not end with the kinds of bigrams that our 1-negatives and 2-negatives end with."

Chapter 3

Feature Accessibility

As we see it, the basic hypothesis is that a large number of one-dimensional feature types are represented in high-dimensional spaces using one-dimensional subspaces, and that the model performs computation by accessing this information and making use of it. In this chapter, we will investigate a necessary condition for such access, namely accessibility. Independently of whether a part of the model in fact makes use of the feature by accessing the available information in a space to perform some computation, one can discuss whether the feature type would, in principle, be accessible. We will try to make precise what is meant by accessibility and use it to formulate and investigate a narrower version of the hypothesis that features are in superposition in Transformer language model hidden spaces. This investigates whether we can have a large number of feature types be linearly accessible. We do this by training and evaluating linear classifiers. At the end of the chapter, we will then consider the weight parameters of the linear classifiers as feature directions to draw connections to previous work about superposition discussed in [16].

3.1 Defining Feature Accessibility

For an data domain $\mathcal{X} \times \mathcal{Y}$, a binary feature type set $\mathcal{Z}$ on $\mathcal{X}$, a neural network $M : \mathcal{X} \to \mathcal{Y}$, and interpretation distribution $\mathcal{P}_{X,Y}$ on which we are aiming to understands the model's behavior, we have the following definitions.

- If M is a residual neural network of S residual blocks, processing the input $X \sim P_X$, with residual stream space $\mathcal{H}$, we say that the residual stream of hidden states evolves like

$$h_0(X) = f_{embed}(X) \in \mathcal{H}$$

$$h_{s+1}(X) = h_s(X) + f_s(h_s(X)) \in \mathcal{H}$$

for $s \in \{1, ..., S\}$ where s is called the *depth* and $f_s : \mathcal{H} \to \mathcal{H}$ is the *block* at depth s, and $f_{embed} : \mathcal{X} \to \mathcal{H}$ is the embedding layer.[1]

- By the term accessibility we are aiming to talk about what information is in principle available in the representations. Thus in a more general context than the restriction to linear representations we might say that for a feature type $z \in \mathcal{Z}$ the "informational feature accessibility" at depth s, is the mutual information $I(h_S(X); z(X))$ under the marginal $\mathcal{P}_X$.
- In the context of the linear representations of binary features, if $\mathcal{H} = \mathbb{R}^d$, for a family $\mathcal{V}$ of linear models $\{\eta_\theta(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b | \theta = (\mathbf{w}, b) \in \mathbb{R}^d \times \mathbb{R}\}$ and loss function $L : \mathcal{H} \times \{0, 1\} \to [0, \infty)$, we might define the linear L-accessibility of z at depth s as the Bayes risk

$$\alpha(z, s, L) = \min_{\eta \in \mathcal{V}} \mathbb{E}_{P_X}[L(z(X), \eta(h_s(X)))]$$

- We would then approximate the linear L-accesibility of z at depth s using empirical risk minimization, by training a linear classifier η_{probe} called a *(binary) linear probe* on a probing data set (that may or may not be distributed as $\mathcal{P}_{X,Y}$) and evaluating the empirical risk:

$$\hat{\alpha}(z, s, L) = \hat{R}(\eta_{probe}) = \frac{1}{|\mathcal{D}_{val}|} \sum_{(x_i, y_i) \in \mathcal{D}_{val}} L(z(x_i), \eta_{probe}(z_i))$$

  for a validation set $\mathcal{D}_{val}$ that should generally be an unbiased sample from the interpretation distribution.
- In a restricted sense a necessary condition for superposition of feature types in a hidden state $h_s(X)$ is that there is a large number of feature types with good linear L-accessibility. We might say that a hidden state $h_s(X) \in \mathbb{R}^d$ has linear L-accessibility ε-superposition wrt. a feature type set $\mathcal{Z}$ if there exists a subset $\mathcal{Z}' \subseteq \mathcal{Z}$ such that both $|\mathcal{Z}'| > d$ and for all $z' \in \mathcal{Z}'$ the $\alpha(z', s, L) < \varepsilon$. If d' was the greatest cardinality among sets for which this is the case, the ratio $\frac{d'}{d}$ might also be an interesting characterisation of the degree of total linear L-accessibility ε-superposition.
- In practice however we might want not use a loss function L but instead evaluate the classifier on a performance metric such as F1 or AUC-PR that are not strictly loss function. In this case we might for example say that a hidden state $h_s(X) \in \mathbb{R}^d$ has linear AUC-PR-accessibility δ-superposition wrt. a feature type set $\mathcal{Z}$ if there exists

[1] Most of the ideas presented should also apply for non-residual networks with hidden states $h_i(X) \in \mathcal{H}_i$ where i is some form of index of the specific hidden state in the network one is interested in.

a subset $\mathcal{Z}' \subseteq \mathcal{Z}$ such that both $|\mathcal{Z}'| > d$ and for all $z' \in \mathcal{Z}'$ the the AUC-PR validation performance is at least δ.

- There is something unelegant about letting the accessibility be defined wrt an arbitrary loss function or classification performance metric. In 4.1.1 we describe $\mathcal{V}$-information as introduced by [47]. We could define the linear accessibility of z at depth s as the $\mathcal{V}$-information $I_{\mathcal{V}}(h_s(X) \to z(X))$, where $\mathcal{V}$ is still the class of linear classifiers, and believe it could be an interesting direction for future work to investigate if it is conceptually/theoretically and experimentally possible to quantify the amount $\mathcal{V}$-information in a hidden state.

3.1.1 The Problem of Positional Binding

With respect to the previous section, it would be most intuitive to think of the residual network as operating on simple hidden state spaces $\mathbb{R}^d$. However, for residual models such as CNN residual networks operating on images, the residual space often comprises elements of shape $W \times H \times d$, where W and H represent the width and height (or downsampled width and height), forming a grid of d-dimensional representations at each position. Likewise, as described in Section A.2, in our GPT models, we in principle have a residual space of shape $T \times d$, where T is the number of tokens in the sequence. In both cases, representations are 'bound' to a position, either token position or position in the image. In both cases, there is usually some part of the architecture that facilitates the flow of information between representations at different positions: in CNNs, the convolutional layers, and in Transformer models, the MSA modules. This gives rise to a problem when trying to interpret and understand models in terms of the representations at a given position, which we call 'the problem of positional binding'.

Many feature types one might care about are not necessarily of a very local character. For properties such as 'the sentence is sarcastic' or 'the sentence is written in French', the sentence might have specific tokens that are more related to the property than others, but it is not obvious how such features should relate to the representations at specific positions. [44] investigates representations of sentiment and finds information about sentiment to be 'summarized' at positions related to stopwords and punctuation. [32] finds that the information about the city of the Eiffel Tower is primarily contained in the residual stream for the Tower and is only transferred to later positions when this information is used for predicting Paris. For a bigram AB, the information about the identity of the bigram might be accessible in the residual stream for B, but even if it was not, the information would, in most cases, be accessible in a subspace of the space associated with the residual stream for A and B considered together $\mathbb{R}^{2\times d}$. The problem of positional

binding should thus be considered when drawing conclusions, and we will consider it more later.

3.2 Experiment

We wish to conduct an experiment to assess the linear accessibility superposition with respect to the bigram feature type set in the residual stream for the position corresponding to B in a bigram AB at low depths. We will try to do so by training thousands of probes for combinations of bigrams and depths and evaluate their performance on T77K. This is done on the Pythia-410m model [7].

3.2.1 Hypothesis

Informal hypothesis: Residual stream spaces linearly represent many more bigrams than there are dimensions.

Specific hypothesis: For the early residual stream hidden states $r_T^s(X)$, $s = 1$ and $s = 2$ of the Pythia-410m and Pythia-70m models, there exists sets of bigram feature types z_κ larger than the dimensionality of $r_T^s(X)$ for which the linear accessibility is good. We consider linear F1-accessibility 0.9 to be good. In other words we hypothesize linear F1-accessibility 0.9-superposition in early residual streams at depth $s = 1$ and $s = 2$.

3.2.2 Method

Linear probing is a common technique in interpretability and beyond [3]. While people can use probing for various reasons, in the context of this chapter *linear probing is a methodology for assessing linear accessibility*: To investigate the linear L-accessibility of a binary feature type z in a hidden state $h_s(X)$ we train a linear model on a probing data set and use the acquired probe to evaluate the loss L on a suitable validation set.

The problem of positional binding means that we have to make a choice as to how we want to investigate the accessibility of a bigram feature type. We will be looking at the residual stream at the last position.

For the bigram feature type z_κ we approximate the linear F1-accessibility of z_κ by training a logistic regression classifier. A logistic regression classifier is a function $C_\theta(\mathbf{x}) : \mathbb{R}^d \to (0, 1)$ with parameters $\theta = (\mathbf{w}, b) \in \mathbb{R}^d \times \mathbb{R}$ st

$$C_\theta(\mathbf{x}) = \sigma(\mathbf{w} \bullet \mathbf{x} + b)$$

where σ is the sigmoid function. By treating $C_\theta(\mathbf{x})$ as inducing a conditional distribution $q(y|\mathbf{x})$[2], the logistic regression classifier is trained by minimiz-

[2] $q(y = 1|\mathbf{x}) = C_\theta(\mathbf{x})$ and $q(y = 0|\mathbf{x}) = 1 - C_\theta(\mathbf{x})$

ing the cross-entropy $\mathbb{E}_{(x,y)\sim p_{data}}[-\log q(y|\mathbf{x})]$ on the empirical distribution p_{data} induced by the training data set using some optimization technique.[3]

For a bigram feature type z_κ, for a depth s, we train a logistic regression classifier (using `sklearn` default settings) by constructing a binary classification training dataset where the input vectors are the vectors from $\mathcal{X}^{train}_{pos}(r^s_T), \mathcal{X}^{train}_{1neg}(r^s_T), \mathcal{X}^{train}_{2neg}(r^s_T)$ (see Section 2.3) and the the positives have label 1 and both 1-negatives and 2-negatives have label 0. We construct a binary classification validation set accordingly. We call the trained logistic regression classifier $C^s_\kappa : \mathbb{R}^d \to (0,1)$ a *linear probe of* z_κ *as depth* s, and the weight vector $\mathbf{w}^s_\kappa \in \mathbb{R}^d$ associated with the linear probe we call the *probing vector* and will later treat it as a "candidate" for the *feature direction* associated with z_κ at depth s.

While we have an associated validation set $\mathcal{X}^{val}(r^s_T)$ we have to be aware that as mentioned, this data set is off-distribution wrt. T77K. Thus the classification metrics of C^s_κ evaluated on $\mathcal{X}^{val}(r^s_T)$ do not form the full story of the performance of the probe: As mentioned, most bigram feature types z_k have very low feature frequency, while in their associated probing validation set the positives constitute $1/5$ of the sequences. This means that the probing validation data set can be used to give us a good idea about the sensitivity, i.e. $P(C^s_\kappa(x) = 1|z_\kappa(x) = 1)$, but other classification metrics can be biased. Consider precision $P(z_\kappa(x) = 1|C^s_\kappa(x) = 1)$, here if the feature frequency is very low, even if the specificity is very low, the total amount of false positives could dominate the number of true positives.

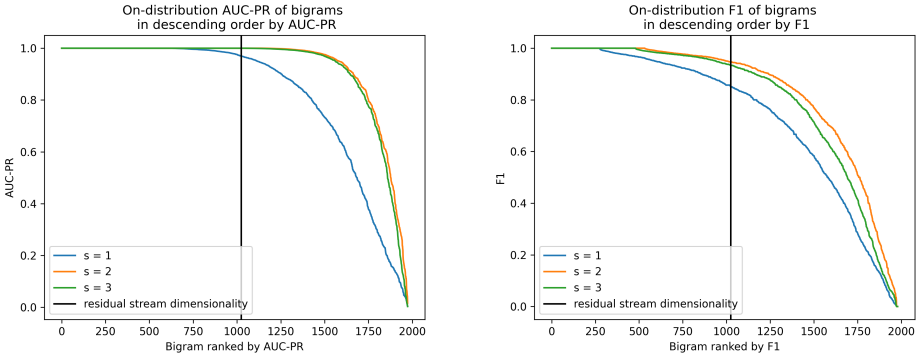
Therefore consider a set $\mathcal{S}_{od}$ of 200 sequences of length $T = 600$ from T10K and evaluate C^s_κ based on the set of $120,000$ vectors $\mathbf{X}^s_{od} = \{r^s_t(x)|x \in \mathcal{S}_{od}, t \in [T]\}$ and associated labels: For an vector $r^s_t(x)$ for some x and t, the associated label will be 1 if the sequence $x_{\leq t}$ ends with the bigram κ, that is if $z_\kappa(x_{\leq t}) = 1$. In this way we are trying to estimate how the probes perform on-distribution wrt. the interpretation distribution.

We wish to determine if indeed the space associated with r^s_T for some depth s is such that many bigram feature types can have good classification, specifically whether there exists a subset $\mathcal{Z}' \subseteq \mathcal{Z}$ such both $|\mathcal{Z}'| > d$ and for all $z' \in \mathcal{Z}'$ the F1 and AUC-PR is good. We consider F1 and AUC-PR because these metrics concern sensitivity and precision. As we will discuss below specificity is of limited interest to us because of the strong class imbalance arising from the low feature frequencies. We will then evaluate on-distribution F1 and AUC-PR for each bigram feature type and order them in decreasing order on a "rank-performance" curve, that allows us to highlight the relationship between the packing and the performance. For a metric such as AUC-PR, the bigram feature whose AUC-PR performance rank is

[3] We use standard `sklearn` settings where the optimizer is LBFGS

$d+1$, consider its AUC-PR performance δ. This probe and its performance thus defines a point on the rank-performance curve where we can say that we have $d+1$ features whose AUC-PR is at least δ in a d-dimensional space. This will give us an evaluation of the degree to which we can say features are packed in the space.

3.2.3 Results and Takeaways



(a) AUC-PR **(b)** F1

Figure 3.1: Rank-performance curves for probes trained on Pythia-410m

In Figure 3.1 we show the before mentioned "rank-performance" for our probes for depths $s \in \{1,2,3\}$ and with AUC-PR and F1. Consider first Figure 3.1a. The x-axis is the rank of the probe associated with the specific depth, this means that for example $x = 1700$ might not correspond to the same bigram for $s = 1$ and $s = 2$ because the rankings of different probes might vary internally between $s = 1$ and $s = 2$. The way we interpret these plots is to say that for the bigrams and associated probes we consider, by looking at the x-axis $x = d+1 = 1025$, for $s = 2$ we can select a set of $d+1$ bigrams where the probe for each will be at least 1, while for $s = 1$ we can select at set of $d+1$ bigrams where the probe for each will be at least 0.958. Similarly for F1-score in Figure 3.1b, for sets $d+1$ bigrams, we get at least F1 of 0.94 for $s = 2$ and F1 of 0.83 for $s = 1$. Since our specific hypothesis formulated in 3.2.1 was that of linear F1-accessibility 0.9-superposition for both $s = 1$ and $s = 2$ we cannot confirm the specific hypothesis from this available data. However, we will discuss reasons why we believe the hypothesis to be true even if the experiment does not confirm it.

There are limitations of the assessments of AUC-PR and F1 because even though we $120,000$ tokens, for some κ and associated C^s_κ the number of tokens that are not true-negatives can be quite small. This especially hold for bigram feature types with very low feature frequencies. This means that

more validation might be needed to evaluate the performance of specific probes but that a general conclusion that the space can indeed allow for many well-performing probes does hold.

We note that the shape of such a "rank-performance" curve also greatly depends on the number of features we are training probes for. For the bigrams we trained probes for we did not find a significant relationship between the performance and the feature frequency of the AUC-PR and or F1 score. This means that for a selected threshold δ we cannot, because of limited compute resources, assess how many bigram feature types as a number d' have an AUC-PR that is at least δ and thereby compute the ratio $\frac{d'}{d}$ which would have been an interesting measure of the the degree of packing of bigram feature types in the d-dimensional space. Furthermore, as will be described below even for the bigrams considered the performance shown in the plots are most likely to be viewed more as a lower bound.

While binary classification for imbalanced data sets is not uncommon in machine learning and therefore has a rich literature [43], the literature on how neural networks, particularly language models, deal with representing features with very low frequency seems to be understudied and could benefit from both theoretical and empirical investigations. Especially, the question of how to think about the relationship between recall and precision is important. In [25], the authors note that for the case where they identify sets of MLP neurons for a feature, *"Low precision and high recall indicates either that the selected neurons are highly polysemantic or the model represents a more general feature than is being probed for. High precision and low recall of the probing classifier may indicate that the identified submodule represents a more specific feature than the feature being probed for"*. As also noted by [25], the trade-off between recall and precision depends on the threshold of the classifier, which is why we also consider AUC-PR. Exactly how a model uses the linear information with its associated recall-precision trade-off might be complex and hard to discern. This is why applying $\mathcal{V}$-information [47] might make more sense than using F1 or AUC-PR, as other classification metrics implicitly assume that models make discrete classifications of features instead of operating with whatever signal is available.

In the case where we associate the classifier C^s_κ and its associated weight vector $\mathbf{w}^s_\kappa$ with a *feature direction* that is intrinsic to the model, which will be discussed in later sections, some of these consideration still apply.

3.2.4 Further Analysis

For $s = 1$ the bigram (to, a) has a AUC-PR of 0.07 and all 1226 positives in the on-distribution sample are incorrectly classified as negatives with the standard threshold. However, using an alternative approach to the construction of the probing dataset, which we will now describe, we can raise the

AUC-PR from 0.07 to 0.996 and the F1 from 0 to 0.98. This suggests that the degree of packing of feature directions in the hidden spaces is greater than what is suggested by the rank-performance curves discussed.

Half-Adversarial Probing Data Sets

The way we try to get better probes is by synthetic alterations based on an idea of embedding-similar bigrams.

For a specific Transformer language model, two bigrams $(a_1, b_1), (a_1, b_2) \in \mathcal{V}^2$ are embedding-similar of Type A or Type B iff:

- Type A embedding-similar: $a_1 = a_1$ and $sim(f_{embed}(b_1), f_{embed}(b_2))$ is "high". If "high" needs to be formalized it has to take into account the fact that $f_{embed}(b_1)$ might be in the k-neighborhood of k nearest neighbours[4] of $f_{embed}(b_2)$ but not the other way around, and that this is most likely very common. Thus, if we want the binary relation to be commutative, we can define "high" as meaning that at least one is within the k-neighborhood of the other for a suitable k.
- Type B embedding-similar: $b_1 = b_2$ and $sim(f_{embed}(a_1), f_{embed}(a_2))$ is "high" with same caveats.

Consider a bigram $\kappa = (a, b)$. We start by having the normal probing dataset $\mathcal{S}_\kappa$. For each 1-negative sequence with probability θ we make a change: A 1-negative is a sequence that ends with (a, b') where $b' \neq b$. With probability θ we choose to make a change to this sequence there b' is replaced with b'' where $b'' \in \mathcal{V}$ is drawn uniformly at random from the k-neighborhood of b. That is, b'' will be a type whose embedding is quite similar to the embedding of b. In this way (a, b'') will be embedding-similar to (a, b). Likewise, for 2-negative sequences that end with (a', b) where $a' \neq a$, we make a substitution where a' is replaced with one of the k types in $\mathcal{V}$ whose embedding is most similar to the embedding of a, and this replacement is done with probability θ. The Bernoulli random variables $X_1, ..., X_{|S_{1neg}|+|S_{2neg}|}$ corresponding to the choice of making a replacement for each sequence are sampled i.i.d from $\text{Ber}(\theta)$.

The original choice of how to select the negative sequences for the probing datasets was already a design choice aimed at creating good probes, and so may have its advantages and disadvantages. For the purpose of assessing the linear accessibility, the design choice should ultimately be judged by its ability to make the on-distribution behavior of the probes good. In principle, what constitutes a good design choice could vary from bigram to bigram.

[4] By k-neighborhood of $f_{embed}(b_1)$ we mean the set S of vectors $f_{embed}(v_i)$ for $i \in [k]$ such that there exists no other $v_j \notin S$ with type embedding $f_{embed}(v_j)$ such that there exists an $s \in [k]$ such that $sim(f_{embed}(v_j), f_{embed}(b_1)) \geq sim(f_{embed}(v_s), f_{embed}(b_1))$

This seems to be the case: Even though for the bigram (to, a) described above, we get an improvement using half-adversarial with $k = 20$, $\theta = 0.2$, and $s = 1$, there are also cases where we get much worse performance than using the standard probing datasets.

On precision for very rare features

Using the half-adversarial probing, one can train a probe for a quite rare bigram. However, when evaluated on-distribution, there may not be any positives. In this case, one can still get an idea about the precision by evaluating the sensitivity on the probing validation set (non-adversarial) and looking at the on-distribution false positives.

Consider the peculiar bigram (this,->). Using half-adversarial probing we can get perfect sensitivity for for example $s = 1$. Its empirical frequency is approximately $4.8 \cdot 10^{-5}$. We use approximately $5 \cdot 10^5$ tokens for evaluation and get no false positives. Thus it seems reasonable to say expect that this probe has a reasonably good precision even if we do not have an exact approximation.

3.3 The Distribution of Feature Directions and Interference

In the previous sections we were interested in the linear accessibility of bigram feature types and used linear classifiers to assess this linear accessibility. We were not directly interested in which exact subspaces encoded the information. There is a certain sense in which a logistic regression classifier C^s_κ associated with a bigram κ in a specific residual stream depth s has an associated feature direction, namely the weight vector $\mathbf{w}^s_\kappa \in \mathbb{R}^d$. Rewriting the weight vector as a scalar multiplied by the L2-normalized weight vector $\mathbf{w}^s_\kappa = c^s_\kappa \mathbf{u}^s_\kappa$ and the input

$$C_\theta(\mathbf{x}) = \sigma(c^s_\kappa \mathbf{u}^s_\kappa \bullet \mathbf{x} + b^s_\kappa)$$

we see that the classifier is ultimately dependent on the cosine-similarity between the input vector $\mathbf{x}$ and the "feature direction" $\mathbf{u}^s_\kappa$.

Given that all the feature vectors cannot be orthogonal, 'activation' of one feature by having a high dot-product with the feature direction can result in the activation of unrelated features, which would ideally have orthogonal feature directions but do not because of superposition (see Appendix B.3). In [16], a notion of 'feature dimensionality' is introduced that tries to quantify some aspects of such 'interference.' The paper primarily investigates how sparsity (what we call feature frequency) influences feature dimensionality. As we will discuss, two feature types with the same feature frequency

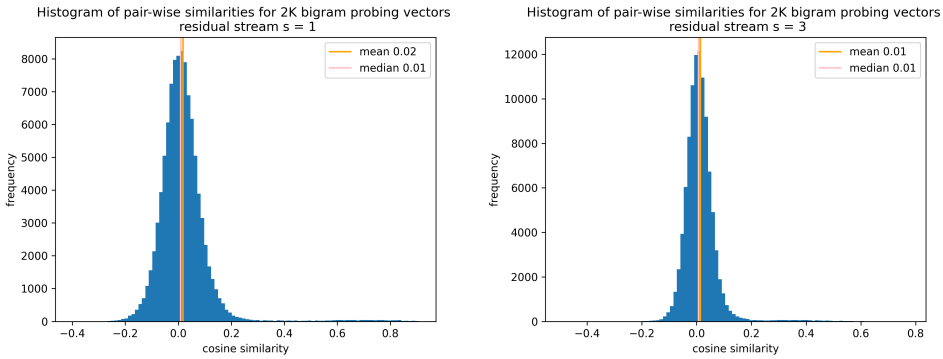
could have different levels of importance to prediction, and we hypothesize that this importance is associated with the feature having a larger part of the representation space allocated to it, in the sense of higher feature dimensionality.

It is not clear exactly how to think about and quantify this importance, and we will present an idea for quantifying it. Furthermore, there are obstacles to applying the concept of feature dimensionality to our use case. We will try to address these obstacles and test the hypothesis that feature dimensionality is related to importance. Let us first, however, focus on a related but simpler aspect of how such feature directions are distributed.

3.3.1 Isotropy of Feature Directions

A general idea underlying the proposal of superposition in [16] is that superposition can be viewed as a model "simulating a larger network" with spaces of higher dimensionality where directions would be orthogonal. This leads to the view of feature directions as being distributed as widely as possible, so that, for example, 5 feature directions in 2-dimensional space would form a pentagon, or the feature directions are "isotropically distributed".

By computing the pairwise cosine similarities between bigram feature vectors belonging to $s = 2$, we find that the mean and median cosine similarity is quite close to 0. This suggests that the feature vectors are quite "spread out", and we find a relatively heavy positive tail where feature vectors are quite close to each other. These are in almost all cases what we call embedding-similar bigrams in 3.2.4.



(a) depth $s = 1$ **(b)** depth $s = 3$

Figure 3.2: Cosine-similarities between pairs of bigram feature vectors for Pythia-410m in residual streams at depth $s = 1$ and $s = 3$

Given the idea that the first MSA module would transfer token embedding vectors using a linear OV-circuit [18] and the general idea of embedding-

similar bigrams, the fact that feature vectors group together is perhaps not surprising. We think it is worth reflecting on whether isotropy is an inherent part of the idea of superposition. On the view that it is, one could either view the observations of the non-isotropic distributions of these bigram feature vectors as providing evidence against the theory, or one could argue in various ways that either this is why bigrams themselves should not be considered features, or that the feature directions used are too noisy due to details in probing. Another argument that it is not problematic is that the isotropy is primarily studied and observed in [16] in a context where the importance is the same for all bigram features.

In concurrent work [10] that uses ”dictionary learning” to identify feature directions in one-layer Transformer language models, the identified feature directions are not isotropically distributed but form rough clusters, and it is said that this could be viewed as ”anisotropic superposition”. Anisotropy, especially when there are clusters, would imply that feature spaces are not ”almost orthogonal”, so the question seems to be: Is almost-orthogonality central to the idea of superposition or is it merely a narrower version of the idea?

These observations and reflections, together with the discussion about the applicability of discrete classification metrics in 3.2.3, suggest to us that a theory of superposition based on representational realism may not be falsifiable. It does not seem that there is any direct way of determining whether a group of feature directions for a proposed set of feature types should be viewed as a simple feature, or what [10] calls a feature manifold, and that the core of the problem is the ill-defined nature of many unsupervised learning problems, including clustering. It is hard to see how one can falsify a theory of true native feature directions if the computational problem of identifying them might be ill-defined. We emphasize that these are only reflections and are not meant as definite conclusions, and also should not have implications for less realist views like forms of representational pragmatism that acknowledge that viewing spaces as having many features in linear subspaces and understanding accessibility, usage, and more using linear methods can be useful in many contexts.

3.3.2 Feature Dimensionality

In a d-dimensional space d one-dimensional feature types could have their own associated basis vector as feature directions. This would make the dot-product and cosine-similarity between feature directions 0. In other words, given a total ”capacity” of d dimensions, each feature type gets a capacity of 1 dimension. As discussed, this is not so when there are more feature types and associated feature directions than dimensions, and [16] introduced a concept of *feature dimensionality* where the intuition is supposed to be that

a specific degree of the total capacity is allocated to the associated feature type.

Given a set of feature vectors $S = w_1, ..., w_n$ that are not necessarily unit vectors, the feature dimensionality for w_i is defined as

$$D_i = \frac{||w_i||^2}{\sum_j (\hat{w}_i^T w_j)^2}$$

where $\hat{w}_i$ is w_i normalized.

The idea is that if there are many w_j that project onto w_i the more w_i will have to share its subspace with others.

One aspect that is not discussed in detail in [16] is the initial assumption of orthogonality and near-orthogonality. While [16] investigates *correlated features* and how they could influence feature dimensionalities, there could be other reasons why features should not ideally be represented as orthogonally as possible, even if they are not correlated, such as the ease of downstream accessibility and use. Bigram feature types are, by construction, non-positively correlated because only one of them can be present at the same time. However, as seen, for example, in the concept of embedding-similarity, they are still not necessarily as orthogonal as possible. Concurrent work [10] considers making a distinction between isotropic and anisotropic superposition.

There are two main challenges in applying the concept of feature dimensionality to our logistic regression feature directions:

First, it is introduced in a context of toy models where one has very fine control of the features, so that S is, in a meaningful sense, the *totality* of all features in the data, the complete set of feature directions. While it should, in principle, be possible to estimate feature directions for the complete set of possible bigrams, this is not feasible. Even if it were, from the representational realism perspective, the set of bigram-features does not constitute the totality of features that would be encoded in this space. Alternatively, one could say that the definition does not align well with our framing of the feature type set as a set that is relative to the investigation one is performing. The main reason why this is problematic is that increasing the size of S to include more feature vectors will increase the denominator, and the feature dimensionality of most features will decrease. Thus, it is not clear if feature dimensionality makes sense in the case where one cannot claim that S is the totality of features.

The second challenge is that the definition of feature dimensionality distinguishes between the direction and norm of the feature vectors in S. Our feature directions, attained by logistic regression, have various norms, and

while these norms are not void of information, it is not clear that the norms are directly useful in the context of feature dimensionality.

We try to deal with the first problem of the dependence of feature dimensionality values on the size of S by limiting ourselves to the *ranking* of feature dimensionalities: Consider a random subset of $P \subset S$ of size k. Computing feature dimensionalities $d_1, ..., d_k$ by using only P, and feature dimensionalities $d'_1, ..., d'_k$ using all of S, we find that the Spearman correlation between the two is very high (> 0.99), even though $d'_1, ..., d'_k$ are much lower in general because of the greater denominators involved. From this observation, we believe that the ordering and comparison of feature dimensionalities might still be meaningful to consider, even if we have this problem.

For the second problem, there is always the option to leave it open and compute feature dimensionalities using both normalized and non-normalized bigram feature directions: Since the intention behind the logistic probing was primarily to find the direction, intuitively we favor the approach of using normalized versions. In the unnormalized case, we have:

$$D_i = \frac{||w_i||^2}{\sum_j (\hat{w}_i^T w_j)^2} = \frac{1}{\sum_j (\hat{w}_i^T \hat{w}_j)^2} = [\sum_j sim(w_i, w_j)^2]^{-1}$$

which is the inverse of the sum of squared cosine similarities. Since we are considering only rank, this means that intuitively feature dimensionality is a matter of having few close neighbors, or about 'how crowded the area of space is', or 'how distinguishable the feature is from other features'.

3.3.3 Determinants of Feature Dimensionality

What might determine the feature dimensionality of a feature type? In [16], the authors investigate how sparsity (feature frequency), correlation, and the importance of the feature are involved as factors. The setup in [16], where importance is a specific parameter associated with a feature, does not easily transfer to our setup, however.

Intuitively, if a feature is very frequent but unimportant in the sense of having no relevance at all for the loss, then we might not expect the feature's accessibility to be high, and there is no reason why the representation should favor a high feature dimensionality. One could also imagine a binary feature that is not very frequent but is very important in the sense that, in the case where it is present in a prefix $x_{\leq t}$ of the input, it has significant implications for attaining low per-token losses on the following tokens $x_{>t}$. Thus, one might suspect that feature dimensionality is determined by the frequency, importance, and the relationship between the two.

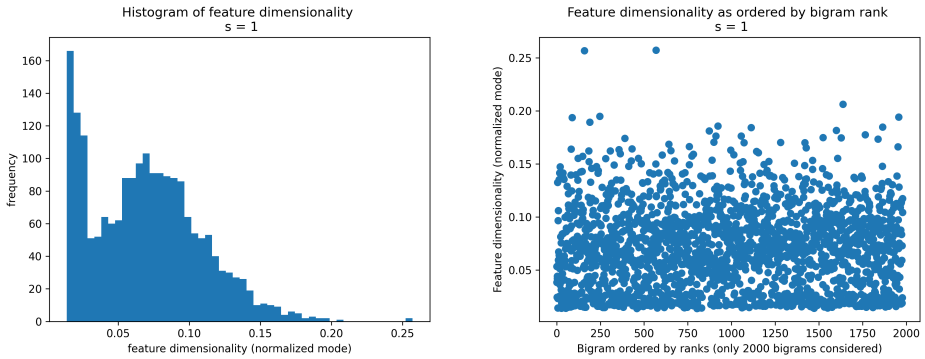
However, other aspects might also be involved: If two features are in some sense similar and have very similar implications in terms of optimal model

behavior, one might hypothesize that their feature directions are similar. Again, we don't see that this similarity should necessarily involve correlation. It might be objected that it seems very unlikely that two features are very similar without being correlated, or that if so, then it would be better to view them as a single feature.

Lastly, one would ultimately expect feature dimensionality and linear accessibility to be quite related. If a feature type has its own allocated dimension, such that no other feature vector has similarity with it, and its feature dimensionality in the unnormalized case is 1, then one should be able to get a very effective linear classifier. Likewise, if the feature dimensionality is very low and many other feature directions are in the neighborhood, then this suggests that it is hard to distinguish and false positives would be one of the problems, giving lower linear accessibility.

3.3.4 Feature Dimensionality of Bigram Feature Vectors

We compute feature dimensionality values using the feature vectors identified in Section 3.2.3. Though we have seen in Section 3.2.4 that using "half-adversarial probing dataset" one can increase the linear accessibility for some bigrams which would lead change in the feature dimensionality, we do not have probes and the associated weight vectors for many bigrams trained using such a setup (one could consider training only on bigrams that achieve low AUC-PR score). It seems that the fact that some bigrams get low probe performance at least in some cases is quite closely related to the fact that its probing vector has many nearby probing vectors as neighbors and therefore we should still be able to get some insights even if using better probes would be advantageous.



(a) **(b)**

Figure 3.3: Feature dimensionality for approximately 2 thousand bigram feature types based on feature vectors in residual stream depth $s = 1$

In Figure 3.3a, we show the computed feature dimensionalities based on 1980 normalized weight vectors from the probes trained on the residual stream $s = 1$. We note again that the absolute values are not necessarily so meaningful because they involve only a small subset of all the feature types. We find the two outliers with the highest feature dimensionality to be the bigrams (_id,=") and (_class,="), which both occur in HTML, and other bigrams associated with HTML such as (td,>), (script,>), and (string,>) also have high feature dimensionality.

The bigrams with the lowest feature dimensionality are bigrams of the type comma followed by words such as "all", "one", "in", "like", "both", called *stopwords*. One hypothesis is that these stopwords are quite similar in terms of their embeddings, and that this similarity carries over to the feature vectors learned by the probes. However, this explanation seems to be false, as the stopwords are not particularly close. Alternatively, it might be related to the comma "," having an embedding whose norm is very small; out of 50K types, it has the 280th lowest norm. Newline has the 263rd lowest norm and is also strongly represented among bigrams with low feature dimensionalities. It is not clear how exactly this would give rise to the feature dimensionalities observed. The tokens have low norms too, in the lowest 2-3% of token embedding norms.[5] However, looking at the norms of the type embeddings in order to understand the feature dimensionalities could be problematic because of the *attention-MLP parallelism* in Pythia models: The MLP in block 0 takes only type-embeddings as input, and therefore the output of this MLP contributes significantly to the representations of individual tokens.

In Figure 3.3b we plot the feature dimensionality according to the feature frequency rank of the bigram feature types: The lowest ranked bigrams are the most frequent. Given that the rank is directly related to the feature frequency, though we did not explicitly hypothesize it, it is surprising that the two are not visibly correlated.

3.3.5 Bigram Feature Importance

As mentioned, one would expect importance of a feature to be a determining factor in its feature dimensionality. In [16], because the authors are investigating a simulated setup, the importance can be specified directly as a coefficient in a loss term. As stated above, we think that intuitively a binary feature type z is important, if for a sequence x, if the prefix sequence $x_{\leq t}$ has the feature, meaning $z(x_{\leq t}) = 1$, then it is useful to use this feature for attaining low per-token losses on the following tokens $x_{>t}$. While we

[5] Specifically, the stopwords have norms of 0.7528, 0.7847, 0.5662, 0.7626, 0.8043, 0.5608 respectively. The proportion of types that have embeddings with a norm less than .8 is less than 2%.

investigated ideas for defining the counterfactual where a bigram feature was not present and looked at the change in loss for the remaining tokens in the sequence under this counterfactual, this proved hard to do and computationally expensive and we present another idea.

As mentioned in 2.2, the n-gram statistics of text imply that bigram features have some value for predicting the next token. As mentioned we can imagine bigram features that are not directly predictive of the next-token but still have importance for prediction. Focusing on only the next-token prediction for measuring importance instead of all later predictions is thus a limitation but more easily accessible.

Given that both the feature frequency and the feature importance ought to be important to the degree to which a model will model a feature there is also the question of the relationship between the two. We propose a notion of *bigram advantage* as a scalar function of a bigram $L : \mathcal{V}^2 \to \mathbb{R}$. For a bigram $(a, b) \in \mathcal{V}^2$, the advantage $L(a, b)$ is measured with respect to a given model, in our case Pythia-410m.

$$L(a, b) = p(ab) \sum_c p(c|ab) \log \frac{p(c|ab)}{p(c|b)}$$

This bigram advantage is thought of as a form of product of the feature frequency and the feature importance. We emphasize that we think this idea of advantage only captures importance of a bigram feature to a very limited degree but we will now introduce the motivation behind the definition of $L(a, b)$.

Consider one of the simplest model of a sequence data distribution which would be a unigram model $q(v)$. This model has a vector of $O(V)$ parameters specifying the probability of occurence of each token. With respect to the data, this model has a certain cross-entropy. Another step of complexity for a model is $q(b|a)$. This model is parameterized by a matrix of $O(V^2)$ parameters specifying the probability of one token following another. Assuming we are not care about the predicting the first token in a sequence, this model has a certain cross-entropy. In the language of features, the model could perhaps be said to operates on the features "the input sequence ends with token a", i.e. "unigram features types".

Assume now that we have an optimal such model where $q(b|a) = p(b|a)$ where p is the data distribution. Assume we are afforded V *additional parameters* in this sense: We can select a single bigram AB, and use the V parameters to model $q(c|a = A, b = B)$ and will be given access to ground truth probability mass function $p(c|a = A, b = B)$ specifying the trigram statistics for AB, so that $q(c|a = A, b = B) = p(c|a = A, b = B)$ for exactly the bigram AB, but in all other cases we make our predictions using the

bigram statistics. In the case we see a sequence ending with AB, we will use $p(c|a = A, b = B)$ for predicting instead of our previous option $p(c|b)$ attained from the $O(V^2)$ matrix. If the goal is to minimize cross-entropy, which bigram AB should we select? The answer to this question is in a narrow sense a possible answer to the question of which bigram feature a model should care more about modelling.

Assume that the model wants to maximize the negative cross-entropy $-H(p, q)$. Because the model cares about no more than trigrams, we can evaluate the cross entropy using the data distribution over trigrams, and write $(abc) \sim p$.[6]

$$\mathbb{E}_{(abc)\sim p}[\log q(c|a, b)] = \sum_{abc} p(abc) \log q(c|a, b) =$$

$$\sum_{ab} p(ab) \sum_{c} p(c|ab) \log q(c|a, b) =$$

Because the model $q(c|a, b)$ will only use both of the preceding tokens for the single bigram we have selected, call it $a'b'$, we can write

$$\left[\sum_{ab} p(ab) \sum_{c} p(c|ab) \log q(c|b)\right] - \left[p(a'b') \sum_{c} p(c|a'b') \log q(c|b')\right] +$$

$$\left[p(a'b') \sum_{c} p(c|a'b') \log q(c|a'b')\right] =$$

Defining $C = \sum_{ab} p(ab) \sum_{c} p(c|ab) \log q(c|b)$ as a constant that is independent of our choice $a'b'$ we can write it as

$$C + p(a'b') \sum_{c} \left[p(c|a'b') \log q(c|a'b') - p(c|a'b') \log q(c|b)\right] =$$

$$C + p(a'b') \sum_{c} p(c|a'b') \log \frac{q(c|a'b')}{q(c|b')} = C + L(a', b')$$

This leads us to the result that the bigram a'b' to be selected in the setup and assumptions should be the one that has the greatest bigram advantage.

This advantage consists of a product of two factors, the first is the probability that a'b' occurs, the feature frequency, the second is the expected log-fold increase with respect to all the continuations c. We call the second factor the unweighted advantage, and the product the weighted advantage.

[6] We can either assume the model is always used to predict tokens given at least a context of two tokens or that it also model the two initial tokens in a sequence, but in any case this should not make a difference in this context.

There are two ways we might try to evaluate $L(a'b')$ in practice, for a specific bigram $a'b'$. We can make a estimate the pmf $p(c|a'b')$ from data, by collecting instances $a'b'c$ for any c, we can, likewise by collecting instances $b'c$ for any c we can estimate the pmf $p(c|b')$ and lastly, as we did in the beginning of the chapter, collect instances of ab for any a and b to estimate $p(a'b')$.

Alternatively we suggest a *model-based* approach for the unweighted advantage: We can use a Transformer language model to compute $p(c|a'b')$ and $p(c|b')$ by forwarding the short sequences (a',b') and (b') through the model and collecting the predictive probabilities. Behind such a choice would be an assumption that for sufficiently trained Transformer language models are sufficiently calibrated models of the bigram and trigram statistics of the data. This is an assumption that might be problematic, and we highlight it as a limitation to a proposed metric that is already has its limitations. As is also mentioned later, we find that unweighted advantages computed using Pythia-410m and Pythia-1b are strongly correlated which could either indicate that they are indeed calibrated to the bigram and trigrams statistics of the data or that they are biased in a similar way.

3.3.6 Hypothesis

Informal hypothesis: In early residual stream layers the feature dimensionality of bigrams features is correlated with the importance of the features.

Specific hypothesis: In residual stream depths $s = 1$ and $s = 2$ of Pythia-410m there is a statistically significant positive Spearman rank correlation between the the feature dimensionalities computed based on the 1980 candidate feature vectors computed in the previous section of the weighted advantage of the bigram feature type.

3.3.7 Method

The methods of feature dimensionality and advantage is described in the above sections.

We choose the same 1980 bigrams considered in the previous experiments. For a depth s, we consider the feature direction for a specific bigram feature type z_κ to be the weight vector $\mathbf{w}_\kappa^s$ associated with the classifier C_κ^s, where the weight vector is scaled to be unit norm For the depth s the associated feature dimensionalities by using the feature directions at that depth s.

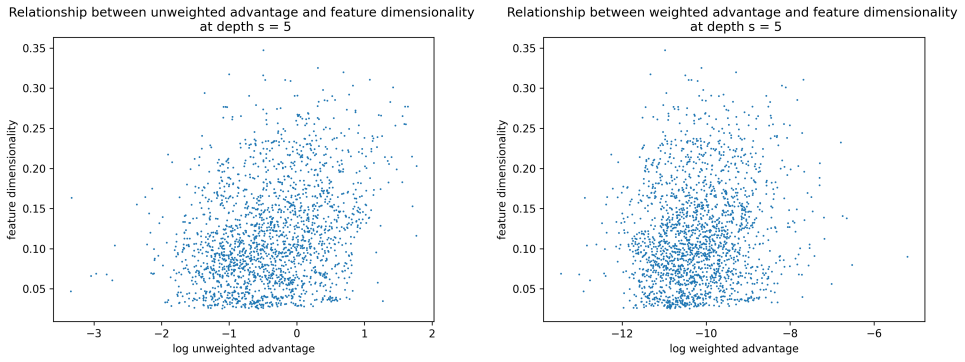
Separately for each of the 1980 bigrams we compute the unweighted advantage using the Pythia-410m. We find that the unweighted advantages computed using Pythia-1b correlate very strongly with those attained for Pythia-410m. For weighted advantage we use the empirical feature frequencies derived from occurence counts in T77K (see 2.2.2)

As mentioned we believe that only the ranking of feature dimensionality is meaningful when we cannot be said to be considering the totality of feature types, and because bigram advantage is also primarily meaningful in terms of ranking, for a depth s we compute the Spearman Rank Correlation between the unweighted advantages and the feature dimensionalities.

3.3.8 Result

For unweighted advantage which is supposed to be our metric for importance, we find that at all depths considered $s \in \{1, 2, 3, 4, 5, 6, 8, 11, 14\}$ the Spearman Rank Correlation is between 0.214 and 0.285 with the highest correlation being at depth $s = 5$ (p-values $< 10^{-20}$)

For weighted advantage which is supposed to be a our metric for the product of feature frequency and feature importance, we find the Spearman Rank Correlation to be the lowest at $s = 2$ where it is 0.087 and highest at $s = 5$ where it is 0.159 (p values $< 10^{-5}$). In Figure 3.4 we show the relationship for both weighted and unweighted advantage. Since weighted advantage is dominated by the the strong differences in feature frequencies described in Section 2.2.2 and we saw that the feature dimensionality seemed to be relatively independent of the rank of the bigram it might not be so surprising that the association between weighted advantage and feature dimensionality is not so strong.



(a) **(b)**

Figure 3.4: Association between feature dimensionality and weighted and unweighted advantage respectively, for $s = 5$ where the effects are the largest among depths considered.

While the effect is stronger for the association between feature dimensionality and bigram advantage, the association is still weak. If there is indeed a pattern here, there are multiple sources of noise and limitations to the experiment we conducted: The feature vectors are limited by aspects of the probing performed, including the construction of the probing datasets and

details about hyperparameters. Secondly, we are normalizing the feature vectors which might be relevant for better idea of the feature dimensionality. Thirdly, we are using a metric of importance that has many limitations described above, including that the importance of many bigrams is not best viewed as their contribution to prediction of the next token in the sense entailed by the definition of bigram advantage.

Lastly, echoing discussion in Sections 3.2.3 and 3.3.1: Feature vectors that are grouping together will have reduced feature dimensionality but this does not necessarily mean that they are not important, maybe the feature types are similar in some sense and should be treated as a group. Focusing on individual feature dimensionalities and importances is maybe mostly meaningful when features are really unrelated and separate aspects of the data.

Chapter 4

Feature Usage

In the previous chapter, we investigated feature accessibility as a way of understanding what features could, in principle, be used by a module that can only use linear information in the residual stream. We also found vectors associated with these features but could not necessarily conclude anything about whether these subspaces related to the model's usage of the feature. However, [35] [16] [25] quite explicitly think of features not only as being accessible but also as being "used" by modules such as MLPs. In this chapter, we will present ideas about what it could mean to say that a model linearly uses a feature and how to investigate it. Ideally, we want to move on from having validated the linear accessibility of a large number of bigram features packed in the spaces of hidden states to validate that these features are being "used" through linear encodings, which would lend support to the idea of superposition in Transformer language models.

Just like there is no clear consensus regarding the notion of features, there is no consensus on the idea of what it would mean for a model to use such features. From the line of work that pursues an understanding of models as having circuits [35], models are understood to represent features, for example, in linear subspaces, and parts of the model, such as neurons, will then perform understandable computations on these features. On this view, in a vision model, a binary car feature is "detected" by what amounts to an almost boolean logical computation on lower-level "window", "wheels", and "car body" features. From this perspective, the most intuitive approach to investigating feature usage is to find lower-level and higher-level features and their encodings and study the relationship between them. It is a relatively explicit goal of work like [16] [10] to understand superposition and identify feature directions in order to proceed to understand such circuits.

A variety of relatively different lines of work seem to consider usage to be about causality: Causal interventions that modify hidden states and encodings of features should have various effects on the following hidden states

and the outputs of the model [3] [6] [28] [41] [20] [32]. For example, [28] proposes a method for evaluating explanations of neurons by constructing tasks associated with the feature and assessing how causal interventions on the neuron affect the model's ability to solve the task. Relatedly, [20] and [46] advance a method where a model's ability to solve a task is explained by an alignment of the hidden states with a causal model proposed by the researcher.

As argued in [6], a necessary condition for a model M using a feature type z is that its output, and therefore its inputs and hidden states, should have significant mutual information with $z(X)$.[1] Following the language of residual networks and linear accessibility we introduced in 3.1, we might say that a necessary condition for a residual block f_s operating on $h_s(X)$ to linearly use $z(X)$ is that z is linearly accessible at depth s. For the model more broadly, a necessary condition for a residual model to linearly use z is that z is linearly accessible at some depth s.

The trigram statistics of the data mean that certain bigrams AB have certain associated tokens C that are likely continuations. This means that one way of investigating the causal aspect of linear encodings is to perform causal interventions and assess the effect on the model's assignment of probability mass to such likely continuations. This has various limitations: It is not a priori clear what the relationship between bigram feature direction and likelihood of C should be. Even if it is most natural to expect that increasing the "activation" of a feature should increase it and "deactivation" decrease it, one would expect that many kinds of features, bigrams included, have a more complex relationship to model outputs. Even if there was a clear relationship for bigram feature types, it does not easily transfer to investigations of more complex features. Despite such limitations, we will attempt investigations in this direction below.

We also propose that there could be ways to understand some aspects feature representations and their involvement in the computation of the forward pass without studying relationships between lower-level and higher-level features or their relationship to specific model outputs: While the simplest interpretation of the circuits view suggest that features are either linearly accessible or not and that feature detectors are localized to specific modules, some lines of work could be viewed as suggesting a different picture where modules act to gradually increase linear accessibility of features and that the linear accessibility of higher-level features gradually increases through circuitry that is spread out across modules [31] [5] [21] [47]. From this perspective a starting point would be to understand the

[1] This is with our terminology, in [6] "concepts" are viewed as random variables for which there exists a joint distribution over inputs and concepts but need not be deterministic functions of the input.

non-interventional/observational change in linearly accessibility of features across depths s and use causal interventions to understand the way blocks or modules are causally involved in changes in linear accessibility. Here our main proposals is to use concept erasure to intervene so as to reduce linear accessibility in one hidden state and investigate its effect on downstream linear accessibility.

4.1 Methods

4.1.1 $\mathcal{V}$-information and Least Squares Concept Erasure

[47] presents an extension of information theory under computational constraints or "usable information". We briefly present the idea here and refer to the paper for formal details. Consider two random variables X and Y with joint distribution $P_{X,Y}$. Let $\mathcal{V}$ be a a class of predictive models $q(y|x)$ for predicting Y using "side information" X or by ignoring the side-information by having a predictive distribution $q(y)$. In our context of binary feature types $\mathcal{V}$ should be thought of as the family of logistic regression classifiers parameterized by the weight vector and bias scalar, and ignoring the side-information should be thought of as the logistic regression classifier having a zero vector as weight vector so that the prediction is only based on the constant b.

They define the conditional $\mathcal{V}$-entropy as the minimal achievable cross-entropy using a model $q(y|x)$ from the family $\mathcal{V}$ so that

$$H_{\mathcal{V}}(Y|X) = \inf_{q \in \mathcal{V}} \mathbb{E}_{P_{X,Y}}[-\log q(y|x)]$$

This can roughly be thought of as the binary cross entropy for a classifier from the previous chapter evaluated on the interpretation distribution, though we do not have any guarantees since the probes are trained on another distribution. They define the $\mathcal{V}$-entropy as conditional $\mathcal{V}$-entropy but with no conditioning on side-information X in the sense that $q \in \mathcal{V}$ here is restricted to be a constant classifier

$$H_{\mathcal{V}}(Y) = \inf_{q_{const.} \in \mathcal{V}} \mathbb{E}_{P(X,Y)}[-\log q_{const.}(y)]$$

In our context of binary feature types, this can be thought of as the on-distribution entropy of $z(X)$. Analagously to mutual information, the $\mathcal{V}$-information from X to Y is the reduction in V-entropy when conditioning on X.

$$I_{\mathcal{V}}(X \to Y) = H_{\mathcal{V}}(Y) - H_{\mathcal{V}}(Y|X)$$

When talking about the linear information about a feature type z in hidden state $h_s(X)$ of a model we mean $I_{\mathcal{V}}(h_s(X) \to z(X))$. It would also be useful to view z being linearly accessible at s to mean that $I_{\mathcal{V}}(h_s(X) \to z(X)) > 0$.

For such a family $\mathcal{V}$ of predictors we say that X guards Y, if $I_{\mathcal{V}}(X \to Y)$ is 0 or very small. [40] We say that X linearly guards Y when $\mathcal{V}$ is a a family of linear predictors such as our logistic regression family.

LEACE [6] is a method for *concept erasure* based on the concept of linear such guardedness. A LEACE eraser $r : \mathbb{R}^d \to \mathbb{R}^d$ is an affine function whose goal is to ensure that $r(X)$ linearly guards Y. This means that $I_{\mathcal{V}}(X \to Y)$ is not necessarily zero but $I_{\mathcal{V}}(r(X) \to Y) = 0$. In our context it means that applying a specific eraser Eraser_s to the hidden state $h_s(X)$ could make $I_{\mathcal{V}}(r(h_s(X)) \to z(X)) = 0$ which would mean that a logistic regression probe for a bigram feature type would perform no better than a constant predictor based only on the bigram feature frequency.

The LEACE eraser is both an affine transformation and the erasure is also trying to limit the "damage" to X by having the expected norm of the change $\mathbb{E}||X - r(X)||$ be minimal.

LEACE should allow us to study feature types, their linear accessibility and how the action of modules relate to this linear accessibility by allowing for a removal or reduction of the linear accessibility from a hidden state.

4.1.2 Causal Interventions

An ablation is a causal intervention where a possibly stochastic *ablation function* $\text{Ablate} : \mathbb{R}^d \to \mathbb{R}^d$ is used to alter a hidden state. For a residual network, an ablation of the residual stream $h_s(X)$ using such an ablation function we denote by

$$h'_s(X) \leftarrow \text{Ablate}(h_s(X))$$

where all downstream processing will now use this new value.[2]

Let Eraser_s^z be the LEACE eraser that makes $\text{Eraser}_s^z(h_s(X))$ linearly guard $z(X)$. Then by erasure of z at s we mean

$$h'_s(X) \leftarrow \text{Eraser}_s^z(h_s(X))$$

The idea of intervening on representations by adding a vector that is supposed to be related to some concept is a simple idea and so is not especially novel, see for example concurrent work [48] and [45]. Intervening by assing a vector v to a hidden state $h_s(X)$ we denote by

$$h'_s(X) \leftarrow h_s(X) + v$$

In directional activation intervention, inspired by the idea of directional activation patching [44], we consider a vector $\mathbf{v}$ and modify the hidden state

[2] Ablations can be performed for a variety of Transformer language models, including Pythia models, using `TransformerLens` hooks functionality. [34]

$h_s(X)$ to have a certain dot product t with $\mathbf{v}$

$$h'_s(X) \leftarrow h_s(X) + (t - h_s(X) \bullet \mathbf{v})\mathbf{v}$$

4.2 Effect of Addition Interventions on Model Output

In the case where we have a specific binary feature type z and believe to be in possession of a *feature direction* $\mathbf{u}_z^s$ in a space such as the residual stream r_s at depth s, there are some cases where we expect the feature to have relatively direct relationship to the model prediction. In the case of bigram features we seek to investigate such relationships using an idea of *empirical next-tokens* and *addition interventions*.

4.2.1 Method

Assume we are dealing with a bigram feature type z_κ and the residual stream r^s at depth s. Then assume, we have a vector $\mathbf{u}_\kappa^s \in \mathbb{R}^d$, for example the logistic regression weight vector $\mathbf{w}_\kappa^s$ associated with the logistic regression classifier C_κ^s that we trained in Section 4.3.1. We could be interested in assessing whether the classifier not merely tells us something about the linear accessibility of z_κ at depth s but whether the direction $\mathbf{w}_\kappa^s$ learned by the classifier relates to the usage of z_κ. We will do so by performing an intervention on 2-negative validation sequences $x \in \mathcal{X}_{2neg}^{val}$:

$$r_T^s(x) \leftarrow r_T^s(x) + c\mathbf{u}_\kappa^s$$

where $\mathbf{u}_\kappa^s$ is a unit vector and c is then the norm of the added vector.

Consequent sets

Inspired by the idea in [33] that models learn discrete prediction rules we view a bigram AB as being the basis of a prediction rule connected to the trigrams statistics of the training data.

Consider a Transformer language model q, we use Pythia-410, the same model as we are investigating, but this need not necessarily be the case, a larger model like Pythia-1b which is also trained on The Pile oculd also be used.

As we did when we considered the bigram advantage in Section 3.3.5, we consider the bigram as a the short sequence (a,b), for which q can give us a probability mass function $q(c|a,b)$ over next tokens. Likewsie we consider (b) as a sequence of one token for which we likewise can get the probability mass functions $q(c|b)$.

We are interested in tokens C for which it holds that C is upregulated when preceeded by AB compared to just B.

$$\log \frac{q(c|a,b)}{q(c|b)} > \gamma$$

We use $\gamma = 2$. Intuitively we are discretizing the learned trigram statistics to view trigrams as specific prediction rules of the form AB $\rightarrow$ C.

We also require that C should be among the top $k = 30$ top tokens according to probability in $q(c|a,b)$. Since models are trained using empirical cross-entropy the details of assigning mass to next-tokens that are very unlikely to occur in the data means that we cannot expect the whole probability mass $q(c|a,b)$ to be calibrated with the whole data distribution. Therefore we might expect that unembedding space might have some structure that could result in tokens that have very low rank in $q(c|a,b)$ and $q(c|b)$ to be heavily upregulated but such tokens are not we are interested in.

For a bigram AB the set of tokens C that fulfill these requirements we call the "consequent set" for AB. With $\gamma = 2$ and $k = 30$, of the 1980 bigram feature types considered in Section 3.2.3, 1423 have non-empty consequent sets.

Empirical Next-Tokens

Another way of identifying suitable tokens C that are continuations of AB would be to rely only on the data set and not a model. Among the top ranking bigram we have 14K bigrams $C \subset K$ for which it is easy to construct probing datasets. We consider the top 1000 top ranking bigrams $B \subset C$ in this set, and consider a specific subset of bigrams $A \subset B$ for which we will try analyze the feature use.

For $\kappa \in A$ and its associated probing *training* dataset $\mathcal{S}_\kappa^{train}$ the following holds:

- The notion of a next-token for a sequence in the probing dataset is defined in Section 2.3. Let $S_{next}^{pos} = (v_0, v_1, v_2) \in \mathcal{V}^3$ be the 3 distinct types that are most common next-tokens in S^{pos}, with c_0, c_1, c_2 being the number of sequences in S^{pos} where they are the next-tokens. Let then the occurence frequencies $f_i = \frac{c_i}{|S^{pos}|}$ for $i \in [3]$
- Let $S_{next}^{2neg} = (u_0, ..., u_{k-1}) \in \mathcal{V}^k$ be the $k = 20$ types that are most common next-tokens in S^{pos}, and associated occurence frequencies g_j for $j \in [k]$
- At least one of the following holds:
  - $f_0 > 2 \cdot g_0$. i.e. the frequency f_0 of the top next-token S_{next}^{pos} is at least twice the frequency g_0 for the top next-token in S_{next}^{2neg}.

– v_0 is not among $u_0, ..., u_{k-1}$.

For $\kappa \in A$, we write $v_0(\kappa) \in \mathcal{V}$ for the top next-token in S^{pos} and call it the empirical next-token for κ. The idea is that $v_0(\kappa)$ is either "up-regulated" or "unique" as a top next-token compared to S_{2neg}.

We experimented with using these kinds of tokens but a problem with the approach is that the statistics of the data of which tokens are likely to *preceed* AB greatly influence what the empirical next-tokens is in ways that are undesirable because they are in a way next-tokens for more complex features.

Model Output

We will consider two aspects of the effect of the intervention on the model's output: The effect of the log-probability of the target token and the rank of the target token.

For an input sequence x of length T to the model q, we denote the predictive distribution for the next-token at position $T+1$ as $q(\dot{|}x)$ which is a probability mass function in the V-simplex Δ_V. For numerical reasons and for ease of assessing changes in probability we focus on the log-probability

$$\pi(x)_i = \log q(i|x) \in \mathbb{R}$$

for each $i \in [V]$. Given a subsequence x in the dataset where the *actual* next token is of type $i \in \mathcal{V}$, the log-probability $\pi(x)_i$ is the *negative of the cross-entropy loss for the token*.

Given $\pi(x) \in \mathbb{R}^V$, some types have higher log-probability than others, and we let the rank of entry i in $\pi(x)$ be denoted as $\rho(x)_i$[3]. If type $i \in \mathcal{V}$ which has the highest probability we will have $\pi(x)_i = 0$

Ablation Vectors Considered

For a bigram feature type z_κ and a depth s we consider different kinds of addition interventions, where we use different vectors. All of these vector will be normalized to have unit norm before use:

- `xmeanpos` The mean of the positives

$$\bar{x}_{pos} = \frac{1}{|\mathcal{X}_{pos}^{train}|} \sum_{x \in \mathcal{X}_{pos}^{train}} r_T^s(x)$$

- `logreg` The logistic regression weight vector $\mathbf{w}^s$ associated with the logistic regression classifier C_κ^s

[3]Though hardly relevant in practice, in case of ties resolve however you please

- `latvec` The first principal component of the difference between positives and 2-negatives: Take the 800 training positives, pair them with 800 of the 1600 2-negatives. From these 800 pairs, compute 800 difference vectors. Take the first principal component of these difference vectors.
- `latvec2` The first principal component of the difference between positives and negatives in general: Take the 800 training positives, pair them with 800 random negatives. Compute the 800 difference vectors. Take the first principal component of these difference vectors.
- `random` A random vector sampled from isotropic Gaussian.

4.2.2 Hypothesis

Informal hypothesis: Feature directions identified by probes have a causal influence on model behavior, so that adding a feature direction for bigram AB will result in an increased probability assigned to tokens C that are likely to follow AB in the training data.

Specific hypothesis: For a bigram AB, associated 2-negative sequences from the probing dataset ending with B but not AB, intervening with addition ablation in $r^s_T(x)$ using the associated feature direction from logistic regression probes, results in an increase in mean log-probability and a decrease in median rank for tokens from AB's consequent set, when aggregating over the 2-negative sequences.

4.2.3 Experiment setup

The experiment is conducted on Pythia-410m. Due to resource limitations, we consider only the first 93 bigrams for which we have the feature directions. We use $s = 2$, a choice which could be justified by reference to the rank-performance curve from Section 3.2.3, but given more resources, we would be interested in other depths. For each such bigram AB, we take an arbitrary C from its consequent set and perform an addition intervention with an addition norm of 12.75. The experiment is set up so that we mostly just care that the intervention is strong enough to see an effect, and thus the exact details of the norm should not matter. However, some investigations indicate that very large norms are not desirable when conducting such experiments, possibly because we are moving off-distribution.

For each bigram AB, we perform the intervention $r^s_T(x) \leftarrow r^s_T(x) + c\mathbf{u}^s_\kappa$ for all 200 sequences $x \in S^{val}_{2neg}$.

4.2.4 Results and Discussion

For the 93 first bigrams, we find that

- In 0.94 of cases mean log-probability assigned to C is higher under addition intervention than under no intervention.
- In 0.875 of cases median rank of C is lower under additiona intervention than under no intervention.
- In 0.8472 of cases mean log-probability assigned to C is higher under addition intervention with feature vector than addition intervention with a random vector of same norm.
- In 0.875 of cases median rank of C is lower under addition intervention with feature vector than addition intervention with a random vector of same norm.

By inspection, we notice that it is especially in cases where one would not naturally regard C as a good continuation of AB where these are not the case, and vice versa, that when C seems like a natural continuation, it is the case.[4]

Because of the noise associated with such interventions, we have opted for an approach where we are only comparing and not measuring the magnitude. This also means that it is not straightforward to conduct a statistical test. However, we consider these numbers sufficient to convince us that the directions have at least moderate causal influence.

A major limitation of this approach is that it is hard to quantify the causality we can attribute to the feature vector. In works like [28], the framework is such that it is possible to quantify how much of the performance of a model can be explained by a specific neuron or direction, which is more appealing even if it is not obvious how to apply such a setup to our use case. If the feature vector considered is similar to another feature vector which is better in terms of accessibility, usage, or both, then we might see some of these effects.

For example, as shown in Figure 4.1, additional intervention with the simple vector $\bar{x}_{pos}$ gives an increase in log-probability of C just as often, and the logistic regression weight vector $\mathbf{w}_\kappa^s$ performs better than $\bar{x}_{pos}$ in terms of log-probability and rank only in 0.59 of cases, not much better than random chance. From a classifier, from the limited investigations we have made, such vectors will not have good performance in terms of recall and precision if used in a logistic regression classifier, but this does not mean that their directions cannot be important in various causal experiments. It does not seem unreasonable to believe that for a feature type, different ways of evaluating causality and usage have different optimal vectors associated with

[4]Cases where the continuations seem natural involve tokens that are natural words, which is not always the case, for example, parts of code and symbols like newline.

them. Even if a feature type has high linear accessibility, this level of accessibility and the associated subspace might be much stronger than the "signal" needed for different modules to use the feature type.

It seems that there is a difference between these two ideas one could have about superposition:

One idea could be that more than d features are packed in a d-dimensional space by non-orthogonal one-dimensional spaces in which linear information is stored and used by the model, and these spaces are monosemantic, which informally means that these spaces are "the only space used to represent and do computation involving the feature". Using the language of [24], monosemanticity could potentially be defined using the definitions of erasure, encapsulation, stability, and containment, coupled with additional ideas about feature usage. Such a perspective might be denoted as "determinate" superposition because it associates with features specific spaces that have special privilege in terms of representation and usage. It seems that determinate superposition would generally be connected to a representational realism perspective.

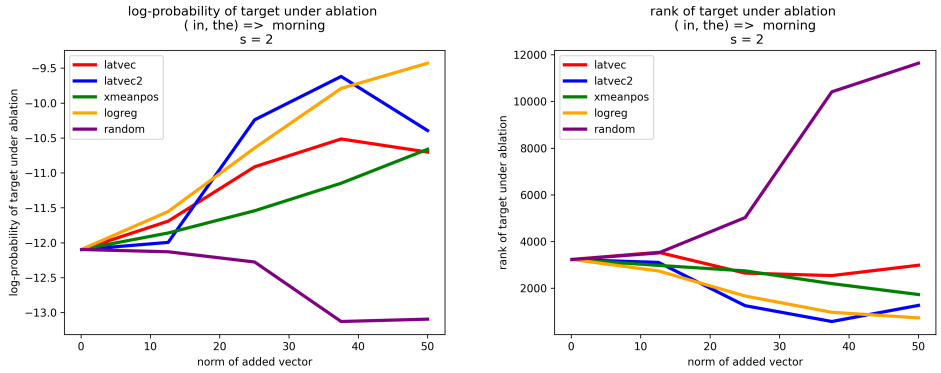
Alternatively, there might be a kind of "indeterminate" superposition, where linear accessibility and usage is more decoupled: Maybe, while there is some space that is more monosemantic in terms of accessibility, some other space or even multiple spaces that are not really monosemantic are being used in various ways because they still contain linear information about the feature. That is seems like a weaker claim about superposition.

Feature Vector	**Log-prob. increases vs no interv.**	**Median Rank decreases vs no interv.**	**Log-prob. inc. vs. random interv.**	**Rank dec. vs. random interv.**
`logreg`	0.94	0.88	0.85	0.8
`latvec`	0.96	0.61	0.72	0.54
`latvec2`	0.85	0.68	0.58	0.69
`xmeanpos`	0.94	0.86	0.77	0.87

Table 4.1: Among 93 bigrams considered, we consider in how many cases the addition intervention results the change in log-probability and rank changes in the hypothesized direction. We do this for 4 different types of candidates of feature vectors.

4.3 Effect of Erasure on Downstream Accessibility

The discussion at the end of the previous section adds to the motivation to use LEACE to study the linear representations of feature types. It seems that one of the problems with the approach above, which could also be a problem



(a) **(b)**

Figure 4.1: An example with the bigram (in, the) and the token " morning" from its consequent set. In this plots the x-axis is the norm of the added vector, which in the above experiment is fixed to $c = 12.5$. We see that as we add the associated logistic regression vector the probability assigned to " morning" increases and the rank decreases.

for similar approaches, is that it divides the problem into two: Identifying a feature direction and evaluating the causal influence of the feature direction. If the goal is to identify feature directions and validate their causal influence on prediction in order to break down a model into pieces, it might be that this is a good approach.

While internally LEACE will perform some form of projection operation, and in this sense associates a feature type with a specific subspace, the overall goal is rather to remove the linear information. One option is to study a variant of the previous experiment where instead of studying how additional intervention increases the probability for specific tokens, removal reduces the probability for specific tokens. While this is worthwhile, the next-token approach has certain limitations for bigram features and might not generalize well to future studies of other kinds of features, and the approach does not give us much insight into the usage of the features by modules, whether in the sense of circuits or some other kind of usage.

In the simplest interpretation of circuits, low-level features are either present in a space or not and form the basis for the detection of a higher-level feature that is localized in the sense of belonging to a specific module, such as an MLP module. Given that the linear accessibility of features can vary at different depths, it might be useful to consider a perspective where linear information about lower-level features gradually increases with depth and linear information about higher-level features gradually increases accordingly. This is not something that is easy for us to study, but given that many features with good linear accessibility are central to superposition, understanding the production of this linear information is relevant to understanding features in superposition.

We emphasize that it is already acknowledged in [6] that the application of LEACE in one hidden state does not imply removal in all later hidden states and this part of the motivation behind the technique of "concept scrubbing" presented in the paper. However the method was not used to study an individual layer and to which degree concept erasure in one state affects linear information in the next state.

4.3.1 Method

We propose that study the production of linear information can be done in at least two ways, non-counterfactually and counterfactually.

We can be interested in how the linear accessibility of a feature type z varies with depth s in an observational non-counterfactual way.

Using the notation for residual networks and V-information in 3.1 and 4.1.1: A block f_s *non-counterfactually linear-consolidates* z if $I_{\mathcal{V}}(h_s(X) \to z(X)) > I_{\mathcal{V}}(h_s(X) \to z(X))$ where $\mathcal{V}$ is the family of linear models.

Alternatively and complimentarily, we can make causal interventions that reduce the linear accessibility at one depth s and use this to investigate how this changes the linear accessibility at later depths $s' > s$.

For a feature type z, for a depth s we fit an associated LEACE eraser Eraser_s (see) and apply it to a causal intervention where we apply the eraser at depth s

$$r'^s_t(x) \leftarrow \text{Eraser}^z_s(r^s_t(x))$$

for all $t \in [T]$ when T is the length of sequence x.[5]

We then evaluate at the linear accessibility at depths $s' > s$ using the methods described in Section and . We use F1 as a metric for linear accessibility. We could say that a block f_s *counterfactually linear-consolidates* z if $I_{\mathcal{V}}(h_{s+1}(X)' \to z(X)) > I_{\mathcal{V}}(h_s(X)' \to z(X))$ under erasure of z at s.

A LEACE eraser is an affine function with the properties described above. However in practice such an eraser has to be fitted to data, limiting the erasure. Since the experiments in [6] mostly concern downstream effects on model performance and downstream tasks there is not much information avialable about evaluating the eraser and so in 4.6 we define the validation performance for a fitted eraser which also helps give an idea about what LEACE means in practice. We will note here that as seen in Figure 4.2 the size of the eraser training set is influences how well the eraser performs and we are unable to completely remove the linear information but that this need not stop us from getting some insights.

[5]Here one could also consider only the last position. We think there are arguments for and against and that it is ultimately just two different experiments.

4.3.2 Hypothesis

Informal hypothesis: Transformer blocks are causally involved increasing linear accessibility for the bigram features so that even when removing the linear accessibility at one depth it will be present later.

Specific hypothesis: For 50 bigrams feature types z_κ, averaging over z_κ, erasure of z_κ at $s = 4$ will reduce the linear F1-accessibility at $s = 4$ but the linear F1-accessibility at $s = 5$ will be substantially higher.

4.3.3 Experiment Setup

It is due to the compute resource intensity that we limit ourselves to 50 bigrams and $s = 4$ for 50 different bigrams (a small set due to compute resource intensity).

For each z_κ we train the eraser as described in 4.6. Then for each s we then train a classifier on the the associated r^s_T using sequences the eraser has not seen, and evaluate the accuracy on both probing validation set and for linear F1-accessibility evaluation we use 120,000 tokens from T10K as a sample from the interpretation distribution (see Section 4.3.1).

4.3.4 Results and Further Analysis

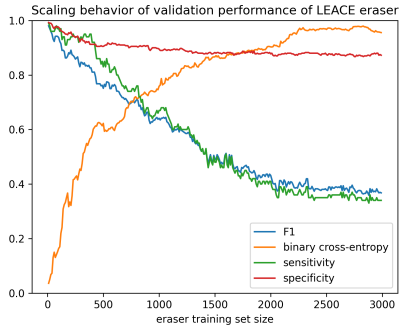


Figure 4.2: Even with large training sets LEACE erasers will not fully remove the linear information.

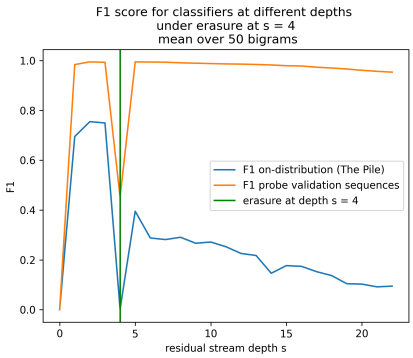


Figure 4.3: Reducing linear information at one depth seems to not result in complete removal at later depths

In Figure 4.3 we show the performances of these classifiers averaging over the 50 bigrams. We see that at $s = 4$ where the erasure occurs, the F1 score drops for both on-distribution and probing sets. For probing set the F1 score ”recovers” again at $s = 5$ while the on-distribution performance is also increasing but thereafter slowly falls.

There is a problem with this experiment, that boils down to The Problem of Positional Binding as described in Section 3.1.1. We cannot conclude

that that block f_4 counterfactually linear-consolidates the bigrams because the residual streams states $r_T^4(X)$ and $r_T^5(X)$ do not correspond to what we denoted $h_s(X)$ and $h_{s+1}(X)$, because these would correspond to *the combined state of the residual streams for all token positions*. In other words it is most plausible that the MSA module in block 4 is responsible for most if not all of this recovery of linear accessibility by transferring information about the identity of the token at position $T-1$ from the residual stream r_{T-1}^4 to the residual stream r_T^5 thereby resulting in a high linear accessibility at $s=5$.

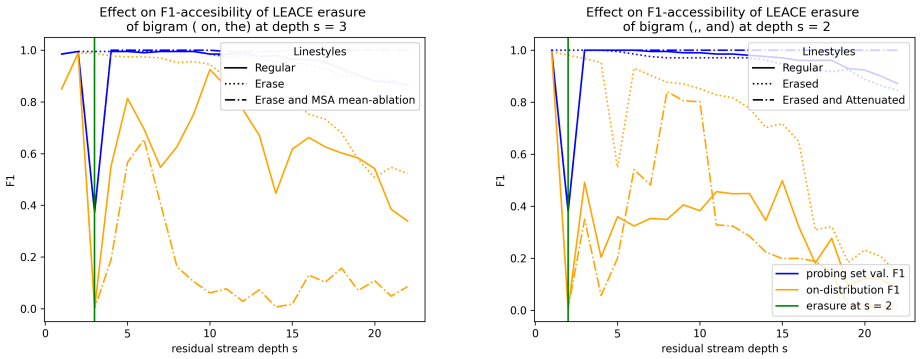
Intervening on MSA outputs

One way of assessing this is to make a more complex causal intervention. In this intervention we will perform erasure of z at s as before but will also perform *batch-mean-ablation* of the MSA module in block f_4. The aim of such an intervention is to remove the transfer of information about the identity of the token at position $T-1$ to the residual stream at position T. We use the notation from A.8.

$$a_T^{s'}(x)' \leftarrow \mathbb{E}_{P_X}[a_T^{s'}(X)]$$

for all downstream depths $s' \geq s$. In practice we will take the mean of all MSA outputs for all tokens in the batch. This has the limitation that it can be a source of noise and error affecting the conclusions. Other choices than batch-mean-ablation could be used, it is however easy to implement and mean-ablation generally does less damage to representations than zero-ablation.

Due to computational limitations we will not perform an experiment on many bigrams again, instead we will perform this intervention on a few bigrams with the hypothesis that the linear accessibility at later depths will now not increase so much.



(a) **(b)**

Figure 4.4: Effect of erasure on downstream linear accessibility when also blocking information transfer by attention module for two different bigrams at depth $s=2$ and $s=3$.

In Figure 4.5 we show examples of two bigrams where we perform erasure at $s = 2$ and $s = 3$ respectively. The thick lines orange line indicates the linear F1-accessibility with only erasure at s and no attention module ablation, while the orange of dots and dashes indicate the more complex intervention described. We see that at depth $s + 1$ the more complex intervention does not result in as high a linear accessibility as with only erasure at s. This is what we hypothesized, indicating that the attention module is involved in contributing to the increase. However, the even when blocking the transfer of information by the attention module in this way the linear accessibility immediately downstream is still considerable, suggesting that the MLP module is also involved. This is in line with another investigation below.

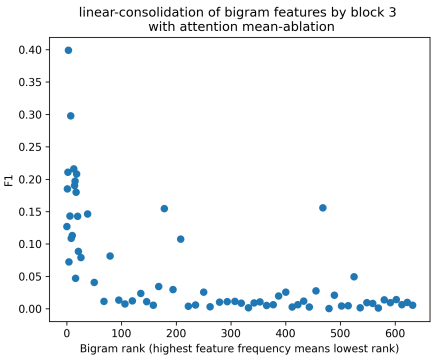
In both figures we see that the downstream accessibility under the complex intervention is markedly different from only erasure at s and that the dash-dotted curves are quite different in the two examples. Since the attention ablation is performed at each downstream depth s' the residual stream $r^T_{s'}(x)$ will be increasingly off-distribution and it is not clear how much we can gain from reflecting on the shapes of these curves especially when it is only two bigrams we are considering.

The relatively substantial linear accessibility immediately downstream under the complex intervention seems to be something that is mainly happening for bigrams with high feature frequency. We consider 50 bigrams with ranks spread relatively uniformly in the interval 0 to 632. For these bigrams we separately perform both kinds of interventions, erasure at $s = 3$ and also both erasure and MSA ablation at $s = 3$ and compare the linear F1-accessibility at $s = 4$. The mean F1-accesibility is 0 at $r^3_T(x)$. For comparison erasure at $s = 3$ the mean F1-accessibility at $s = 4$ is 0.16 and median 0.05. For erasure and MSA ablation the mean F1-accessibility at $s = 4$ is 0.03 and median 0.01. This means that when aggregating over these bigrams block 3 does not seem to actively linear-consolidates bigrams, or that this only happens for high-ranking bigrams. In Figure 4.5a we show how this active linear-consolidation is related to the rank of the bigrams.

Accessibility in the Following MLP

The problem of positional binding is not present if we restrict ourselves to focusing on how after an intervention for erasure of z at s the following MLP in block s processes the ablated $r'^s_T(x)$.

After the at layer s, the ablated residual stream $r'^s_T(x)$ will enter the MLP: First it will go through LayerNorm as described in Section A.4, where we also describe how LayerNorm can be understood as a process of demeaning followed by normalization followed by an affine transformation the last of which can be composed with the first affine transformation of the MLP.



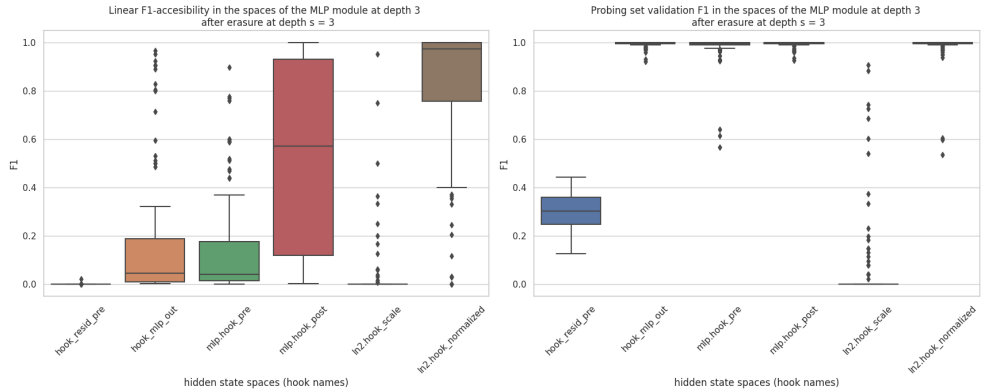
(a) By performing erasure at depth $s = 3$ and mean-ablating the attention module at $s = 3$ it seems that the MLP at $s = 3$ has a small but actual resposibility for actively increasing the linear accessibility of the most frequent bigram feature types.

Figure 4.5

We perform erasure at $s = 3$ for 100 different bigram feature types, and assess the linear F1-accessibility in for the different hidden states in the MLP for the last token position. In Figure 4.6 the names on the x-axis correspond to names of the spaces (so called "hook names" in `TransformerLens`). The boxplot for `hook_mlp_out` is the the space of MLP outputs that are added to the residual stream. The fact that the linear accessibility is low in this space in 4.6a in this space agrees with the results above that the block does not have any significiant level of active linear-consolidation. It is interesting to note however that the state `ln2.hook_normalized` has very high F1-values suggest that LayerNorm can undo the linear erasure, even though this information seems to be destroyed in the following spaces of pre-activations, post-activations and MLP outputs. It seems plausible that the limitations of these kinds of erasure showcased in Figure 4.2 is relevant to understanding unintuitive variation in F1-performance between these spaces.

Conclusion from Further Analysis

Though the original results at the top of this section suggested that Transformer blocks might counterfactually increase the linear accessibility of bigram feature types, further investigations cast doubt on this. In one experiment, we blocked the transfer of information from the residual stream for the token at position $T - 1$ to the residual stream for the token at position T. In the second experiment, we looked at linear accessibility in spaces such as the space of MLP outputs for the MLP at depth s. From these two experiments, we conclude that the attention module is mostly responsible for the increases, but it might be that MLPs are actively involved for bigrams with



(a) Performance on The Pile (T10K) **(b)** Performance on probing validation set

Figure 4.6: F1 performance for assessment of linear accessibility in the following MLP under erasure at $s = 3$

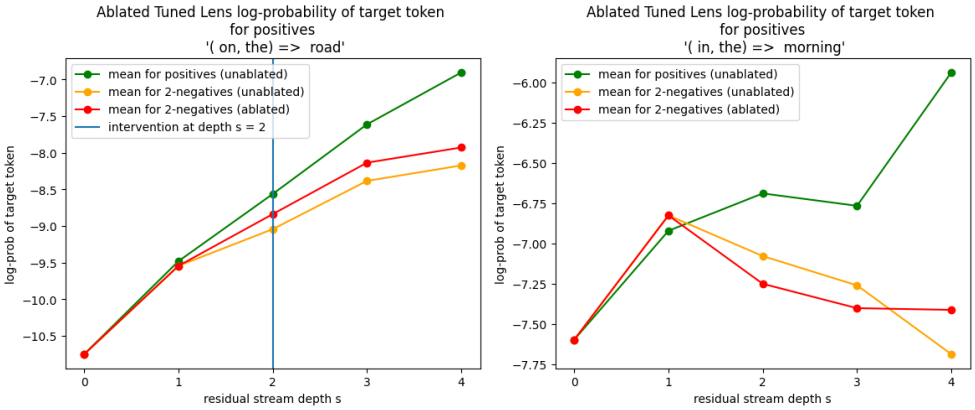
high feature frequencies. The idea that high-frequency bigrams perform differently also suggests that it would be beneficial to repeat the original experiment with bigrams that have a wider variety of feature frequencies, as it is most likely a factor that strongly influences the results.

The idea that modules like MLPs actively increase linear accessibility under the counterfactual of the removal of linear accessibility in the MLP's input space was already quite "bold", though there is evidence that [31] modules perform adaptively under counterfactuals."

4.4 Effect of Directional Activation Intervention on Preliminary Probabilities

The results in Section 4.2 indicate that the feature vectors identified by logistic regression could, to some degree, be viewed as being involved in the model's usage of the feature, though more evidence could be used. In Section 4.3, we motivated the experiment with the view that, alternatively or complementarily to the circuit-view of how models make use of features, one might view the usage of features in terms of modules gradually increasing the linear accessibility of high-level features based on the linear accessibility of low-level features. In this section, we propose to further evaluate the causal influence of the feature directions using the tools of the Tuned Lens [5] [14] and directional activation intervention in a way that might also give some insight into the view just mentioned. As we will explain below, the Tuned Lens allows us to get a view of the "preliminary" predictive distribution [21]. We can then try to understand if a causal intervention using a feature vector for bigram AB has an influence on the preliminary probability for a token C in the consequent set for AB.

This kind of investigation would fill a middle ground between the kind of investigation in Section 4.2, which looked at just the causal relationship between local representation and final output, and the investigation in Section 4.3, which looked at local causal relationships but did not link the feature to something other than itself, like the next-token C. Consider Figure 4.7, which shows two bigrams with associated consequent tokens. We will explain below the technical details behind such plots. At each depth s, the Tuned Lens gives probabilities for sequences x in the associated dataset, and we can look at how the probabilities for the consequent token change when we perform a causal intervention that "activates" the bigram feature in 2-negative sequences that do not end with the bigram. The figures contain one example where the probabilities increase (red versus yellow lines), as one would hope, and another where they do not. We would hypothesize that at least when aggregating over the many bigram interventions, the probabilities move in the right direction.



(a) **(b)**

Figure 4.7: Example of Tuned Lens intermediate predictions of a next-token C for a bigram AB and an example of how applying directional activation patching in residual depth $s = 2$ affects the intermediate predictions for this next-token C.

4.4.1 Method

Direction activation intervention

In Section 4.1.2 we describe in general what we call directional activation intervention. If we are intervening at layer s, we compute the dot-products of positives $\mathcal{X}_{pos}^{train}(r_T^s)$ (see Section 2.3) with $\mathbf{w}_\kappa^2$, and consider the mean t_{mean} and max t_{max}

$$t_{mean} = \frac{1}{|\mathcal{X}_{pos}^{train}(r_T^s)|} \sum_{x \in \mathcal{X}_{pos}^{train}(r_T^s)} [r_T^s(x) \bullet \mathbf{w}_\kappa^s]$$

$$t_{max} = \max_{x \in \mathcal{X}_{pos}^{train}} [r_T^s(x) \bullet \mathbf{w}_\kappa^s]$$

We will perform direction activation intervention at depth s for the last-token position for sequences that are 2-negatives:

$$r_T^s(x) \leftarrow r_T^s + (t - r_T^s(x) \bullet \mathbf{w}_\kappa^s)\mathbf{w}_\kappa^s$$

We will use t_{max} in the experiment because we are interested in getting a strong effect but it might be that using t_{mean} more principled approach.

Tuned Lens

For each depth s in Pythia-410m we are in possession of a Tuned Lens affine transformation that have already been trained for Pythia models [5]. They allows the skipping of the rest of the blocks at depth $s' > s$ to give an early prediction of the final residual stream at depth s_{max} stream $r_T^{s_{max}}(x)$ (see Appendix A).

$$f_s^{tuned}(r_t^s(x)) = \mathbf{W}_s^{tuned} r_t^s(x) + \mathbf{b}_s^{tuned} \in \mathbb{R}^d$$

This is an alternative to the "logit lens" [13] [5] [31], where the prediction for the final residual stream is simply the identity operation

$$f_s^{logit}(r_t^s(x)) = r_t^s(x)$$

We denote by $u_s^{tuned} : \mathbb{R}^d \to \mathbb{R}^V$ as the application of the tuned affine f_s^{tuned} followed by the deembedding operations which results predictive logits over the vocabulary.

$$u_s^{tuned}(x) = E_{out}\text{LayerNorm}(f_s^{tuned}(r_s(x))) + b_{out} \in \mathbb{R}^V$$

We denote by $\pi_s^{tuned} : \mathbb{R}^d \to \mathbb{R}^V$ the log-probabilities gained by using u_s^{tuned}

$$\pi_s^{tuned}(x) = \log(\text{Softmax}(u^{tuned}(s))) \in \mathbb{R}^V$$

For an input sequence x and a target token i, for example (x being "Windows Media Player is developed by" and i being "Microsoft") we have a corresponding series of log-probabilies using the tuned lens, one log-probability for each layer $s \in S$: $\pi_1^{tuned}(x)_1, ..., \pi_S^{tuned}(x)_i \in \mathbb{R}$. which we call a log-probability trajectory.

These Tuned Lenses for Pythia models have been trained on The Pile using a form of distillation loss minimizing the KL-divergence between the preliminary predictive distribution and the predictive distribution obtained all Transformer blocks [5].

Using the Tuned Lenses, for each bigram AB with associated consequent token C, for each 2-negative sequence we have such a trajectory of log-probabilities for the target token C. We then average over 2-negative sequences to get an average log-probability trajectory for 2-negatives $\hat{\pi}^s_{2neg}$ for $s \in [s_{max}]$. The same we do for positives $\hat{\pi}^s_{pos}$. These correspond to respectively the yellow and green lines in Figure 4.7.

When we perform the direction activation interventions at depth $s_{int.}$ on 2-negatives we likewise we an interventional average log-probability trajectory $\hat{\pi}^s_{int,}$ for the 2-negatives, which corresponds to the red lines in Figure 4.7.

Effect ratio

Since we hypothesize that the intervention will move the 2-negative log-probabilites more in the direciton of the log-probabilities for the postives, we will consider the ratio

$$\Delta^s_{pos} = \hat{\pi}^s_{pos} - \hat{\pi}^s_{2neg}$$

$$\Delta^s_{abl} = \hat{\pi}^s_{abl} - \hat{\pi}^s_{2neg}$$

$$\gamma^s = \frac{\Delta_{abl}}{\Delta_{pos}}$$

For $s < s_{int.}$ there is no difference between $\hat{\pi}^s_{abl}$ and $\hat{\pi}^s_{2neg}$ and so we only care about these when $s \geq s_{int.}$

We mentioned in Section 4.2.4 that one unsatisfactory aspect about addition interventions is that quantification of the effect was difficult and how works like [28] try to quantify the "fraction" explained. This kind of ratio will not directly allow for this since it is not guaranteed to be in $[0, 1]$ but could maybe be modified to allow for something like this.

4.4.2 Hypothesis

Informal hypothesis: Directional activation intervention will increase preliminary probabilities for consequent tokens.

Specific hypothesis: For a set of 49 bigram feature types, performing directional activation interventions at depth $s = 2$ will result in positive effect ratios γ^s for most of the bigrams for $s \in \{2, 3, 4\}$.

4.4.3 Results and Discussion

For s being 2, 3 and 4 respectively we find that the fraction of the bigrams considered for which $\gamma^s > 0$ is 0.82, 0.91 and 0.96 respectively. We count this as a confirmation of the specific hypothesis.

We find that for $s = 2$, the Δ^s_{pos} averaged over the bigrams is 1.3 and Δ^s_{abl} is 0.97.

We find that γ^s for individual bigrams have outliers, and as mentioned we are not guaranteed for them to be in $[0, 1]$. We consider median values and for s being $2, 3, 4$ respectively, we find median γ^s for a bigram to be 0.7, 0.75 and 0.84 respectively.

One limitation is the way we evaluate the effect in terms of the ratio: There are many alternative choices one could make in defining these things. For example, we are comparing a ratio of the difference of aggregates. One could also consider the ratio of aggregates of differences, or aggregate of ratios of difference. Such choices might influence the results, and it is not clear what the most principled choice is. Our motivation for this is that we believe the individual effects of the interventions to be quite noisy, and it might be that this way of calculating the ratio is helpful in this regard.

The mentioned noise points to another limitation: The Tuned Lenses have their limitations, especially in this application. They are trained to produce predictive distributions with low *expected* KL-divergence from the actual predictive distribution, and so it is not guaranteed that they should be accurate "lenses" into preliminary predictions for specific bigrams.

Another factor that might be the most important next step in this direction would be to use a larger and more varied set of bigrams. The bigrams we considered have relatively high feature frequencies, and as seen in the previous section, this can have a strong influence on the results.

Furthermore, as we have mentioned in other contexts, the feature vectors identified with logistic regression have their limitations, and it would be worth considering other types of feature directions. Lastly, there are limits to how useful the consequent tokens are for understanding model usage, and choices of model and the parameters used for consequent sets could influence the results. We believe, however, that consequent sets are better suited than empirical next-tokens in this case because consequent tokens might be common among the actual next-tokens for the positive sequences, and thereby our computation of $\hat{\pi}^s_{pos}$ is more directly linked to the bigram feature we care about and avoids other features influencing the log-probability trajectories for positive sequences.

Despite the limitations, we believe that this kind of method, where directional activation interventions or similar kinds of methods are used together with probes for related information, is valuable. In our case, it was model output, and Tuned Lens was used as a form of probe (which is also a perspective described in the paper). While the target token cannot be viewed in our terminology as a feature type because it is not a deterministic function of the data, the approach could be expanded to consider the relationships

between multiple features and their interaction in terms of interventions and linear accessibility assessed with linear probes. It might be that bigram feature types and trigram feature types have some kind of partially hierarchical relationship where bigram feature types are lower-level features and trigram features are higher-level. Using probes for each at different depths and causal interventions to intervene on the relationship could be a possible avenue. Improving the methods of evaluation beyond the limitations of our idea of the effect ratio would also be important in this context.

4.5 Effect of Erasure on General Model Performance

In this last section of the chapter, we perform an investigation that is not set up as an experiment; it is more a form of exploration with a few qualitative evaluations. In 3.3, we considered how certain features could be more important than others and how it might influence the distribution of feature directions.

We might be interested in assessing the degree to which a model linearly uses a feature type at some depth s and compare it to how much it uses two feature types z_1, z_2 at the same depth s. One aspect of such an assessment could be to separately look at the effect of erasure at s of each of the two feature types and evaluate how much it affects the model's overall validation loss.

4.5.1 Conceptual Considerations

However, there are various conceptual aspects to consider:

- While LEACE is supposed to cause minimal collateral damage to the representations, suppose that because of superposition, the linear information about a feature type z_1 is very connected to the representation of the linear information about another feature type z_1', which is quite unrelated in its meaning. Then, the erasure of z_1 could have a significant effect on the loss due to the collateral damage to the linear information about feature type z_1'. If z_2 is in some sense more important than z_1, but z_2 does not have a similarly important z_2' that is significantly affected by the erasure of z_2, then it could seem like z_1 is more important than z_2, even if it is not.
  - This suggests that this approach does not investigate the importance of a feature type in isolation. Instead, it tells us something about the way z_1 is located with respect to other features. If z_1 is related to z_1' in this way, then there is some degree of indistinguishability in the representations of the information about z_1 versus z_1'. While z_1 might not be important in the intuitive sense

of being important completely separately from any other feature, by having z_1 linked to an important feature z_1' in this way, it will be important in the narrow sense that erasure has collateral damage on features like z_1'.

- Another aspect (which might be very related to the above concern depending on how you look at it) might be that with such an approach, it could seem that z_1 is important, but really z_1 is just a narrow version of a more general feature z_1'. The erasure of z_1 does damage to the more general feature z_1', resulting in an overestimation of the importance.
  - First, it is not clear whether the idea is well-defined that we can have cases where z_1 is a more narrow version of a more general feature z_1', and that z_1 is for this reason not a "real" feature. Even so, the linear information in z_1 would carry a signal about the general feature. As long as there is linear information about z_1, how exactly it is being used is a different question than whether and how much it is being used. Even if the information is being used to do computation that does not specifically involve the narrow feature, this does not necessarily mean that the feature is not being used. If the erasure of narrow z_1 ends up having so much collateral damage because of its connection to a more general feature z_1', then this also suggests that the model might not significantly distinguish between the broad and narrow feature.

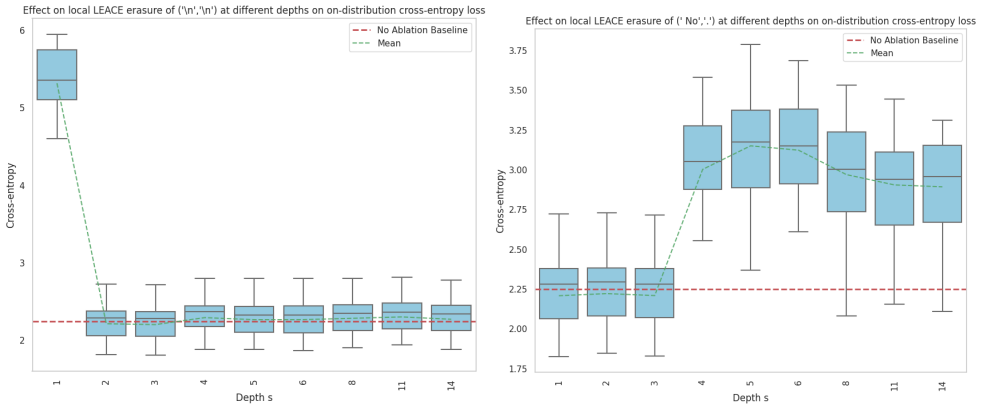
4.5.2 Experiment Setup

There is a limitation to investigating these things: There are many bigram feature types, and for each we might want to investigate erasures at different depths, but for each such erasure we also need to compute the loss on a sufficiently amount of validation data to get an idea of the effect. Thus we use a smaller set of bigrams.

For a set of z_κ, we perform erasure of z_κ at s for $s \in \{1, 2, 3, 4, 8, 11, 14\}$ and compute the resulting cross-entropy under this erasure on 110 sequences of length 600 from T77K, a sample from the interpretation distribution. While this constitutes $66,000$ tokens, it is not a large amount of tokens for evaluation of general model performance, and this is a very big limitation of this idea. The same sequences is being used for each z_κ and s, with a batch size of 10.

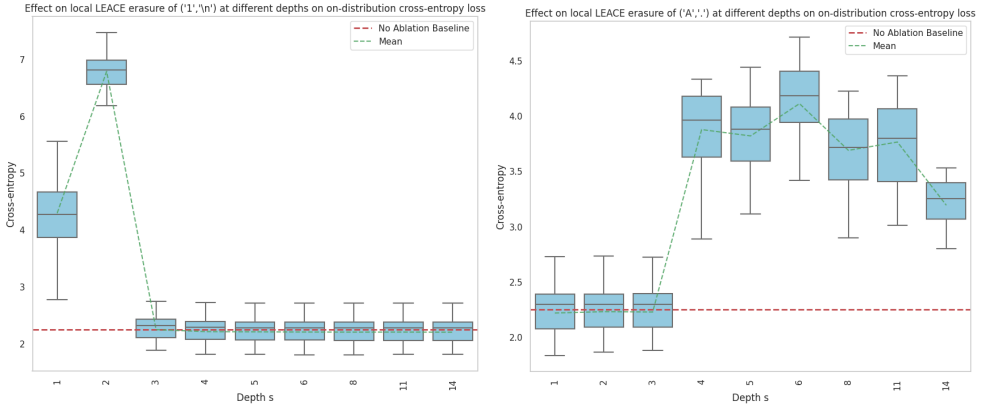
4.5.3 Results

There is considerable variability in the loss among batches, and to highlight the uncertainty we have about the results in Figures 4.8 and 4.9, we show



(a) Bigram (newline, newline) **(b)** Bigram (No., .)

Figure 4.8: Increase in general model performance under erasure of specific bigrams at specific depths s in the residual stream.



(a) Bigram (1, newline) **(b)** Bigram (A,.)

Figure 4.9: Increase in general model performance under erasure of specific bigrams at specific depths s in the residual stream.

the distribution of per-batch loss as boxplots for four different bigrams. We see that the curves of the effect can vary considerably between different z_κ.

For the bigram (newline, newline), which is the most frequently occurring bigram, the effect is strong at $s = 1$ and modest at other depths. For the bigram (1, newline), the effect is still considerable at $s = 1$ but less so, while the effect is very strong at $s = 2$.

For two bigrams (A,.) and (_No, .), the effects are small before $s = 4$, at which point the effect becomes large and remains so for the remaining depths considered. From previous explorations of the PCA of the probing dataset for the bigram (_type,=''), we found that at $s = 4$, the first principal component of $\mathcal{X}(r_T^4)$ sharply separates sequences that end with punctuation (which by

construction can only be 1-negatives) from all other sequences. It has been observed that despite theoretical arguments for why residual streams should give any value to the standard basis, in practice, residual streams can have outlier dimensions, which happens in Pythia-410m at $s = 4$ [17][18] [5]. It might be that there is a relationship between these phenomena.

While ideally, the LEACE eraser and the probing dataset with its 2-negatives that also end with punctuation should allow for removing the linear information about these two bigrams without keeping the effect on punctuation minimal. However, as we have seen, fitted erasers are not perfect (see also Section 4.6.1). While this is something that would be interesting to investigate further, limitations on time and compute resources make us leave these questions open.

We think that the variation in the curves for different bigrams and, in some cases, quite localized effects suggest that approaches like this could make sense. While holding all the mentioned caveats in mind, we also believe that the idea holds that these kinds of computations can allow one to compare different feature types at a specific depth, given the strong effects seen in some of the cases. It would require more rigorous analysis, and the computational costs are a big limitation.

4.6 Chapter Appendix: Defining the Validation Performance of a Fitted LEACE Eraser

For a bigram κ, and depth s, consider the associated training and validation probing dataset $\mathcal{X}(r_T^s)$. As an example we will use the bigram (_type,=") and the depth $s = 4$.

Now consider random splitting of $\mathcal{X}^{train}$ into two equally sized $\mathcal{X}_A^{train}$ and $\mathcal{X}_B^{train}$[6]. We keep the validation set intact as $\mathcal{X}^{val}$

We *fit* a LEACE eraser $\text{Eraser}_A : \mathbb{R}^d \to \mathbb{R}^d$ on $\mathcal{X}_A^{train}$, and thus call $\mathcal{X}_A^{train}$ the eraser training set.

We apply the fitted Eraser_A to transform each of the 3 datasets:

$\mathcal{X}_A^{train'} = \{\text{Eraser}_A(r_T^s) | r_T^s \in \mathcal{X}_A^{train}\}$ and $\mathcal{X}_B^{train'}$, $\mathcal{X}^{val'}$ are defined in a similar manner.

For purpose of understanding, let us look at what happens when we train logistic regression classifier C_1 on the unerased training set $\mathcal{X}_A^{train}$. Then performance of C_1 on each of the datasets is

- $\mathcal{X}_A^{train}$: F1 = 1, because C_1 is trained on this data set.

[6]For now, we do random splitting on the whole set of sequences and not random split for each category of positive, 1-negative and 2-negative separately

- $\mathcal{X}^{val}$: F1 = 1, because $\mathcal{X}^{val}$ is distributed as $\mathcal{X}_A^{train}$ and C_1 has on-distribution generalization.
- $\mathcal{X}_B^{train}$: F1 = 1, because $\mathcal{X}_B^{train}$ is distributed as $\mathcal{X}_A^{train}$ and C_1 has on-distribution generalization.
- $\mathcal{X}_A^{train'}$: F1 = 0, because Eraser$_A$ is trained on this dataset to cause no classifier to be able to get non-trivial performance on this dataset. Thus C_1 which is a classifier does not get non-trivial performance.
- $\mathcal{X}^{val'}$: F1 = 0, because $\mathcal{X}^{val'}$ is distributed as $\mathcal{X}_A^{train'}$ on which Eraser$_A$ is trained to have such a result. Thus Eraser$_A$ generalizes for C_1. (And same for $\mathcal{X}_B^{train'}$)

Now let us train another logistic regression classifier C_2 on $\mathcal{X}_B^{train'}$, and this data set we call the retraining dataset. The performance of C_2 is

- $\mathcal{X}_B^{train'}$ F1 = 0.70 because the classifier is trained on this data and Eraser$_A$ only partly generalizes in its erasure to retraining set and it is a tall order to require a classifier which could be overfit not to have considerable performance on its training set. We call the performance in this case γ.
- $\mathcal{X}_A^{train'}$: F1 = 0.38. This score is not so interesting because it mixes things: The eraser is trained on the $\mathcal{X}_A^{train}$ so we might expect a classifier trained on the erased-distribution to perform worse here than on $\mathcal{X}_B^{train'}$
- $\mathcal{X}^{val'}$: F1 = 0.53. The performance in this case we call ω. This score does not mix things: The eraser is trained on $\mathcal{X}_A^{train}$, the classifier we evaluate the erasure is trained on $\mathcal{X}_B^{train'}$ and is now evaluated on $\mathcal{X}^{val'}$ which is distributed as $\mathcal{X}_B^{train'}$ but neither has the eraser seen $\mathcal{X}^{val}$ nor has the classifier seen $\mathcal{X}^{val'}$.

"With this setup, we believe it is sensible to define γ and ω as the erasure training F1 and erasure validation F1, respectively. Ideally, F1 should be *low* and approach 0, meaning that C_B cannot learn. This is equivalent for other binary metrics derived from the behavior of C_B on $\mathcal{X}_B^{train'}$ and $\mathcal{X}_B^{val'}$, respectively.

These ideas should generally apply to using LEACE in general. Note, however, that for our use case, our probing datasets are generally off-distribution with respect to The Pile and therefore do not show the whole picture.

4.6.1 Validation Performance as a Function of Training Set Size

To understand how the erasure validation performance depends on the size of the eraser training set, we hold the sizes of $\mathcal{X}_B^{train}$ and $\mathcal{X}_B^{val}$ fixed at 1000

and 400, respectively, and likewise for their erased counterparts $\mathcal{X}_B^{train'}$ and $\mathcal{X}_B^{val'}$. Then we randomly permute $\mathcal{X}_A^{train}$ of 3000 elements and for different values of $k \leq 3000$ (step size 10), we train erasers on the first k elements and record the performance metrics shown above. In Figure 4.2, we plot the eraser validation F1 as a function of the size of the eraser training set. As we would hope, the F1 score for the C_B classifier on $\mathcal{X}_B^{val'}$ is lower when the eraser training set is larger; however, we see that beyond 2000 training sequences, the F1 no longer decreases significantly. While this is undesirable, we believe that there are still ways to make use of this erasure for studying feature usage."

Chapter 5

Discussion and Conclusion

The overall goal of this work has been to investigate the idea of features in superposition in realistic Transformer language models. We summarize the work and consider various topics for discussion.

In Chapter 2, we discussed the difficulty in determining what counts as a feature. We opted for a generic approach to features as functions of the data, while taking into account other perspectives, and argued for studying a set of bigram features of sufficient size to understand how such features would be packed in a space by superposition. We advocated for the use of linear probing as a method of investigating superposition and presented the data used.

In Chapter 3, we considered the idea of linear feature accessibility as being a necessary condition for superposition and presented how linear probes could be used to assess accessibility. We conducted a large-scale experiment on the Pythia-410m language model, training large sets of probes. Based on this experiment, we concluded that there was evidence that large sets of features had good linear accessibility. Given that superposition of binary features implies a packing of feature vectors in a way where they are not orthogonal, we considered how these features were distributed. We found that some feature directions were quite cosine similar and that, in this sense, the superposition was not ”isotropic,” which raised questions about whether this should be used as evidence against bigrams being seen as features or how central isotropy is to the idea of superposition. From the idea that different features have different levels of importance in modeling language, we hypothesized that the importance of features is related to how feature vectors are distributed in the space, where more important features are afforded more ”feature dimensionality,” a concept from the original work on superposition that aims to quantify how much capacity is allocated to a feature. We found significant but weak evidence for this idea. It raised questions echoing the preceding discussions in the chapter and in Chapter 3 about the

notion of features and whether features should be seen as separate factors in the data or can group together.

Given that we found evidence for linear accessibility in Chapter 4, we moved on to considering the idea of feature usage as being another necessary condition for superposition, even though we were not able to establish a concrete definition. We conducted experiments where we applied feature directions from the previous chapter in causal interventions, where the feature vectors were added to the residual streams. We hypothesized that such interventions would ”boost” a bigram feature and lead to a change in the model's output behavior, involving an increase in tokens that could be viewed as sensible token-continuations of the bigram. We found some evidence that this was the case and concluded that this was evidence of feature usage. We also performed the experiment using alternative feature vectors, which would not be associated with similar levels of recall and precision, and in this sense were less ”monosemantic” subspaces. Even so, some of these spaces showed considerable evidence of having causal importance. This raised questions about how central to the idea of superposition it is that there exist spaces which are highly monosemantic and are also uniquely responsible for the model's usage of the feature.

We considered that the original motivation for studying features in superposition was to identify such subspaces in order to understand higher-level connections between such features in ”circuits”. We argued that a ”strong” view of circuits entails that representations and usage of features are localized to specific spaces in which they are being used by adjacent modules to detect higher-level features, and that this forms the most natural understanding of what feature usage means. We considered whether superposition could be viewed as an idea that is separate from the hypothesis of circuits and considered an idea of feature usage as involving the gradual increase in linear accessibility of features, which would then be causally related to the linear accessibility of higher-level features. We attempted an investigation into how modules might actively work to make bigram features linearly accessible from the perspective that this question was connected to the view that the model would use such features. By using the concept erasure method, we conducted an experiment where we intervened in a hidden state to remove the linear accessibility and evaluated how a Transformer layer would be responsible for recovering the linear accessibility of the feature. We concluded that the attention module was mainly responsible for the recovery observed, but the experiment had several limitations, including the efficacy of the concept erasers and the choice of bigrams. We found weak evidence that MLP modules might be involved in actively increasing the linear accessibility for features that are very frequent.

We conducted an additional experiment aiming to assess the causal rele-

vance of the feature directions associated with the bigrams. By performing causal interventions that aimed to 'activate' the bigram feature in inputs that did not contain the feature, we investigated how such interventions were associated with changes in the probability of the tokens that are likely continuations of the bigram. Instead of considering the final probabilities in the model output, we used the tool Tuned Lens to inspect the model's 'preliminary' predictive distribution after the intervention. Using this approach, we found some additional evidence for the causal relevance of the feature vectors. We argued that some aspects of the methods in this experiment, including the directional activation patching, evaluating the causal effect on a target as a ratio, and connecting probes for multiple features, could be a promising direction for future work.

In line with the investigation of the distribution of feature vectors and their relationship to each other, and considerations about the importance of feature vectors, we experimented with the idea that concept erasure could be applied to a feature in some state to form an ablated model whose new overall cross-entropy performance could give insight into the importance a feature plays in a model. Considerations in this context mirrored the earlier questions about the grouping of features.

Where does this leave us? The results are evidence for a narrow version of superposition where language models can represent many features in a hidden space where these features have good linear accessibility and some degree of usage. While there are certain limitations of the experiments that could be addressed, looking forward, it seems that linear probes combined with causal linear interventions could be extended to study a wider variety of features, especially the relationships between features, which has not been studied in this work because the features considered are mutually exclusive by construction."

If we allow ourselves to discuss this topic at a more abstract level, which is somewhat removed from specific conclusions that can be directly drawn from the experiments but seems to underlie some of the discussions, we would mention the following: It is worth being explicit and formal about the different possible views and perspectives on the notions of features, representations, superposition, and circuits. In the introduction, we articulated the 'strong view' of these concepts and their relationships. Strong views are good because they make it easier to know what is being claimed and not being claimed, and in some cases, will allow a theory to be falsifiable. We have claimed that our results contain evidence for a narrow form of superposition described in terms of accessibility and usage, but a key difficulty is establishing specific criteria for usage. Especially the notion of features seems to be difficult, and it would be helpful to articulate strong, falsifiable views of what it would mean for models to operate on discrete fea-

tures versus something more continuous, and where the line is to be drawn. Falsifiable views on circuits versus alternative views on the model's computational process would likewise be beneficial. It seems that focusing on the cases in which a model performs somewhat symbolic computation that might require discrete features could be an important aspect of formalizing such strong views.

Appendix A

GPT architecture

In this thesis, we study decoder-only causal Transformer language models. In this chapter we present a self-contained introduction of this architecture that will provide the necessary information necessary to present the following chapters. The most well-known decoder Transformer language models are the GPT models by OpenAI, and therefore we will refer to them as GPT models. [39] However, architectural details is a (very) active field of research and we are specifically focusing on the architectural details of the models that we are studying.

Let the vocabulary $\mathcal{V} = \{1, 2, ..., V\}$ be a finite set. Our dataset $\mathcal{D} = \{D_i\}_i^N$ consists of sequences $D_i = (d_1, ..., d_{T_i}) \in \mathcal{V}^{T_i}$ of various lengths T_i for $i \in [N]$. Thus a dataset is a set of N such sequences. It is common to refer elements of $\mathcal{V}$ as a token and likewise a specific instance of such element in a specific sequence as a token. In many cases this ambiguity is not a big deal, but due to the details of our studies we will though not common in machine learning, adopt the type-token distinction used in other fields: We call an element $v \in \mathcal{V}$ a *type* and specific instances of this type we call *tokens*. Thus a sequence in the dataset consists of tokens, while a model might predict a specific type as the next token. [37]

Behaviorally, the model M is a function mapping a sequence of T tokens to T predictive distributions over the vocabulary. If Δ_V denotes the space of probability mass functions over the vocabulary, the V-simplex,

$$M : \mathcal{V}^T \to \Delta_V^T$$

For an input sequence $s \in \mathcal{V}^T$, the *predictive distribution* $M(s)_t \in \Delta_V$ is a probability mass function over the vocabulary predicting the next token to after $s_{\leq t}$. However, we will only be concerned with the last of these predictive distibutions $M(s)_T \in \Delta_V$

A GPT model has a set of hyperparameters, for our use case the most important ones are: the size of the vocabulary $V = |\mathcal{V}|$, the hidden dimension

$d \in \mathbb{N}$, the number of Transformer blocks $n_{layers} \in \mathbb{N}$ and $n_{heads} \in \mathbb{N}$ which is an an integer, such that d is divisible by n_{heads} specifying the number of *attention heads* in a layer.

A.1 Embedding layer

Behaviorally, the embedding layer is a function mapping from vocabulary set to $\mathbb{R}^d$

$$f_{embed} : \mathcal{V} \to \mathbb{R}^d$$

Implementation wise, the embedding layer consists of a $d \times V$ real matrix E_{in}, each column is the *embedding* for the associated token in $\mathcal{V}$, meaning that for each token in $\mathcal{V}$ there is an associated vector in $\mathbb{R}^d$. Furthermore the embedding layer has a bias vector $b_{in} \in \mathbb{R}^d$.

Given a type $v \in \mathcal{V}$, the one-hot encoding $\text{onehot}(v) \in \mathbb{R}^V$ makes the embedding layer an affine transformation

$$f_{embed}(v) = E_{in}\text{onehot}(v) + b_{in}$$

The input to the model is a sequence $(d_1, ..., d_T)$, and we now associate with each token in the sequence its embedding vector, giving $(e_1, ..., e_T) \in \mathbb{R}^{T \times D}$ where $e_t \in \mathbb{R}^d$ for $t \in [T]$, where $e_t = f_{embed}(d_t)$

A.2 Residual streams

The Transformer has a *residual* network architecture.

A residual neural network of n_{layers} layers, is one where each hidden state $s \in [n_{layers}]$ additively updates the hidden state. Assuming the input is x_{input} and layer s is denoted f_s

$$x_0 = x_{input}$$

$$x_s = x_{s-1} + f_s(x_{s-1})$$

for $s > 0$. We call $f_s(x_{s-1})$ a residual update at *depth* s.

In GPT models we start with the sequence of embedding vectors $(e_1, ..., e_T)$ and each layer is a *Transformer block* that updates this sequence of embedding vectors additively:

$$(r_1, ..., r_T)_0 = (e_1, ..., e_T) \in \mathbb{R}^{T \times d}$$

$$(r_1, ..., r_T)_s = (r_1, ..., r_T)_{s-1} + f_s((r_1, ..., r_T)_{s-1}) \in \mathbb{R}^{T \times d}$$

This addition operation is position wise, meaning $(a_1, ..., a_T) + (b_1, ..., b_T) = (a_1 + b_1, ..., a_T + b_T)$ and $f_s : \mathbb{R}^{T \times d} \to \mathbb{R}^{T \times d}$

We use the notation r_t^s to denote the hidden state at position t at depth s. Fixing a specific position t, the the sequence of hidden states $r_t^1, ..., r_t^{n_{layers}}$ we call the *residual stream* for position t. Thus we view the computational process as one where T residual streams are iteratively updated. [18]

A.3 Transformer block

Behaviorally, the Transformer block at depth s is a function

$$f_s : \mathbb{R}^{T \times d} \to \mathbb{R}^{T \times d}$$

providing the residual updates described above. A Transformer block consist of two main parts, Multi-head Self Attention (MSA) and position wise multi-layer perceptron (MLP) both of which are preceded by LayerNorm (LN). The models we study are *parallel* Transformer blocks where MSA and MLP operates independtly on the input:

$$f_s(r) = (f_s^{att} \circ f_s^{ln1})(r) + (f_s^{mlp} \circ f_s^{ln2})(r) := MSA(r) + MLP(r)$$

This is in contrast to a non-parallel approach which is very common:

$$f_s(r) = (f_s^{mlp} \circ f_s^{ln2} \circ f_s^{att} \circ f_s^{ln1})(r)$$

A.4 LayerNorm

A LayerNorm layer [2] has the signature

$$f_s^{ln} : \mathbb{R}^{T \times d} \to \mathbb{R}^{T \times d}$$

It has two parameters $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$ It operates indepently on each position $t \in [T]$:

$$f_s^{ln}(r_1, ..., r^T) = (g(r_1), ..., g(r_T))$$

where $g_s^{ln} : \mathbb{R}^d \to \mathbb{R}^d$ is

$$g(\mathbf{x}) = \mathbf{a} \odot \frac{\mathbf{x} - \text{mean}(x)}{\text{std}(x)} + \mathbf{b}$$

where $\odot$ is element-wise multiplication (Hadamard product).

In essence LayerNorm operates on each position independently by Z-normalizing the entries and applying the same learned scaling and shift vector.

Another interpretation is also useful in some cases: The operation of subtracting the mean is a linear operation $\mathbf{x}' = \mathbf{x} - \text{mean}(\mathbf{x}) = \mathbf{D}\mathbf{x}$ where $\mathbf{D} = \in$

$\mathbb{R}^{d\times d}$. This linear transformation projects out the vector $(1,...,1)^T \in \mathbb{R}^d$. Since $\mathbf{x}'$ has mean 0, the standard deviation is proportional to it's L2-norm.

$$\text{std}(\mathbf{x} - \text{mean}(x)) \propto ||\mathbf{x} - \text{mean}(x)||_2$$

This means that LayerNorm can also be understood as projecting out a single dimension and L2-normalizing. The consequence is that MLPs and MSAs can be viewed as only operating on a *view of the residual stream where all vectors are unit-norm*, this "view" is the L2-normalized demeaned residual stream.

A.5 Multi-head Self Attention

Since the internals of multi-head self attention modules are not involved in the work (which is arguably a limitation), we refer to [38] [18] for information about self-attention and to [42] and https://blog.eleuther.ai/rotary-embeddings/ for information about rotational embeddings.

A.6 Multi-Layer Perceptron

The MLP layer has the signature

$$f_s^{mlp} : \mathbb{R}^{T\times d} \to \mathbb{R}^{T\times d}$$

It operates independently on each position

$$f_s^{mlp}(r_1,...,r^T) = (g_s^{mlp}(r_1),...,g_s^{mlp}(r_T))$$

where $g_s^{mlp} : \mathbb{R}^d \to \mathbb{R}^d$ is a multi-layer perceptron with one hidden layer of size $4d$ with GELU activation. [23] [27]

A.7 Unembedding Layer

The unembedding layer has the signature

$$f^{out} : \mathbb{R}^{T\times d} \to \Delta_V^T$$

For each position in the sequence it produces a PMF over the vocabulary $\mathcal{V}$.

It operates independently on each position:

$$f^{out}(\mathbf{r}_1,...,\mathbf{r}^T) = (g^{out}(\mathbf{r}_1),...,g^{out}(\mathbf{r}_T))$$

so if $r \in \mathbb{R}^d$

$$g^{out}(\mathbf{r}) = \text{Softmax}(E_{out}\text{LayerNorm}_{unembed}(\mathbf{r}) + b_{out}) \in \Delta_V$$

Where $E_{out} \in \mathbb{R}^{V \times d}$, $b_{out} \in \mathcal{R}^d$

The output $(o_1, ..., o_T)$ of the unembedding layer are T probability mass functions over the vocabulary $p_1, ..., p_T$. The probability mass function p_t is used to predict the token at position $t + 1$.

These networks have then been trained on a large data set to maximise the likelihood of the training data by minimising the cross-entropy using variants of stochastic gradient descent.

A.8 Hook Names

When studying a model M, we will consider the vectors generated at different locations within the network. These vectors can be accessed and intervened upon using the PyTorch library `TransformerLens` [34], which defines various hook names. We will use the TransformerLens hook name notation to refer to these locations:

Transformer blocks are zero-indexed, and we will primarily refer to the following hooks: For layer s (zero-indexed) and token position t (1-indexed) we have hook names

- "`block.<layer>.hook_mlp_in`" we refer to using notation $m_t^s \in \mathbb{R}^d$
- "`block.<layer>.hook_attn_out`" and $a_t^s \in \mathbb{R}^d$
- "`block.<layer>.hook_resid_pre`" and $r_t^s \in \mathbb{R}^d$
- "`block.<layer>.hook_resid_post`" and $r_t^{s+1} \in \mathbb{R}^d$

A specific hookname name could be the MLP output in layer 3 (zero-indexed): `blocks.<layer>.hook_mlp_out` which we denote $m_t^3 \in \mathbb{R}^{T \times d}$.

When the input token sequence $x \in \mathcal{V}^T$ is made implicit we write $m_t^s(x) \in \mathbb{R}^d$ (respectively a and r), and when x is implicit we write m_t^s. When we omit the subscript t we are referring to the last token position $m^s = m_s^T$

Due to the residual structure and the parallelism in the models, we have the relationship

$$r_t^{s+1}(x) = r_t^s(x) + a_t^s(x) + b_t^s(x)$$

When we have a large set $\mathcal{S}$ of sequences of the same length, it eases the presentation to think of the input token sequence as a random variable $X \in \mathcal{V}^T$, giving associated random variables $M_t^s, A_t^s, R_t^s \in \mathbb{R}^d$

Appendix B

Intuitions about High-Dimensional Spaces and Feature Directions

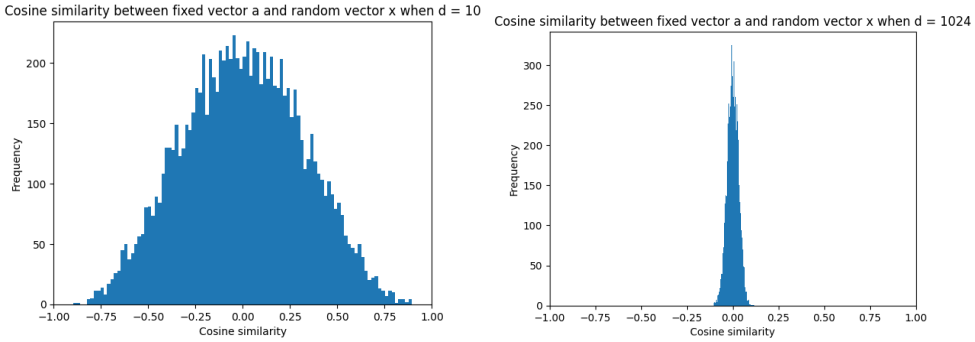
In this section we will discuss some aspects of high-dimensional spaces through simple simulations and reflect on the relevance for how neural networks that operate on high-dimensional spaces might represent features of the data. The main models we will study later has hidden dimension $d = 1024$. However much larger models such as the open source model Falcon-180B has $d = 14848$.

B.1 Concentration of measure

Fix a (random) vector $\mathbf{a}$ on a d-dimensional unit-sphere, consider then another random vector $\mathbf{x}$ sampled uniformly on the unit-sphere [1]. What is the distribution over cosine similarity $sim(\mathbf{a}, \mathbf{x})$? It depends on d. For example, when d is very low such as 10 there is a decent probability that the similarity is greater than 0.2 but when $d = 1024$ the probability that the similarity is greater than 0.2 is vanishingly small. In Figure B.1 we plot histograms of similarity when sampling a random a and 10K random $\mathbf{x}_i$ recording $sim(\mathbf{a}, \mathbf{x}_i)$.

This extremely low chance of being outside an interval around 0 leads to the idea of a pair of vectors being pseudo-orthogonal: In a high-dimensional space, as long as the absolute cosine-similarity between two vectors is within a decently small ε, we can consider them pseudo-orthogonal or simply dissimilar. On the other hand, if cosine-similarity is greater than ε, we can consider that this is not something that is very likely by chance and therefore consider them similar.

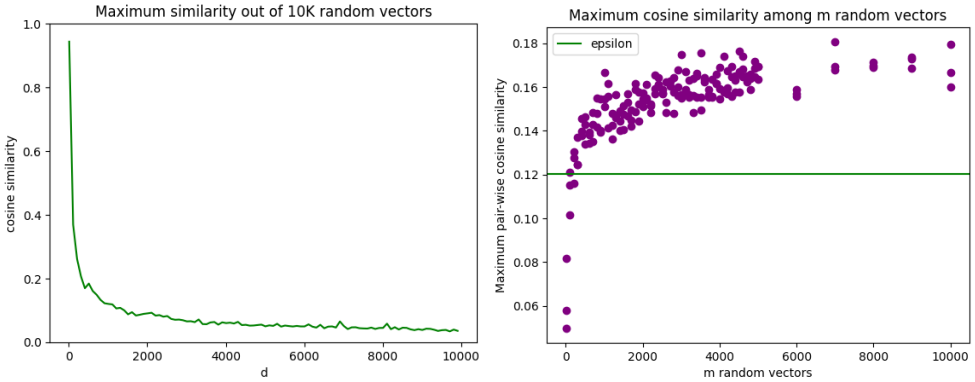
[1] One can efficiently sample uniformly from the unit-sphere by sampling from an isotropic multivariate Gaussian and rescaling to unit norm



(a) d = 10 **(b)** d = 1024

Figure B.1: Histogram showing the cosine-similarity between two randomly chosen vectors. In high dimensional spaces two random vectors are very unlikely to have significant cosine-similarity. This could form the basis of having very many vectors being almost-orthogonal.

Of the 10K random vectors how close is the closest one? If we let this be the definition of ε we can run simulations like the ones above, by sampling $n = 10K$ random vectors for different d. Of course, in the limit of sampling $N \to \infty$ vectors, $\varepsilon \to 1$ so what we are concerned about it just having a sufficiently low probability of similarity. In Figure B.2a we see how this maximum decreases rapidly as d increases towards $d = 2000$ after which it is very small and decreasing slower.



(a) As the dimensionality increases the highest similarity observed among 10 thousand random vectors decreases rapidly.

(b) As we increase the number of random vectors in 1024 dimensional space the maximum similarity between any pair grows quite slowly.

Figure B.2

B.2 Packing dissimilar vectors

In d dimensions it is possible to have d vectors that are all mutually orthogonal, for example the standard basis vectors. It is one thing to say that it is unlikely for two random vectors to be similar, and another thing to talk about how many vectors we can find that are all mutually dissimilar. If our bound of similarity is sufficiently high, it turns out that it is a lot:

Consider again $d = 1024$ and now m randomly sampled vectors from the unit sphere. We can compute the mutual cosine-similarities between all $\binom{m}{2}$ pairs of vectors and identify the greatest cosine-similarity. Conducting this 3 times for multiple versions of m we plot the result in Figure B.2b. For example sampling 2K random vectors we find that there is a pair that has cosine similarity 0.16 but all other pairs are are less similar.

The vertical line $y = \varepsilon = 0.12$ denotes the threshold above which the m random vectors are no longer mutually dissimilar in the sense that all mutual cosine similarities are below ε from the previous subsection. We notice that this threshold is crossed very early, meaning that sampling randomly we get a much worse outcome than choosing the standard basis.

However, we can also notice that the trend is for the maximum similarity to grow only very slowly as m increases. Even for $m = 10K$ the maximum similarity is still less than 0.2

It is worth noting that originally we defined ε as the threshold for similarity based on the vector $\mathbf{x}$ that was most similar to $\mathbf{a}$ out 10K random $\mathbf{x}_i$. This means that this threshold can be increased to $\varepsilon' > \varepsilon$ and it would only mean that the probability of a random unit-norm vector X being similar to $\mathbf{a}$ is lower

$$\varepsilon' > \varepsilon \implies P(sim(X, a) > \varepsilon') < P(sim(X, a) > \varepsilon)$$

The maximum similarity plot above suggests that $\varepsilon' = 0.18$ is a reasonable threshold for similarity when $d = 1024$.

Why do we care about m? In the context of deep learning, we are interested in m because we can consider these m random directions as *feature directions*: When ε' is the threshold, the d-dimensional can support a very high number $m > d$ of pseudo-orthogonal directions. Instead of having a vector space where each dimension represents a one-dimensional feature and thus being limited to d one-dimensional features, pseudo-orthogonality might allow for representing a much larger number of one-dimensional features in a smaller space.

B.3 How many features can be bound in a vector?

In the fully-orthogonal case where each standard basis vector corresponds to a feature direction, any vector $\mathbf{u}$ can encode the scalar value of its features in its entries which can vary independently. If $\mathbf{e}_i$ are the standard basis vectors and u_i is entry i of u, then

$$\mathbf{u} = u_1 \cdot \mathbf{e}_1 + u_2 \cdot \mathbf{e}_2 + ... + u_d \cdot \mathbf{e}_d = \mathbf{I}u$$

Furthermore, the degree to which a feature u_i is present in a vector $\mathbf{u}$ can be determined by the dot product

$$u_i = \mathbf{u} \cdot \mathbf{e}_i$$

and for cases where both $\mathbf{u}$ and basis vectors $\mathbf{e}_i$ are unit vectors, this is equivalent to the cosine similarity $u_i = sim(\mathbf{u}, \mathbf{e}_i)$

In the pseudo-orthogonal case where there are $m > d$ feature directions $\mathbf{f}_1, ..., \mathbf{f}_m$, this is not possible. If we maintain the idea of encoding feature content by similarity, we can denote the amount z_i of feature $i \in [m]$ encoded in $\mathbf{z}$ by

$$z_i = \mathbf{u} \cdot \mathbf{f}_i$$

we are less flexible, since changing $\mathbf{u}$ will change the similarity to many different $\mathbf{f}_j$. We are less flexible, but how less flexible?

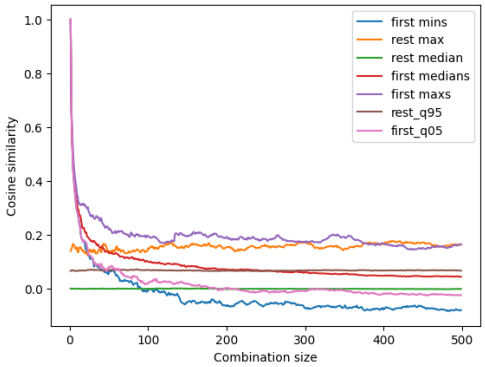
Consider the case where $\mathbf{f}_i$ are random directions on the d-dimensional unit-sphere. In the fully-orthogonal case, we might want a vector that encodes the presence of feature u_1, u_2, u_3 but absence of all other other features $u_3, ..., u_d$. This would be done by having a vector such as $\mathbf{u} = (1, 1, 1, 0, ..., 0) = \mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_3$.

In the pseudo-orthogonal case we might want a vector that encodes the presence of features z_1, z_2, z_3 but absence of all other features $z_4, ..., z_m$. One attempt at doing this could be having a vector such as $\mathbf{u} = \mathbf{f}_1 + \mathbf{f}_2 + \mathbf{f}_3$. In it generalized form representing the presence of features $z_1, ..., z_k$ is done by $u = \sum_j^k \mathbf{f}_j$

The goal would be that $\mathbf{u}$ will be similar to $\mathbf{f}_1, ..., \mathbf{f}_k$ but *not* similar to $\mathbf{f}_{k+1}, ..., \mathbf{f}_m$. Does this work, and if so how large can k be before this no longer works?

In a simulation we now consider $d = 1000$ and consider various numbers of present features $k = 1, ..., 500$. For each k we define a $\mathbf{u} = \sum_j^k f_j$ and compute the scores, $z_i = \mathbf{u} \cdot \mathbf{f}_i$. We consider consider $z_1, ..., z_k$ the "first" and $z_{k+1}, ..., z_m$ the rest. We would want all firsts to be above ε', indicating similarity and equivalently "presence" of the feature, while we want to avoid that any of the rest are significantly above ε' indicating dissimilarity.

What we find is that it is that already at a low value of k , there will a f_j for $j > k$ and a f_i for $i \leq k$ where $sim(u, f_j) > sim(u, f_i)$. This is at the point where the blue and the orange line crosses. At this point, if we try to let u represent the presence of k features as a linear combination we will also start having false postives, where u is understood to represent the presence features that we did not intend.



(a) We can try to understand how many random feature directions a vector can be similar to at the same time while being non-similar to the other feature directions.

Figure B.3

This suggests that if we have data that has a certain property, that in a data point, only a limited number of features are present at the same time, then using pseudo-orthogonality has its uses, but on the other hand pseudo-orthogonality is very limited in its ability to allow *all* features to vary freely. Within the smaller number of features we can vary freely.

Appendix C

Reproducibility

- Code and data is available at https://github.com/kmrasmussen/pythia_tools/tree/main/superposition
- Experiments were conducted on Google Colab using a T4 GPU with 15 GB VRAM and 50 GB CPU RAM which currently consumes 2 Google Colab compute units per hour. https://colab.research.google.com/. Though the exact amount of GPU compute has not been tracked, we estimate it to be at most 600 units. Using a calculator that estimates CO_2 consumption mlco2.github.io/impact we estimate this to be below 30 kg CO_2 eq. (depending on the geographical location of the servers). According to the calculator Google Colab uses full carbon offsetting.
- The primary Transformer language model used is Pythia-410m https://huggingface.co/EleutherAI/pythia-410m, specifically the version `EleutherAI/pythia-410m-deduped-v0` which is also available as part of TransformerLens [34] https://github.com/neelnanda-io/TransformerLens
- The data used for experiment is from The Pile-v1 https://pile.eleuther.ai/. At the time of the experiments, The Pile was not available from the website and the data for T322K was downloaded from https://academictorrents.com/details/0d366035664fdf51cfbe9f733953ba325776e667m. The Pile Validation from which T77K and T10K was downloaded from https://pile.eleuther.ai/ when it was available.

Appendix D

Note on Relation to Project Description

The original title of the project was "Understanding features in superposition in GPT feed-forward layers" and emphasized the MLP modules. Even at the stage of the project description, it was clear that there were some ambiguities regarding how to think about superposition in the residual stream and the MLP spaces. The work shifted its focus to the residual stream for the following reasons:

- MLP activation spaces are generally four times as large as the residual stream, and studying superposition in these spaces using the approach we considered would require many more features and more computational resources.
- Since the usage of features is arguably an important aspect of superposition, we considered superposition in the residual stream to be more important, as it is more natural to think of modules as utilizing features represented in the residual stream.
- The project description proposed that the relationship between MLPs and superposition might be viewed from the perspective that MLPs use features in the residual stream. Therefore, the work aimed to establish features in the residual stream to better understand how MLPs might use such features. However, apart from some parts of Section 4.3, this has not been the primary focus as intended.
- One idea was that, following [25], one could attempt to understand early MLPs in relation to bigrams. However, it was found that even in the residual stream at depth $s = 1$, many probes for bigrams performed well. Therefore, the work shifted its focus to understanding this aspect.

- Some work was done with the intention of better understanding the causal influence of MLPs, but this work remains unfinished and has not been included in the thesis.

Despite the original intended focus on MLPs, the most important motivation was the general idea of studying superposition in realistic Transformer language models. The methods and experiments conducted should apply to MLPs, assuming sufficient resources to conduct the experiments in higher spaces which which requires more data and more compute and time for training and storage in cases probing data sets are cached.

Bibliography

[1] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *ArXiv*, abs/1610.01644, 2016.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

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

[4] Nora Belrose. Cohere for ai - community talks - on concept erasure and elicit latent knowledge. YouTube, 2023. Available: https://www.youtube.com/watch?v=y6Z8CYZz1eo.

[5] Nora Belrose et al. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

[6] Nora Belrose, David Schneider-Joseph, Shauli Ravfogel, Ryan Cotterell, Edward Raff, and Stella Biderman. Leace: Perfect linear concept erasure in closed form, 2023.

[7] Stella Rose Biderman, Hailey Schoelkopf, Quentin G. Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling. *ArXiv*, abs/2304.01373, 2023.

[8] Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models. https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html, 2023.

[9] Tolga Bolukbasi, Adam Pearce, Ann Yuan, Andy Coenen, Emily Reif, Fernanda Vi'egas, and Martin Wattenberg. An interpretability illusion for bert. *ArXiv*, abs/2104.07143, 2021.

[10] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[11] 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. *CoRR*, abs/2005.14165, 2020.

[12] Rosa Cao. Putting representations to use. *Synthese*, 200(2), 2022.

[13] Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. Analyzing transformers in embedding space. *ArXiv*, abs/2209.02535, 2022.

[14] Alexander Yom Din et al. Jump to conclusions: Short-cutting transformers with linear transformations. *arXiv preprint arXiv:2303.09435*, 2023.

[15] Nelson Elhage, Tristan Hume, Catherine Olsson, Neel Nanda, Tom Henighan, Scott Johnston, Sheer ElShowk, Nicholas Joseph, Nova DasSarma, Ben Mann, Danny Hernandez, Amanda Askell, Kamal Ndousse, Andy Jones, Dawn Drain, Anna Chen, Yuntao Bai, Deep Ganguli, Liane Lovitt, Zac Hatfield-Dodds, Jackson Kernion, Tom Conerly, Shauna Kravec, Stanislav Fort, Saurav Kadavath, Josh Jacobson, Eli Tran-Johnson, Jared Kaplan, Jack Clark, Tom Brown, Sam McCandlish, Dario Amodei, and Christopher Olah. Softmax linear units. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/solu/index.html.

[16] 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. https://transformer-circuits.pub/2022/toy$_m$*odel*/*index.html*.

[17] Nelson Elhage, Robert Lasenby, and Christopher Olah. Privileged bases in the transformer residual stream. Anthropic, Mar 2023. Core Research Contributor: Nelson Elhage; Correspondence to: Christopher Olah (colah@anthropic.com).

[18] 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. https://transformer-circuits.pub/2021/framework/index.html.

[19] Leo Gao, Stella Rose Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling. *ArXiv*, abs/2101.00027, 2020.

[20] Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas F. Icard, and Noah D. Goodman. Finding alignments between interpretable causal variables and distributed neural representations. *ArXiv*, abs/2303.02536, 2023.

[21] Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *arXiv preprint arXiv:2203.14680*, 3 2022. Cited by 58.

[22] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 12 2020.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[24] Clément Guerner, Anej Svete, Tianyu Liu, Alexander Warstadt, and Ryan Cotterell. A geometric notion of causal probing. *arXiv preprint arXiv:2307.15054*, 2023.

[25] Wes Gurnee et al. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*, 2023.

[26] Jacqueline Harding. Operationalising representation in natural language processing. *British Journal for the Philosophy of Science*, forthcoming.

[27] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[28] Jing Huang, Atticus Geiger, Karel D'Oosterlinck, Zhengxuan Wu, and Christopher Potts. Rigorously assessing natural language explanations of neurons. *arXiv preprint arXiv:2309.10312*, 9 2023.

[29] et al. Ilyas, Andrew. Adversarial examples are not bugs, they are features. In *Advances in neural information processing systems 32*, 2019.

[30] Pentti Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, 1(2):139–159, 2009.

[31] Tom McGrath, Matthew Rahtz, János Kramár, Vladimir Mikulik, and Shane Legg. The hydra effect: Emergent self-repair in language model computations. *ArXiv*, abs/2307.15771, 2023.

[32] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. In *Neural Information Processing Systems*, 2022.

[33] Eric J. Michaud, Ziming Liu, Uzay Girit, and Max Tegmark. The quantization model of neural scaling. *ArXiv*, abs/2303.13506, 2023.

[34] Neel Nanda and Joseph Bloom. Transformerlens, 2022.

[35] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. https://distill.pub/2020/circuits/zoom-in.

[36] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

[37] Charles Sanders Peirce. Prolegomena to an apology for pragmaticism. *Monist*, 16:492–546, 1906.

[38] Mary Phuong and Marcus Hutter. Formal algorithms for transformers. *ArXiv*, abs/2207.09238, 2022.

[39] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019. https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.

[40] Shauli Ravfogel, Yoav Goldberg, and Ryan Cotterell. Linear guardedness and its implications. *arXiv preprint arXiv:2210.10012*, 10 2022.

[41] Shauli Ravfogel, Michael Twiton, Yoav Goldberg, and Ryan D Cotterell. Linear adversarial concept erasure. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 18400–18421. PMLR, 17–23 Jul 2022.

[42] Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *ArXiv*, abs/2104.09864, 2021.

[43] Yanmin Sun, Andrew K. C. Wong, and Mohamed S. Kamel. Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(04):687–719, 2009.

[44] Curt Tigges et al. Linear representations of sentiment in large language models. *arXiv preprint arXiv:2310.15154*, 2023.

[45] Alex Turner et al. Activation addition: Steering language models without optimization. *arXiv preprint arXiv:2308.10248*, 2023.

[46] Zhengxuan Wu, Atticus Geiger, Christopher Potts, and Noah D. Goodman. Interpretability at scale: Identifying causal mechanisms in alpaca. *ArXiv*, abs/2305.08809, 2023.

[47] Yilun Xu et al. A theory of usable information under computational constraints. *arXiv preprint arXiv:2002.10689*, 2020.

[48] Andy Zou et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

UNDERSTANDING FEATURES IN SUPERPOSITION IN TRANSFORMER LANGUAGE MODELS

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
RASMUSSEN	**KASPER**

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
ZURICH, NOVEMBER 13, 2023	

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.