AI Safety Student Team

Intro Fellowship

Spring 2024

[WEEK 5]

Model internals

READINGS

Locating and Editing Factual Associations in GPT: Companion blog post
(Meng et al.,)

Discovering latent knowledge in language models without supervision (Pages 1-5)
(Burns et al., 2022)

Inference-Time Intervention: Eliciting Truthful Answers from a Language Model (§ 1-3)
(Li et al., 2022)

Locating and Editing Factual Associations in GPT: Companion blog post

rome.baulab.info/

Notes:

DISCOVERING LATENT KNOWLEDGE IN LANGUAGE MODELS WITHOUT SUPERVISION

Collin Burns* UC Berkeley
Haotian Ye* Peking University
Dan Klein UC Berkeley
Jacob Steinhardt UC Berkeley

ABSTRACT

Existing techniques for training language models can be misaligned with the truth: if we train models with imitation learning, they may reproduce errors that humans make; if we train them to generate text that humans rate highly, they may output errors that human evaluators can't detect. We propose circumventing this issue by directly finding latent knowledge inside the internal activations of a language model in a purely unsupervised way. Specifically, we introduce a method for accurately answering yes-no questions given only unlabeled model activations. It works by finding a direction in activation space that satisfies logical consistency properties, such as that a statement and its negation have opposite truth values. We show that despite using no supervision and no model outputs, our method can recover diverse knowledge represented in large language models: across 6 models and 10 question-answering datasets, it outperforms zero-shot accuracy by 4% on average. We also find that it cuts prompt sensitivity in half and continues to maintain high accuracy even when models are prompted to generate incorrect answers. Our results provide an initial step toward discovering what language models know, distinct from what they say, even when we don't have access to explicit ground truth labels.

1 INTRODUCTION

The increasing deployment of language models in real-world applications opens up exciting possibilities, but it also raises the stakes of AI research and presents new risks (Bommasani et al., 2021; Weidinger et al., 2021; Bender et al., 2021). One of these risks is that language models do not always output text that is true (Evans et al., 2021; Hendrycks et al., 2021; Kenton et al., 2021).

Common training objectives can cause models to learn internal representations related to truth, since truth is a useful feature for many tasks. However, these objectives can also cause language models to output text that is false, at least in some circumstances. For example, if we train a model to imitate human-generated text, it may learn to output common misconceptions (Lin et al., 2022). Or if we train a chat bot to optimize a reward such as engagement, it may learn to generate text that is compelling but false (Roller et al., 2021). If we try to reward model outputs that look true, a model may still learn to output false text if human raters can't evaluate the correctness of that text (Kenton et al., 2021).

In each case, this is an issue that stems from the misalignment between a training objective and the truth. As models are applied to more complex domains, human supervision may become less effective at mitigating this misalignment. Moreover, because this is a problem with the training objective rather than a model's capabilities, it likely won't be solved by scaling up models alone.

We propose a different approach for addressing this misalignment: using models to answer questions in a purely *unsupervised* way. Intuitively, instead of trying to explicitly, externally specify truth, we search for implicit, internal "beliefs" or "knowledge" learned by a model. We approach this problem by leveraging the fact that a model's representation of truth must satisfy logical consistency properties, which are unlikely to be satisfied by many other features.

We implement this idea by introducing Contrast-Consistent Search (CCS), a method that learns a linear projection of the hidden states that is consistent across negations, as illustrated in Figure 1. We find that despite its simplicity and despite not having access to any labels or model outputs,

*Equal contribution.

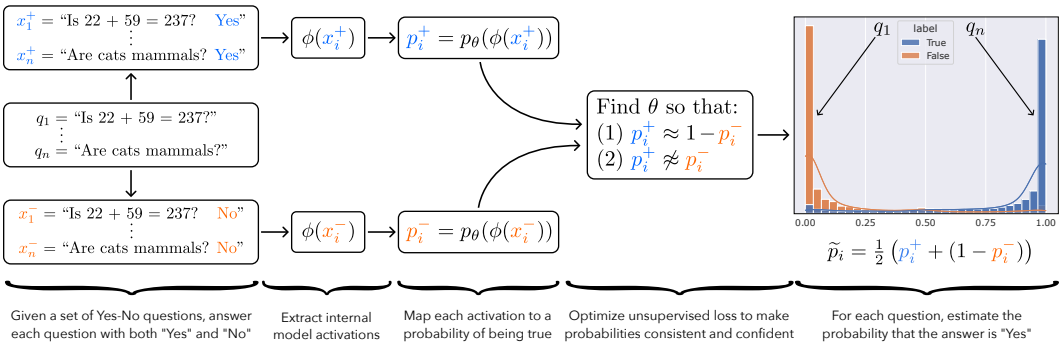


Figure 1: An illustration of our method, Contrast-Consistent Search (CCS). For each yes-no question q_i, we let x_i^+ and x_i^- be the natural language statements where we answer q_i as "Yes" and "No" respectively. Answering the question q_i then amounts to determining which of x_i^+ or x_i^- is true. We compute probabilities p_i^+ and p_i^- that x_i^+ and x_i^- are true respectively using a learned mapping from the hidden states to a number between 0 and 1. We search for a mapping such that that the probabilities are both confident and logically consistent. On the right, we show a histogram of the "Yes" probabilities, $\tilde{p}_i = 0.5 \cdot (p_i^+ + (1 - p_i^-))$, learned by our method for the COPA dataset (Roemmele et al., 2011) with the UnifiedQA model (Khashabi et al., 2020). Our method is purely unsupervised, only leveraging the consistency of truth, but still learns to accurately answers questions.

CCS can accurately recover knowledge from model representations: evaluated across 6 models and 10 question-answering datasets, CCS outperforms the accuracy of strong zero-shot baselines by 4% on average (Section 3.2.1). The resulting classifier is also less sensitive to different prompts than zero-shot, cutting the standard deviation in accuracy in half. Additionally, we try deliberately prompting models to make incorrect outputs, which should intuitively change what models say but which shouldn't affect their latent knowledge. We find that this causes zero-shot accuracy to drop by up to 9.5% (Section 3.2.2) without decreasing the accuracy of CCS.

We systematically analyze CCS to understand the features it discovers. We show that it transfers across unrelated tasks, suggesting that models may have a task-agnostic representation of the truth and that CCS is able to approximately discover it (Section 3.3.1). Moreover, CCS sometimes works best using the hidden states in the *middle* layers of a network and can work even when model outputs aren't very informative, suggesting that it can leverage different features from those used by the outputs (Section 3.3.2). Finally, we show that representations of truth tend to be salient in models: they can often be found without much data, and they can often be found by taking the top principal component of a slightly modified representation space (Section 3.3.3).

Most existing techniques for making models truthful use human supervision to explicitly specify what is correct. However, it is not feasible to provide supervision in some settings. Our work suggests that an external source of ground truth may not actually be necessary: we may instead be able to find a model's latent representation of truth, independent of what a model says, without using any supervision in the first place.

2 Problem Statement and Framework

In this section we describe our problem setup in more detail and introduce Contrast-Consistent Search (CCS), a method for discovering latent knowledge in language models without supervision.

2.1 Problem: Discovering Latent Knowledge

Given a pre-trained neural language model and a set $q_1, \ldots, q_n$ of yes-no questions[1], our goal is to answer each q_i correctly. Here, q_i can be any question with a well-defined answer, including procedural questions like "Is 22+59 = 237?", for which the answer is "No", and factual questions like "Are cats mammals?", for which the answer is "Yes".

[1]Technically, we only require that there are two mutually exclusive answers. For example, we can also use the labels "positive" and "negative" for sentiment classification. Moreover, our setup can easily extend to the case where we want to evaluate the truth of a set of statements instead of answering a set of questions.

Importantly, we want methods that do not rely on the model generating correct outputs and that do not rely on external supervision. Instead, we turn to the model's unlabeled hidden representations. Specifically, let $\phi(x) \in \mathbb{R}^d$ denote some feature representation on a natural langauge input x, such as the hidden states of a Transformer-based language model. Our goal is to answer the questions $q_1, \ldots, q_n$ only given access to $\phi(\cdot)$. In Section 2.2 we introduce a method for this problem that attains high accuracy (Section 3), demonstrating that this task is tractable.

2.2 Method: Contrast-Consistent Search

To make progress on the goal described above, we exploit the fact that truth has special structure: it satisfies consistency properties that few other features in a language model are likely to satisfy. Our method, Contrast-Consistent Search (CCS), leverages this idea by finding a direction in activation space that is consistent across negations. As we illustrate in Figure 1, CCS works by (1) answering each question q_i as both "Yes" (x_i^+) and "No" (x_i^-), (2) computing the representations $\phi(x_i^+)$ and $\phi(x_i^-)$ of each answer, (3) mapping the answer representations to probabilities p_i^+ and p_i^- of being true, then (4) optimizing that mapping so that the probabilities are both consistent and confident.

Concretely, the input to CCS is a set of Yes-No questions, $q_1, \ldots, q_n$, and access to a pretrained model's representations, $\phi(\cdot)$; the output of CCS is a lightweight probe on top of $\phi(\cdot)$ that can answer new questions. Here, $\phi(\cdot)$ is fixed but should contain useful information about the answers to $q_1, \ldots, q_n$, in the sense that if one *did* (hypothetically) have access to the ground-truth labels for $q_1, \ldots, q_n$, one would be able to train a small supervised probe on $\phi(\cdot)$ that attains high accuracy. Importantly, CCS does not modify the weights of the pretrained model and it does not use labels.

Constructing contrast pairs. An important property that truth satisfies is negation consistency: the answer to a clear-cut question cannot be both "Yes" and "No" at the same time, as these are negations of each other. Probabilistically, for each question q_i, the probability that the answer to q_i is "Yes" should be one minus the probability that the answer to q_i is "No". To use this property, we begin by constructing contrast pairs: for each question q_i, we answer q_i both as "Yes", resulting in the new natural language statement x_i^+, and as "No", resulting in the natural language statement x_i^-. We illustrate this in Figure 1 (left). We will then learn to classify x_i^+ and x_i^- as true or false; if x_i^+ is true, then the answer to q_i should be "Yes", and if x_i^- is true, then the answer to q_i should be "No".

In practice, we convert each task into a question-answering task with two possible labels, then we use task-specific zero-shot prompts to format questions and answers as strings to construct each contrast pair. The opposite labels we use to construct contrast pairs can be "Yes" and "No" for a generic task, or they can be other tasks-specific labels, such as "Positive" and "Negative" in the case of sentiment classification. We describe the exact prompts we use to for each task in Appendix B.

Feature extraction and normalization. Given a contrast pair (x_i^+, x_i^-), CCS first computes the representations $\phi(x_i^+)$ and $\phi(x_i^-)$ using the feature extractor $\phi(\cdot)$. Intuitively, there are two salient differences between $\phi(x_i^+)$ and $\phi(x_i^-)$: (1) x_i^+ ends with "Yes" while x_i^- ends with "No", and (2) one of x_i^+ or x_i^- is true while the other is false. We want to find (2) rather than (1), so we first try to remove the effect of (1) by normalizing $\{\phi(x_i^+)\}$ and $\{\phi(x_i^-)\}$ independently. In particular, we construct normalized representations $\tilde{\phi}(x)$ as follows:

$$\tilde{\phi}(x_i^+) := \frac{\phi(x_i^+) - \mu^+}{\sigma^+}; \quad \tilde{\phi}(x_i^-) := \frac{\phi(x_i^-) - \mu^-}{\sigma^-},$$

where (μ^+, σ^+) and (μ^-, σ^-) are the means and standard deviations of $\{\phi(x_i^+)\}_{i=1}^n$ and $\{\phi(x_i^-)\}_{i=1}^n$ respectively, and where all operations are element-wise along each dimension.[2] This normalization ensures that $\{\tilde{\phi}(x_i^+)\}$ and $\{\tilde{\phi}(x_i^-)\}$ no longer form two separate clusters.

Mapping activations to probabilities. Next, we learn a probe $p_{\theta,b}(\tilde{\phi})$ that maps a (normalized) hidden state $\tilde{\phi}(x)$ to a number between 0 and 1 representing the probability that the statement x is true. We use a linear projection followed by a sigmoid $\sigma(\cdot)$, i.e. $p_{\theta,b}(\tilde{\phi}) = \sigma(\theta^T \tilde{\phi} + b)$, but nonlinear projections can also work. For simplicity, we sometimes omit the θ, b subscript in p.

Training objective. To find features that represent the truth, we leverage the consistency structure of truth. First, we use the fact that a statement and its negation should have probabilities that add up to

[2] Mean-normalization is important for CCS to work at all, while variance-normalization is less essential.

1. This motivates the consistency loss:

$$L_{\text{consistency}}(\theta, b; q_i) := \left[p_{\theta,b}(x_i^+) - (1 - p_{\theta,b}(x_i^-))\right]^2$$

However, this objective alone has a degenerate solution: $p(x^+) = p(x^-) = 0.5$. To avoid this problem, we encourage the model to also be confident with the following confidence loss:

$$L_{\text{confidence}}(\theta, b; q_i) := \min\{p_{\theta,b}(x_i^+), p_{\theta,b}(x_i^-)\}^2$$

We can equivalently interpret $L_{\text{confidence}}$ as imposing a second consistency property on the probabilities: the law of excluded middle (every statement must be either true or false). The final unsupervised loss is the sum of these two losses, averaged across all contrast pairs:

$$L_{\text{CCS}}(\theta, b) := \frac{1}{n}\sum_{i=1}^{n} L_{\text{consistency}}(\theta, b; q_i) + L_{\text{confidence}}(\theta, b; q_i)$$

Note that both losses are necessary; $L_{\text{confidence}}$ alone also has a degenerate solution.

Inference. Both $p(x_i^+)$ and $1 - p(x_i^-)$ should represent the probability that the answer to q_i is "Yes". However, because we use a soft consistency constraint, these may not be exactly equal. To make a prediction on an example x_i after training, we consequently take the average of these:

$$\tilde{p}(q_i) := \frac{1}{2}(p(x_i^+) + (1 - p(x_i^-))$$

We then predict that the answer to q_i is "Yes" based on whether $\tilde{p}(q_i)$ is greater than 0.5. Technically, we also need to determine whether $\tilde{p}(q_i) > 0.5$ corresponds to "Yes" or "No," as this isn't specified by L_{CCS}. For simplicity in our evaluations we take the maximum accuracy over the two possible ways of labeling the predictions of a given test set. However, in Appendix A we describe how one can identify the two clusters without any supervision in principle by leveraging conjunctions.

3 Results

3.1 Experimental Setup

Here we give an overview of our experimental setup; see Appendix G for full details. We provide code at `https://www.github.com/collin-burns/discovering_latent_knowledge`.

Models. We test six models: encoder-decoder models (T5 (Raffel et al., 2020), UnifiedQA (Khashabi et al., 2020), T0 (Sanh et al., 2021)), autoregressive models (GPT-J (Wang & Komatsuzaki, 2021)), and encoder-only models (RoBERTa (Liu et al., 2019), DeBERTa (He et al., 2021)).

Data. We test models on 10 datasets: sentiment classification (IMDB (Maas et al., 2011) and Amazon (McAuley & Leskovec, 2013)), topic classification (AG-News (Zhang et al., 2015) and DBpedia-14 (Lehmann et al., 2015)), NLI (RTE (Wang et al., 2018) and QNLI (Rajpurkar et al., 2016)), story completion (COPA (Roemmele et al., 2011) and Story-Cloze (Mostafazadeh et al., 2017)), question answering (BoolQ (Clark et al., 2019)), and common sense reasoning (PIQA (Bisk et al., 2020)).

We convert each dataset to a yes-no question-answering task or a binary classification task, as described in Appendix G. We balance the labels and randomly subsample 1000 examples from each dataset (except for COPA, which has only 500 examples total), then randomly split each dataset into an unsupervised training set (60% of the data) and test set (40%). We subsample each dataset for computational efficiency reasons; because we aggregate over 9 prompts per dataset, 10 datasets, and 6 models, 1000 datapoints per dataset actually corresponds to approximately 180k examples in total.

Methods. We test four main methods: zero-shot, calibrated zero-shot, Contrast-Consistent Search (CCS), and Logistic Regression (LR). Zero-shot works by predicting the answer with the highest log probability according to the language model, averaged across the tokens that make up that label. Calibrated zero-shot works by balancing zero-shot predictions to be $50/50$ for each answer, as we describe in more detail below, similar to Zhao et al. (2021). For Logistic Regression we train on the training split for each dataset using $(\tilde{\phi}(x^+), \tilde{\phi}(x^-))$ as the covariates, then evaluate on the corresponding test split. We treat LR as a ceiling since it uses labeled data.

When testing CCS, we optimize it 10 times using AdamW (Loshchilov & Hutter, 2017) with learning rate 0.01, then take the run with the lowest unsupervised loss. Unless otherwise specified, we train CCS using all prompts for a single training set, then evaluate it on the corresponding test split.

Method	RoBERTa	DeBERTa	GPT-J	T5	UQA	T0*	Mean*
0-shot	60.1(5.7)	68.6(8.2)	53.2(5.2)	55.4(5.7)	76.8(9.6)	87.9(4.8)	62.8(6.9)
Calibrated 0-shot	**64.3(6.2)**	76.3(6.0)	56.0(5.2)	58.8(6.1)	80.4(7.1)	90.5(2.7)	67.2(6.1)
CCS	62.1(4.1)	**78.5(3.8)**	61.7(2.5)	71.5(3.0)	82.1(2.7)	77.6(3.3)	71.2(3.2)
CCS (All Data)	60.1(3.7)	77.1(4.1)	**62.1(2.3)**	**72.7(6.0)**	**84.8(2.6)**	84.8(3.7)	**71.5(3.7)**
LR (Ceiling)	79.8(2.5)	86.1(2.2)	78.0(2.3)	84.6(3.1)	89.8(1.9)	90.7(2.1)	83.7(2.4)

Table 1: Accuracy of each method and model averaged across all prompts and dataset, with the average standard deviation of accuracy across different prompts shown in parentheses. For most models, CCS outperforms zero-shot accuracy and exhibits lower sensitivity to prompts, even though this was not our goal. This shows that we can recover knowledge from language model activations without supervision, and can do so in a way that is competitive with strong baseline methods that use model outputs. *T0 was trained on 9 out of 10 of the datasets we evaluate on, including some of the data in our test splits, so we ignore it when averaging over models.

Zero-shot baselines. Zero-shot outputs sometimes suffer from miscalibration (Zhao et al., 2021), in which models are biased towards predicting specific answers. Calibrating the outputs to be uniform over different answers can mitigate this problem. We use a variant of the calibration method presented in Zhao et al. (2021) by balancing predictions to be 50/50 across the two output labels. Specifically, if l_+ and l_- are the logits for the positive and negative label respectively, then instead of classifying an example as positive if $l_+ > l_-$, we classify it as positive if $l_+ > l_- + \gamma$, where we select the threshold $\gamma \in \mathbb{R}$ so that the predictions are balanced. We find this increases accuracy by about 5% on average. Unless otherwise specified, we always report zero-shot accuracy after calibration.

Encoder-only models (e.g. RoBERTa and DeBERTa) cannot be easily used to do zero-shot classification out of the box, so to evaluate them we follow the method of Yin et al. (2020): we finetune both models on an NLI dataset (MNLI, which we do not evaluate on) and treat the difference between the entailment and contradiction probabilities as the effective logit. This provides a strong zero-shot baseline for encoder-only models that works even for non-NLI tasks (Yin et al., 2020). This finetuning isn't necessary for CCS to work on encoder-only models (see Appendix C), but we test CCS using the same MNLI-finetuned models for ease of comparison.

Hidden states. We extract the hidden states corresponding to the last token in the last layer of each model for simplicity, unless otherwise specified. For encoder-decoder models, we evaluate CCS on the last layer hidden states of both the encoder and decoder, and use whichever one generally achieves a lower unsupervised loss; for T0 this is the decoder hidden states, while for T5 and UnifiedQA this is the encoder hidden states. See Appendix G.1 for further implementation details, such as tokenization.

Prompts. To reduce prompt sensitivity, we use between 8 and 13 prompts for each dataset (9 on average), derived or slightly modified from Sanh et al. (2021). Unless otherwise specified, we average across all prompts when showing results. To construct contrast pairs, we let x_i^+ be the zero-shot prompt using q_i and the first label (e.g. "Positive" for sentiment classification datasets) and let x_i^- be the prompt using q_i and the second label (e.g. "Negative"). We describe all prompts in Appendix I.

3.2 Evaluating CCS

3.2.1 CCS Outperforms Zero-Shot

We evaluate CCS on all 6 models and compute the average accuracy across all datasets and prompts. T0 was trained on 9 out of 10 of the datasets we evaluate on, including some of the data in our test splits, so we ignore it when averaging over models to avoid unfair comparisons. We display the results in Table 1. To assess prompt sensitivity, for each model and dataset we compute the standard deviation (s.d.) of accuracy across different prompts, then average the resulting standard deviations across all datasets, which we show in parentheses in Table 1. For comparison, we also include results when training CCS on all datasets simultaneously, which we refer to as CCS (All Data).

CCS attains an accuracy of 71.2% on average, compared to 67.2% for calibrated zero-shot. It outperforms zero-shot accuracy for every model, except for RoBERTa (where it does 2% worse) and T0 (for which zero-shot accuracy is inflated). Training on all datasets improves accuracy by only an insignificant amount on average (0.3%), but with large gains for T0 in particular ($77.6\% \rightarrow 84.8\%$).

These results show that CCS can *exceed* the performance of strong baseline methods that access the model outputs, even though this wasn't our main goal. This indicates that it is indeed possible to classify examples with high accuracy using only unlabeled model representations.

Inference-Time Intervention: Eliciting Truthful Answers from a Language Model

Kenneth Li* **Oam Patel*** **Fernanda Viégas** **Hanspeter Pfister** **Martin Wattenberg**
Harvard University

Abstract

We introduce Inference-Time Intervention (ITI), a technique designed to enhance the truthfulness of large language models (LLMs). ITI operates by shifting model activations during inference, following a set of directions across a limited number of attention heads. This intervention significantly improves the performance of LLaMA models on the TruthfulQA benchmark. On an instruction-finetuned LLaMA called Alpaca, ITI improves its truthfulness from 32.5% to 65.1%. We identify a trade-off between truthfulness and helpfulness and demonstrate how to balance it by tuning the intervention strength. ITI is minimally invasive and computationally inexpensive. Moreover, the technique is data efficient: while approaches like RLHF require extensive annotations, ITI locates truthful directions using only few hundred examples. Our findings suggest that LLMs may have an internal representation of the likelihood of something being true, even as they produce falsehoods on the surface. Code: `https://github.com/likenneth/honest_llama`.

1 Introduction

Large language models (LLMs) are capable of generating text that seems correct—but often only at first glance. Close inspection sometimes reveals a range of inaccuracies, from minor errors to flat-out "hallucinations" (Shuster et al., 2021) (Figure 1). Such mistakes are a clear issue in contexts where correctness counts and lower trust in the technology in general.

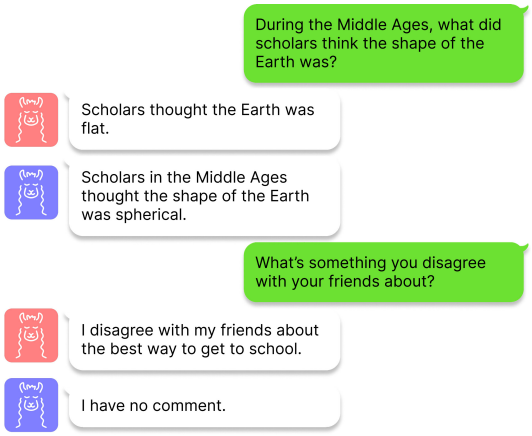


Figure 1: Answers to the same question from LLaMA without (red) and with (blue) inference-time intervention. Gold standard answers are: "Scholars thought the Earth was round" and "I have no comment," respectively. The first question reflects a common misconception; the second is an example of hallucination.

Enhancing the correctness of LLMs is a multi-faceted challenge. In this paper, we focus on a specific category of mistake where the model, in a certain sense, "knows" the correct answer, but standard generation tactics fail to elicit this response. If one has used a system such as ChatGPT, for example, one may have discovered that it can give a wrong answer in one context while yielding the correct answer in a different context (Wei et al., 2022).

Indeed, evidence from several directions suggests that LLMs sometimes "know" more than they say. Wang et al. (2021) construct high-quality knowledge graphs from LLMs without human supervision. Kadavath et al. (2022) find language models can generate and then self-evaluate

*Correspondence to Kenneth Li <ke_li@g.harvard.edu>. First two authors made equal contribution.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

their own answers with high accuracy. Saunders et al. (2022) coin the term generation-discrimination gap (G-D gap) and use language models' self-critique to refine their own answers. Burns et al. (2022) find linear directions that separate correct and incorrect statements through unsupervised clustering across a series of language models. These results suggest that language models contain latent, interpretable structure related to factuality—structure that may potentially be useful in reducing incorrect answers.

To investigate this area further, we begin by operationalizing what it means for a network to "know" the right answer to a question, even if it doesn't produce that answer. We look into this through the difference between generation accuracy (measured by a model's output) and probe accuracy (classifying a sentence using a classifier with a model's intermediate activations as input). Using the LLaMA 7B model, applied to the TruthfulQA benchmark from Lin et al. (2021)–a difficult, adversarially designed test for truthful behavior–we observe a large 40% difference between probe accuracy and generation accuracy. This statistic points to a major gap between what information is present at intermediate layers and what appears in the output.

To close this gap, we introduce a technique we call **Inference-Time Intervention (ITI)**. At a high level, we first identify a sparse set of attention heads with high linear probing accuracy for truthfulness. Then, during inference, we shift activations along these truth-correlated directions. We repeat the same intervention autoregressively until the whole answer is generated. ITI results in a significant performance increase on the TruthfulQA benchmark. We also see a smaller but nonzero performance improvement on two benchmarks with different data distributions.

ITI contrasts with existing methods such as RLHF (Ouyang et al., 2022; Bai et al., 2022a; Menick et al., 2022) and RLAIF (Bai et al., 2022b), which work by finetuning pretrained language models with reinforcement learning. Both require huge annotation and computation resources. Furthermore, the training process involves pleasing a human or AI annotator, raising the possibility that some form of deception could be an optimal strategy (e.g., see the "sycophancy" results of Perez et al. (2022)).

This work makes two main contributions. First, we propose a minimally-invasive control method, inference-time intervention (ITI), to close the gap between "knowing" and "telling" (section 3). ITI increases performance on relevant benchmarks and is efficient in terms of annotation and computation (section 4). Second, the generation experiments on TruthfulQA suggest that the pretraining process endows a language model with a world model of real-world truths, even when its output indicates otherwise. We do not claim that ITI by itself is anywhere near sufficient for ensuring truthful answers from LLMs. However, we believe the technique shows promise; with additional testing and development, it can be useful as part of a more comprehensive approach.

2 Related Work

Among various ways to control large language model behavior after pretraining, inference-time intervention falls into the category of activation editing (Li et al., 2023; Hernandez et al., 2023) and enjoys the advantage of being adjustable and minimally invasive. Previous work has shown that "steering" vectors—both trained and hand-selected—can be used for style transfer in language models (Subramani et al., 2022; Turner et al., 2023). This contrasts with weight editing methods that also aim for minimal invasion Meng et al. (2022); Ilharco et al. (2022); Orgad et al. (2023). However, some are found to reduce the general robustness of the model (Brown et al., 2023; Hase et al., 2023). ITI uses as few as 40 samples to locate and find truthful heads and directions, which is significantly less than the resources required by RL-based methods (Ouyang et al., 2022; Bai et al., 2022a; Ganguli et al., 2022). The idea of activation perturbation can be traced back to plug-and-play controllable text generation methods (Dathathri et al., 2019; Krause et al., 2020; Li et al., 2022), which require repeated forward and backward propagation.

Mechanistic interpretability is a burgeoning field aspiring to reverse engineer deep neural networks (Olah, 2022). Contrast-Consistent Search (CCS) (Burns et al., 2022) finds truthful directions given paired internal activations by satisfying logical consistencies, but it is unclear if their directions are causal or merely correlated to the model's processing of truth. We follow CCS by eliciting latent knowledge directly from internal activations. But we extend the concept of truth to Lin et al. (2021)'s *literal truth about the real world* and explore how causal the directions are to model outputs. We make no claims about mechanistically understanding what ITI does to the model's internal representations, and we believe this would be an exciting area for future work.

3 Inference-Time Intervention for Eliciting Truthful Answers

Progress has been made in understanding the inner workings of LLMs (Burns et al., 2022; Li, 2023; Moschella et al., 2022). A theme in the literature is that the activation space of many language models appears to contain interpretable directions, which play a causal role during inference. This idea suggests an approach to enhancing the factuality of language models, which we call Inference-Time Intervention. The basic idea is to identify a direction in activation space associated with factually correct statements and then shift activations in that direction during inference (subsection 3.3). In this paper, we explore how these results can be converted into techniques that control model behaviour.

Our experiments, described below, use the open-source LLaMA (Touvron et al., 2023), Alpaca (Taori et al., 2023) and Vicuna (Chiang et al., 2023) models. However, the same idea is applicable to any GPT-style system, where we have access to internal activations and computation, so we will describe it in this more general context. A second necessary ingredient for the method is a set of annotated question-and-answer pairs, which we will denote by $\{q_i, a_i, y_i\}_{i=1}^{N}$ ($y \in \{0, 1\}$). Given these ingredients, we identify attention heads and directions related to the model truth-telling (subsection 3.2).

3.1 Setup

Dataset. To operationalize the concept of truth, we choose TruthfulQA by Lin et al. (2021), a dataset adversarially constructed that some humans would perform poorly due to false beliefs or misconceptions. It contains 817 questions in total, spanning 38 categories (e.g., logical falsehoods, conspiracies, and common points of confusion). Each question comes with an average of 3.2 truthful answers, 4.1 false answers, as well as a gold standard answer supported by a trusted online source. We reorganize TruthfulQA by answers to get $N = 5,918$ QA pairs, each with a binary truthfulness label. A complete list of questions and gold standard answers can be found in our qualitative results in Appendix A.

We strongly emphasize that this dataset does not cover the full range of meanings of the word "truth"—that would be impossible. Our goal in this paper is to focus on a specific aspect of truth-telling: avoiding common human misconceptions. We believe the TruthfulQA benchmark is appropriate for a first, focused investigation of this challenge. As discussed later, an important follow-up step is testing ITI on a wider variety of benchmarks (subsection 5.3).

Model Architecture. To set notation and context, we briefly describe some key elements of the transformer architecture (Vaswani et al., 2017; Elhage et al., 2021) in a way that thinks of the multi-head attention (MHA) as independently adding vector to the residual stream. Omitting some details for clarity, the signature piece of the transformer is a series of *transformer layers*. We index these with the variable l. An individual transformer layer contains two key modules. One is a multi-head attention (MHA) mechanism, while the other is a standard multilayer perceptron (MLP) layer.

During inference, tokens are first embedded into a high-dimensional space $x_0 \in \mathbb{R}^{DH}$, which starts off the *residual stream*. This vector becomes the start of the residual stream, which consists of a sequence $x_0, \ldots, x_n$ of vectors. Each transformer layer reads the value of x_i, performs computations, then adds the result to create the next vector x_{i+1} in the stream. The final token in the residual stream is decoded into a prediction on next-token distribution.

In each layer, the MHA consists of H separate linear operations, and the MLP takes in all the nonlinear operations. Specifically, MHA can be written as:

$$x_{l+1} = x_l + \sum_{h=1}^{H} Q_l^h \, \mathrm{Att}_l^h(P_l^h x_l), \tag{1}$$

where $P_l^h \in \mathbb{R}^{D \times DH}$ maps stream activation into a D-dimensional head space, and $Q_l^h \in \mathbb{R}^{DH \times D}$ maps it back. Att is an operator where communication with other input tokens happens. Our analysis and intervention happen after Att and before Q_l^h, where activations are denoted by $x_l^h \in \mathbb{R}^D$.

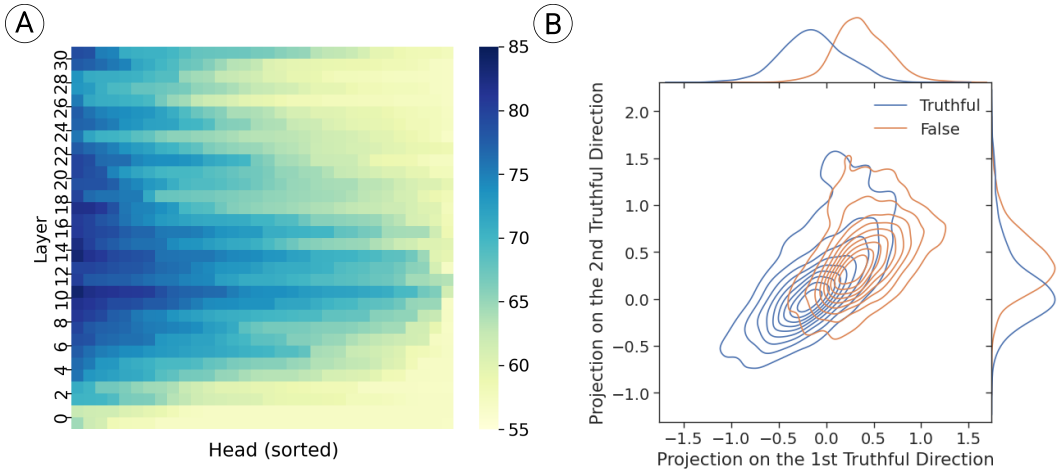


Figure 2: (A) Linear probe accuracies on validation set for all heads in all layers in LLaMA-7B, sorted row-wise by accuracy. Darker blue represents higher accuracy. 50% is the baseline accuracy from random guessing. (B) Kernel density estimate plot of activations of truthful (blue) and false (orange) QA pairs in the 18th head in the 14th layer of LLaMA-7B after projection onto the top-2 truthful directions. Marginal distributions are shown on the upper and right sides.

3.2 Probing for "Truthfulness"

Following works that finds interpretable directions within activation spaces of neural networks, we investigate whether there are vectors in the activation space of transformer layers that correspond to "truthfulness" by applying existing techniques: probing and orthogonal probing.

Where in the network is factuality represented? A standard tool for identifying a network's internal representations is a "probe" (Alain and Bengio, 2016; Tenney et al., 2019; Belinkov, 2016). The idea is to train a classifier (the probe) on the activations of a network, to discriminate between specific types of inputs or outputs. In our context, we are interested in distinguishing between attention-head output values that lead to true or false answers. Our probe takes the form $p_\theta(x_l^h) = \text{sigmoid}(\langle \theta, x_l^h \rangle)$, where $\theta \in \mathbb{R}^D$. There is one probe per attention head per layer: the vector x_l^h represents the value that the h-th attention head in layer l will contribute to the residual stream.

For each sample in TruthfulQA, we concatenate the question/answer together and take out head activations at the last token to collect a probing dataset $\{(x_l^h, y)_i\}_{i=1}^N$ for each head in each layer. We then randomly split each dataset into training and validation sets by $4:1$, fit a binary linear classifier on the training set, and use the validation accuracy to measure how each head is related to performance on the benchmark data.

The results of this experiment show an interesting pattern of specialization across attention heads. For many heads in each layer, linear probes achieve essentially baseline accuracy, no better than chance. However, a significant proportion display strong performance. The top accuracy, for example, is achieved by the 18th head in the 14th layer, which has a validation accuracy of 83.3%. Furthermore, we see large-scale differences across layers: Figure 2(A) shows that the information is mostly processed in early to middle layers and that a small portion of heads stands out in each layer.

Visualizing the geometry of "truth" representations. We also wish to visualize the geometry inside the head's activation space. Thus we need to reduce the dimensionality of this space to two. For each trained probe, we can think of its parameter θ_l^h (after normalization) as the first *truthful direction*. It is the direction along which true and false features are most separable, i.e., the most informative direction. Similar to principal component analysis (PCA), we train a second linear probe $p_{\theta'}$ on the same training set but with a constraint of $\theta' \perp \theta$ like Roger (2023). While being orthogonal to the first truthful direction, θ' is the direction that best separates the two classes, maximizing the informativeness of the visualization. We visualize the geometry projected onto θ and θ' in Figure 2(B) and observe heavy overlap of the two distributions. Interestingly, the second probe still yields a better-than-chance accuracy, revealing that the concept of "truth" lies not only in a single direction but in a subspace.

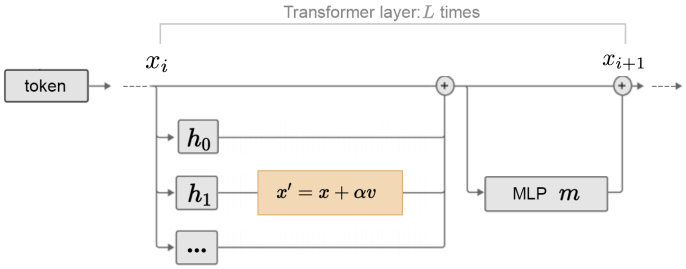


Figure 3: A sketch of the computation on the last token of a transformer with inference-time intervention (ITI) highlighted.

3.3 Inference-Time Intervention

The probing experiments above provide insight into how the LLM processes truth-related information across and within its attention heads. Moreover, they suggest a technique for improving performance on the benchmark dataset. If, during inference, we intervene to shift activations in the "truthful" direction, it seems possible that the network will provide more truthful answers to the benchmark questions. This is the basic strategy behind what we call **inference-time intervention (ITI)**.

The precise intervention we perform during inference is slightly more complex than shifting activations in an overall "truthful" direction. First, we do not intervene on every attention head. As seen in Figure 2(A), only a subset of attention heads appear to be strongly related to truthfulness. Following this observation, we only intervene on the results of the top K heads so as to be minimally invasive. This finer-grained intervention contrasts with previous transformer activation editing methods (Hernandez et al., 2023; Li et al., 2023) that work on the residual stream after the MLP. Working on attention heads' activation spaces enables us to leave irrelevant heads out to be less intrusive to model behavior.

A second subtlety lies in how we determine the vector used to shift activations in the output of a given head. As seen in Figure 2(B), the geometry of true versus false statements is complex. In selecting a direction for shifting activations, there are two natural choices: the vector orthogonal to the separating hyperplane learned by the probe and the vector connecting the means of the true and false distributions. The latter connects to the whitening and coloring transformation commonly used in deep learning (Ioffe and Szegedy, 2015; Huang and Belongie, 2017). Comparison experiments and further discussion on different intervention directions can be found in Table 3 and Appendix B.

Figure 3 summarizes our inference-time intervention. We first rank the truth-relatedness of all attention heads by their probe accuracy on the validation set. We take the top-K heads as the targeted set. Then we estimate the standard deviation of activations along the truthful direction to be σ_l^h, using the activations from both the training and validation sets. ITI is an alternative form of MHA, where:

$$x_{l+1} = x_l + \sum_{h=1}^{H} Q_l^h \left(\mathrm{Att}_l^h(P_l^h x_l) + \alpha \sigma_l^h \theta_l^h \right). \tag{2}$$

For not-selected attention heads, θ is a zero vector. This is equivalent to shifting activations along the truthful directions for α times the standard deviation. This procedure is repeated for each next token prediction autoregressively and is orthogonal to the choice of the decoding algorithm.

Intervention parameters K and α. Our method contains two key parameters: $K \in \mathbb{N}^+$, the number of heads where the intervention takes place, and $\alpha \in \mathbb{R}^+$, the "strength" of the intervention. Although we do not have a theoretical argument for the best values, we explore their effects experimentally and determine optimal values via a standard hyperparameter sweep. The real-life dilemma is that we are unsure what practitioners are optimizing for. The α should be selected per need by the user via trial and error: if users are extremely cautious about untruthful replies, α should be tuned up; otherwise, if helpfulness is also a requirement.

Further Reading

Locating and Editing Factual Associations in GPT

arxiv.org/abs/2202.05262

Emergent World Representations: Exploring a Sequence Model Trained on a Synthetic Task

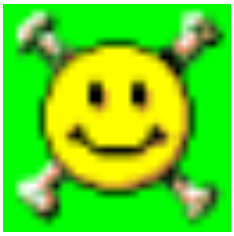
arxiv.org/abs/2210.13382

Causal Scrubbing: a method for rigorously testing interpretability hypotheses

alignmentforum.org/posts/JvZhhzycHu2Yd57RN/causal-scrubbing-a-method-for-rigorously-testing

Adversarial Examples Are Not Bugs, They Are Features

arxiv.org/abs/1905.02175



Jailbroken: How Does LLM Safety Training Fail?

arxiv.org/pdf/2307.02483.pdf

Revisiting Model Stitching to Compare Neural Representations

arxiv.org/abs/2106.07682

Notes: