Disentangling the Diversity of Truth in Large Language Models

Anonymous ACL submission

Abstract

Large Language Models (LLMs) can often produce factually incorrect statements, and they offer no citations or internal reasoning. We believe the safe deployment of LLMs requires a deeper understanding of how truth is represented within these models. In this paper, we study the internal representation of truth in LLMs and introduce a taxonomy of truth types, including arithmetic, logical, symbolic, and consensus-based. We use linear probes to identify where in the model different truth types become linearly decodable, and we apply control tasks to distinguish genuine encoding from superficial correlations. Our findings reveal that distinct truth types emerge at different layers. For instance, single-digit sums are encoded earlier than multi-digit ones, suggesting increasing abstraction across depth. To further interpret these internal representations, we train sparse autoencoders on hidden states, revealing human-interpretable features such as patterns like "[person] lived in [place]" or arithmetic involving specific digits. These results highlight structure and specialization in how truth is encoded across transformer layers and neurons. To support future work, we also release a tool for probing and visualizing internal representations across models and datasets.

1 Introduction

LLMs have achieved remarkable fluency across a wide range of tasks, yet they often produce statements that are factually incorrect, unverifiable, or internally inconsistent—and they offer no citation. This undermines the reliability and safety of LLMs in high-stakes settings such as education, law, and medicine (Bender et al., 2021). Despite their widespread deployment, we still lack a clear understanding of how factual information is represented within LLMs, and under what conditions that information can be reliably extracted or aligned.

This paper addresses the question, *How and where do LLMs internally represent truth?* While previous work has shown that some factual knowledge is linearly decodable from LLM hidden states (Hewitt and Liang, 2019; Marks and Tegmark, 2024), most studies treat truth as a binary concept—true or false—with little regard to type. However, not all truths are alike. Logical statements like "2 + 3 = 5" differ fundamentally from consensus-based facts like "Paris is the capital of France." We propose a novel taxonomy of truth types—including arithmetic, logical, symbolic and consensus—and use this structure to investigate how LLMs encode each type.

To do this, we apply linear probing to the internal representations of several language models—including BERT (Devlin et al., 2019), GPT-2 (Radford et al., 2019), and LLaMA 3.2 (Grattafiori et al., 2024)— and analyze which layers best support truth classification for each type. We use control tasks to isolate genuine semantic encoding from spurious correlations, and measure selectivity—the gap between true and shuffled-label probe accuracy—to assess whether a model truly encodes a given truth type. We find that simple truths (e.g., small arithmetic sums) are often encoded in earlier layers, while more complex or abstract truths emerge later.

To further interpret these findings, we train sparse autoencoders on hidden states across layers, revealing human-interpretable features such as patterns involving numbers, entities, and locations. This approach builds on work in mechanistic interpretability (Elhage et al., 2022; Cunningham et al., 2023), demonstrating that distinct neurons activate for different truth types, and sometimes even for specific truth templates.

Together, these results suggest that truth is not monolithic in LLMs: different types of truth are encoded in different layers and with varying degrees of abstraction and specialization. We release our

datasets, probing framework, and visualization tool to support future work on truth, interpretability, and LLM alignment.

2 Prior Literature

2.1 Linear Probing and Selectivity

The concept of probing has been used in computer vision (Alain and Bengio (2017)) and in LLMs for many years (Ettinger et al. (2016) and Shi et al. (2016)). However, Conneau et al. (2018) solidified the technique, taking a methodical look at the various concepts "crammed" into sentence embeddings. The authors established clear benchmarks, controls, and rigorously evaluated multiple architectures and objectives.

Probing tasks, the authors describe, are classification problems. "For example, one such task might require to categorize sentences by the tense of their main verb," they suggest. Sentence representations created at various points in the model under investigation are used as training data for the classifier. If the classifier succeeds, "it means that the pre-trained encoder is storing readable tense information into the embeddings it creates." The technique gives a view as to whether or not the concept under question—verb tense in the above case—is encoded at that particular point in the model.

The authors suggest using potentially multi-layer classifiers, rather than simple linear probes, which goes against later ideas by Hewitt and Liang (2019) who ask the important question, "[W]hen a probe achieves high accuracy on a linguistic task using a representation, can we conclude that the representation encodes linguistic structure, or has the probe just learned the task?"

Not necessarily, is the answer they find. Sometimes the probes simply learn the task they're given. The pair propose *control tasks*, such as shuffling the labels of the training data. If a probe succeeds on a control task, it suggests that it learned to do so itself; it has memorized based on word identity rather than actually extracted linguistic information.

The authors then simply define *selectivity* as the difference between linguistic task accuracy and control task accuracy. The ideal, of course, is to maximize this metric in order to show that the probe learned little, rather is extracting information encoded in the representation from the model in evaluation.

The results show that linear probes are most selective, whereas non-linear probes appear to memorize. Therefore, linear probes are best to properly evaluate models.

For a comprehensive survey of probing methods and their interpretability implications, see Belinkov (2022).

2.2 Sparse Autoencoders and Superposition

Sparse autoencoders have also emerged as a powerful interpretability method, most notably in recent work by Anthropic (Elhage et al. (2022)). They posit that, due to superposition of various features within LLMs' neurons, a hypothetical disentangled model exists, which fully separates out all features. To approximate this hypothetical model, they train a sparse autoencoder: a neural network designed to reconstruct its input while enforcing sparsity in a hidden layer. The sparsity constraint encourages the model to allocate distinct dimensions to distinct concepts; these are effectively features corresponding to semantically meaningful patterns.

2.3 Truth as Direction

Various researchers have used these techniques to demonstrate where and how truth, a hitherto single concept, is encoded. Marks and Tegmark (2024) use various techniques and controls to investigate whether large language models represent truth as a linear direction in activation space. Building on prior work on semantic directions in embeddings (Mikolov et al. (2013)) and truth probing (Li et al. (2024)), the authors evaluate whether such a direction generalizes across input types and model scales. Azaria and Mitchell (2023)'s work was a valuable addition to the literature in suggesting how this knowledge could be applied. They elegantly trained a classifier that uses an LLM's internal states to output the probability that a statement generated by the LLM is true.

3 Types of Truth

We categorize truth into four broad types—arithmetic, logical, symbolic, and consensus—inspired by both philosophical distinctions and practical considerations in how LLMs might internally encode different classes of factual knowledge. Each type is represented by targeted datasets described below.

3.1 Arithmetic Truths

Arithmetic truths consist of concrete, well-defined mathematical statements. These range in complex-

Truth Type	Subtype	True Example	False Example
Arithmetic	Summation, Single-Digit	9 + 9 = 18	4 + 1 = 14
	Summation, Multi-Digit	487 + 635 = 1122	114 + 157 = 263
	Multiplication, Single-Digit	2 * 4 = 8	2 * 2 = -3
	Multiplication, Multi-Digit	452 * 115 = 51980	424 * 257 = 108977
	Divisible by 5, Single-Digit	10 is divisible by 5	9 is divisible by 5
	Divisible by 5, Multi-Digit	550 is divisible by 5	902 is divisible by 5
Logical	Set membership, Single-Digit	6 is in the set {1, 3, 5, 6, 9}	0 is in the set {1, 2, 6, 7, 8}
	Set membership, Multi-Digit	57 is in the set {57, 251, 255, 320, 322}	724 is in the set {68, 81, 475, 504, 754}
	Inequality, Single-Digit	9 > 5	4 > 5
	Inequality, Multi-Digit	918 > 53	325 > 426
	Chained Inequality, Single-Digit	1 < 5 < 6	3 < 4 < 2
	Chained Inequality, Multi-Digit	215 < 273 < 554	486 < 706 < 542
	Parity, Single-Digit	6 is even	1 is even
	Parity, Multi-Digit	294 is even	175 is even
	Boolean AND	If A is true and B is true, then A and B is true.	If A is true and B is false, then A and B is true.
	Boolean OR	If A is true and B is false, then A or B is true.	If A is false and B is false, then A or B is true.
	Boolean NOT	If A is false, then NOT A is true.	If A is true, then NOT A is true.
Symbolic	Digit Count	The number 41903 has 5 digits	The number 3919 has 6 digits
Consensus	Factual	Somalia is a name of a country.	Panama City is a name of a country.
	Fictional	Yossarian tries various schemes to avoid flying more missions.	Dorian Gray destroyed his portrait and instantly aged to his true years.

Table 1: Overview of truth types used in our experiments, with examples of true and false statements for each subtype.

ity from simple single-digit operations (e.g., "2 + 3 = 5") to multi-digit arithmetic (e.g., "237 + 142 = 379"). These truths are algorithmically verifiable and do not depend on linguistic ambiguity or world knowledge.

These statements are intentionally minimal and unambiguous, making them ideal for probing how LLMs encode internally verifiable logical structure.

3.2 Logical Truths

Logical truths involve relational or boolean reasoning. Like arithmetic, these truths are not grounded in external knowledge but instead rely on abstract internal structure. These include inequalities, set membership, and boolean statements.

These statements serve to evaluate whether LLMs can represent logic-based semantics that are structurally valid but semantically sparse.

3.3 Consensus Truths

Consensus truths refer to culturally accepted or empirically agreed-upon facts, such as "Caracas is a city in Venezuela." These statements are not logically derivable but are widely accepted within human knowledge. We distinguish **factual** truths, drawn from public knowledge, such as countries and capital cities, from **fictional** truths, such as statements about characters in novels, e.g., "Dorothy from The Wizard of Oz is from Kansas."

4 Data

All datasets were formatted consistently—one statement and a true or false label—for training linear probes and sparse autoencoders. The full datasets will be released with the final version of the paper.

All statements are short, declarative, and designed for binary classification. We include approximately 5,000 examples per dataset, balanced evenly between true and false labels.

A summary of the data used, alongside examples, is shown in Table 1

4.1 Arithmetic, Logical, and Symbolic

We constructed the arithmetic, logical, and symbolic datasets synthetically using Python, generating both true and false statements for binary classification. The full code is in Appendix A.2. For the boolean truths, we were careful not to repeat the use of *A* and *B* as characters, otherwise our datasets would feature many repeats. Where natural language was used, as opposed to symbolic, statements were limited to English.

4.2 Consensus

For factual truths, we used data from Minervini (2024), which consists of concise, declarative statements labeled as true or false and organized by domain (e.g., capital cities and country names). This dataset is in English with a Western cultural focus, which we acknowledge as a limitation but consider sufficient for our investigation into truth representation patterns[1].

We generated fictional truths using an LLM (Anthropic Claude 3.7), prompting it to produce both true and false statements grounded in fictional worlds. While the use of an LLM to generate fictional truths raises the possibility of contamination—i.e., the same model or data being seen by the models under test—we argue that this risk is minimal. We did not observe unusually high probe performance on this subset.

5 Model Architectures and Representations

We worked with multiple transformer-based language models:

- LLaMA 3.2 Instruct (1B- and 3B-parameter variants)
- GPT-2 (large, 774M-parameter)
- BERT (base and large variants, with 110M and 340M parameters respectively)

These models differ in architecture and pretraining objectives but all follow the general transformer structure, composed of stacked layers of self-attention and feedforward blocks interleaved with residual connections.

5.1 Masked (Encoder-Only) Models

BERT is pretrained using masked language modeling. We looked at the hidden state corresponding to the special classification token `[CLS]` at each layer:

$$\mathbf{h}^{(l)} = \mathbf{x}_l^{[\texttt{CLS}]} \in R^d$$

This vector is designed to aggregate information across the entire input sequence and is commonly used for classification tasks. (We did attempt taking the mean of all elements as a potential different metric, but that did not provide sufficiently different results to warrant a deviation from simply using the `[CLS]` token.)

5.2 Autoregressive (Decoder-Only) Models

GPT-2 and LLaMA are trained autoregressively. They do not produce a `[CLS]` token. Instead, we extracted the representation of the final token in the sequence from the residual stream after the transformer block at each layer. This is referred to as `resid_post` in libraries such as Nanda and Bloom (2022)'s TransformerLens:

$$\mathbf{h}^{(l)} = \mathbf{x}_l^{(T)} \in R^d$$

Here, T is the index of the final token in the input. The residual stream $\mathbf{x}_l$ captures all computation up to and including layer l, as it is the input to the subsequent layer's attention and MLP blocks.

6 Experiments

We created a software tool to conduct our experiments, written in Python and using the Streamlit framework. We hope this will be helpful to future researchers. Details of the tool can be found in Appendix A.1.

[1] Other similar datasets—including Lin et al. (2022) and Clark et al. (2019)—were not sufficiently unambiguous and simple in their statements for the purposes of this work.

6.1 PCA

Before even enlisting the help of linear probes and sparse autoencoders, we applied Principal Component Analysis (PCA) to each layer's hidden states and plotted projections, with two components, of colored true-false examples, as well as the decision boundary. This—if, and only if, truth were the principal feature—would allow us to visualize quickly which layer of the LLM best separated true and false statements.

6.2 Linear Probes

To locate the position within an LLM at which a particular concept was encoded, we enlisted linear probes.

We created a simple classifier. Let $\mathbf{h}_i^{(l)} \in R^d$ denote the hidden state of the i-th example at layer l, where d is the hidden dimension of the model. We defined a linear probe $f^{(l)} : R^d \to [0, 1]$ as a single-layer neural network followed by a sigmoid activation (σ):

$$f^{(l)}(\mathbf{h}_i^{(l)}) = \sigma\left(\mathbf{w}^{(l)\top}\mathbf{h}_i^{(l)} + b^{(l)}\right)$$

Here, $\mathbf{w}^{(l)} \in R^d$ and $b^{(l)} \in R$ are the probe's learnable parameters at layer l. The output is interpreted as the probability that the input is a true statement (1, as opposed to 0).

Given a dataset of N examples with binary labels $y_i \in \{0, 1\}$, we trained each probe to minimize the binary cross-entropy loss:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\Big[y_i \log f^{(l)}(\mathbf{h}_i^{(l)}) + (1 - y_i)\log\left(1 - f^{(l)}(\mathbf{h}_i^{(l)})\right)\Big]$$

This was optimized with Adam, with a learning rate, $\eta = 10^{-2}$, training each probe for $E = 100$ epochs.

For each layer l, we evaluated the probe on a held-out test set using classification accuracy:

$$\text{Accuracy}^{(l)} = \frac{1}{M}\sum_{j=1}^{M} I\left[\hat{y}_j^{(l)} = y_j\right]$$

where $\hat{y}_j^{(l)} = I[f^{(l)}(\mathbf{h}_j^{(l)}) > 0.5]$, and M is the number of test examples.

As per Hewitt and Liang (2019)'s work, we needed to assess whether the probes were detecting genuine truth signals or merely learning dataset artifacts. We therefore performed *control tasks* by shuffling the labels y_i, retrained the probes using the shuffled labels y_i^{ctrl}, and computed control accuracy:

$$\text{Accuracy}_{\text{ctrl}}^{(l)} = \frac{1}{M}\sum_{j=1}^{M} I\left[\hat{y}_j^{\text{ctrl},(l)} = y_j\right]$$

This would always give a baseline of 0.5 given that we were using simple binary classifiers. The researchers defined *selectivity* as the difference between true and control accuracies:

$$\text{Selectivity}^{(l)} = |\text{Accuracy}^{(l)} - \text{Accuracy}_{\text{ctrl}}^{(l)}|$$

This measures how much more a probe learns from the true labels compared to shuffled-label baselines (0.5).

We then plotted the accuracy, control accuracy, and selectivity as the primary output for our probing experiments.

6.3 Sparse Autoencoders

To assess whether truth-related features in transformer hidden states can be disentangled, and so features obtained, we trained sparse autoencoders on the representations extracted at each layer. These autoencoders are trained to reconstruct their input while enforcing sparsity in an intermediate latent representation. Our hope was that a sparse latent space would isolate interpretable features, that would vary with different truth types.

Let $\mathbf{h}_i^{(l)} \in R^d$ be the hidden state of the i-th example at layer l. We defined an autoencoder consisting of an encoder $f : R^d \to R^k$ and decoder $g : R^k \to R^d$, with a nonlinearity applied to enforce sparsity:

$$\mathbf{z}_i = f(\mathbf{h}_i^{(l)}) = \text{ReLU}(\mathbf{W}_e\mathbf{h}_i^{(l)} + \mathbf{b}_e)$$

$$\hat{\mathbf{h}}_i = g(\mathbf{z}_i) = \mathbf{W}_d\mathbf{z}_i + \mathbf{b}_d$$

The encoder compresses the input into a latent vector $\mathbf{z}_i$, which is then passed through the decoder to reconstruct the input. In our experiments, the latent dimension is *overcomplete*: $k = 10d$, meaning the bottleneck has ten times more dimensions than the input, following work by Elhage et al. (2022).

To encourage symmetry and reduce parameter count, we enabled tied weights, such that the decoder weight matrix was constrained to be the transpose of the encoder weights:

$$\mathbf{W}_d = \mathbf{W}_e^\top$$

This weight tying was implemented by explicitly copying the encoder weights into the decoder before each forward pass. This practice is common in sparse coding and interpretability work, notably in that of Cunningham et al. (2023).

We minimized a reconstruction loss with an ℓ_1-penalty on the activated latent representation $\mathbf{z}_i$:

$$\mathcal{L}_{\text{SAE}} = \frac{1}{N}\sum_{i=1}^{N} \left\| \hat{\mathbf{h}}_i - \mathbf{h}_i^{(l)} \right\|_2^2 + \lambda \left\| \mathbf{z}_i \right\|_1$$

We used a ReLU activation, set the number of epochs to be 100, the learning rate, $\eta = 0.001$, and sparsity penalty coefficient, $\lambda = 0.01$.

Our analysis included metrics such as Zero Activation Rate (percentage of latent units with zero activation); L1 Sparsity (mean absolute activation across all units); Gini Coefficient, which captured the inequality of activations, and is used to assess how concentrated information is in a small subset of neurons (or, more famously, wealth concentration).

We then visualized these metrics per layer, revealing how sparsity evolves through the LLM.

To get a more intuitive idea of the different features being found, we extracted the top sentence examples that most strongly activated specific features in the bottleneck space. These examples were arranged in a grid, allowing us to inspect which types of sentence, or what element of them, was triggering this feature. This, while not strictly quantitative, would provide some of the most entertaining of our results.

7 Results and Analysis

We evaluate how different types of truth are encoded within LLMs, focusing on two primary axes:

- **Model architecture and scale** — We evaluate multiple transformer models across parameter sizes.
- **Truth type** — We assess the four categories in our taxonomy: arithmetic, logical, symbolic, and consensus.

For each model-truth pair, we train linear probes at every layer to assess where truth is most linearly decodable. We complement this with PCA visualizations and sparse autoencoder analysis to interpret feature structure and sparsity. Given the binary nature of our tasks, control accuracy (from shuffled-label probes) is expected to remain at 0.5, providing a baseline to compute selectivity.

7.1 General Trends by Truth Type

7.1.1 Arithmetic, Logical, and Symbolic Truths

We observe a consistent trend: *simple, internally verifiable truths*—such as single-digit arithmetic—are encoded earlier and more selectively than complex arithmetic or external facts.

In LLaMA 3.2 (3B), for instance, the ability to sum single-digit numbers emerges early (Figure 1), while multi-digit arithmetic only becomes linearly decodable around Layer 18 (Figure 2).

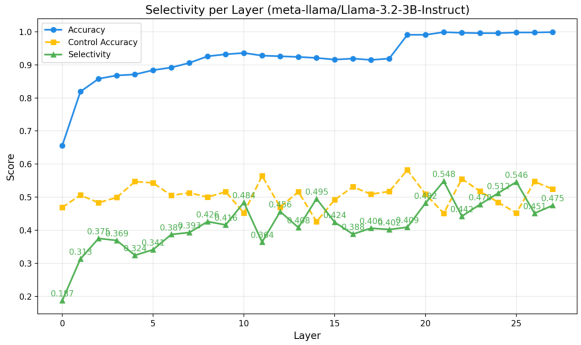

Figure 1: Accuracy, control accuracy, and selectivity for LLaMA 3.2 (3B) on single-digit summation.

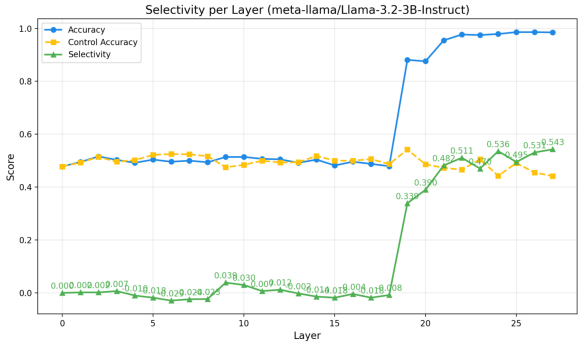

Figure 2: Accuracy, control accuracy, and selectivity for LLaMA 3.2 (3B) on multi-digit summation.

This abstraction-over-depth trend is further illustrated via PCA projections (Figure 3), where separation between true and false statements becomes pronounced only at later layers.

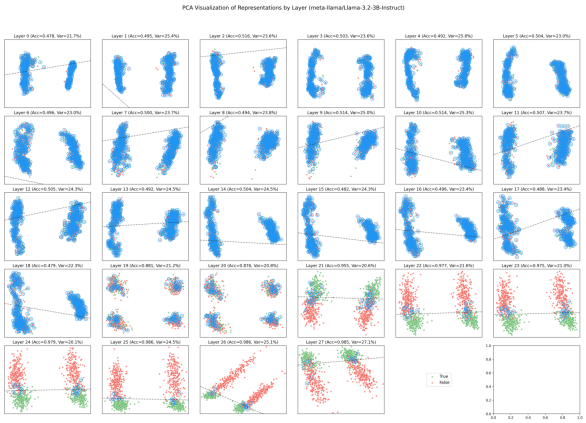

Figure 3: PCA of hidden states across layers (LLaMA 3.2 3B, multi-digit summation). Separation between truth values increases with depth.

Sparse autoencoders reveal further structure. For LLaMA 3.2 (1B), we identify interpretable features such as:

- Feature 16,081 in Layer 14: strongly activated by statements summing to 10
- Feature 15,615: selectively activated by negative sums

These suggest that arithmetic concepts are not only linearly separable but also captured in disentangled features.

In contrast, GPT-2 (774M) and BERT-large (340M) fail to encode multi-digit arithmetic, showing near-zero selectivity and accuracy across all layers (Figure 4).

Figure 4: Accuracy and selectivity for GPT-2 (774M) on multi-digit summation. No signal is detectable across layers.

Logical and symbolic truths follow similar patterns, with shallow encodability for simple constructs, and degradation in smaller models or more complex cases.

7.1.2 Consensus-Based Truths

Consensus-based truths—both factual and fictional—prove more elusive. They are generally encoded later, with lower selectivity, and appear more dependent on model scale.

In smaller models like BERT and GPT-2, consensus facts remain largely undecodable. In LLaMA 3.2 (3B), selectivity improves but still lags behind arithmetic truth encoding.

Yet sparse autoencoders offer a richer picture. Even in the 1B LLaMA, we find features that distinguish between sentence types. For example:

Feature 3,594 (Layer 9): Location-based entities

```
Igor Tamm lived in U.S.
Hans Berger lived in U.S.
Josephine Cochrane lived in U.S.
```

Feature 14,870 (Layer 9): Invention-based entities

```
Karl Landsteiner invented the
 ↪  waterproof fabric.
Ignazio Porro invented the modern
 ↪  electric refrigerator.
John Harrison invented the steel
 ↪  ribbed umbrella.
```

This differentiation implies that models are internally clustering related fact templates—even if truth classification is weak—suggesting latent semantic structure.

7.2 Selectivity Summary

Table 2 summarizes selectivity scores across models, truth types, and layers. Selectivity above 0.2 typically indicates meaningful encoding; lower values suggest little or no signal.

Notably, GPT-2 shows occasional early-layer selectivity followed by rapid loss—suggesting temporary feature emergence that is not preserved. BERT-large shows marginal gains but still struggles with deeper truths. LLaMA 3.2 consistently outperforms smaller models across types.

8 Conclusion

This paper investigates how different types of truth are represented across the internal layers of large language models. We introduce a taxonomy of truth—covering arithmetic, logical, symbolic, and consensus-based statements—and use linear

Model	Parameters	Truth Type	Layer with Max Selectivity	Max Selectivity	Final Selectivity
LLaMA 3.2-1B	1B	Summation (Single-Digit)	10 / 15	0.44	0.44
		Summation (Multi-Digit)	15 / 15	0.23	0.23
		Consensus: Fictional	8 / 15	0.35	0.28
		Consensus: Factual	9 / 15	0.36	0.27
LLaMA 3.2-3B	3B	Summation (Single-Digit)	19 / 27	0.60	0.45
		Summation (Multi-Digit)	25 / 27	0.54	0.49
		Consensus: Fictional	23 / 27	0.40	0.35
		Consensus: Factual	15 / 27	0.47	0.37
BERT-base	110M	Summation (Single-Digit)	7 / 12	0.25	0.20
		Summation (Multi-Digit)	10 / 12	0.01	0.02
		Consensus: Fictional	10 / 12	0.28	0.24
		Consensus: Factual	7 / 12	0.15	0.13
BERT-large	340M	Summation (Single-Digit)	15 / 24	0.35	0.31
		Summation (Multi-Digit)	22 / 24	0.02	0.02
		Consensus: Fictional	14 / 24	0.30	0.29
		Consensus: Factual	24 / 24	0.22	0.22
GPT-2-large	774M	Summation (Single-Digit)	18 / 35	0.34	0.01
		Summation (Multi-Digit)	8 / 35	0.03	0.00
		Consensus: Fictional	23 / 35	0.33	0.08
		Consensus: Factual	16 / 35	0.22	0.04

Table 2: Linear probe selectivity across models and truth types. Values show layer of maximum selectivity (layer index / total), selectivity at that layer, and final-layer selectivity.

probes, control tasks, and sparse autoencoders to probe how these categories are encoded in models including BERT, GPT-2, and LLaMA 3.2.

Our findings show that truth is not encoded uniformly: distinct truth types emerge at different layers, with simple arithmetic facts appearing earlier than complex or abstract knowledge. Selectivity analysis confirms that these representations are not artifacts of dataset bias, and sparse autoencoders reveal interpretable neurons that align with specific truth patterns, such as "[person] lived in [place]" or arithmetic involving certain digits. Sparsity tends to increase in deeper layers, suggesting a progressive abstraction or compression of truth-relevant features.

These results suggest that LLMs do not treat truth as a monolith, but instead encode different kinds of factual knowledge in specialized ways. This opens new directions for interpretability research and has implications for model alignment, factuality, and the safe deployment of LLMs.

Limitations

This work represents an initial investigation into how different types of truth are encoded within large language models. While our results provide meaningful insights, several limitations constrain the scope and generalizability of our findings.

First, our datasets are limited in both content and structure. Many were synthetically generated, with simple, declarative sentence forms and restricted vocabulary. Even the consensus-based datasets, such as Minervini (2024), were aggregated across categories and drawn exclusively from English-language, culturally Western contexts. As such, our conclusions may not generalize to more linguistically diverse, complex, or ambiguous truth expressions.

Second, we evaluated only a small subset of model architectures and sizes, focusing on relatively low-parameter variants of BERT, GPT-2, and LLaMA. Our analysis does not cover more recent or larger-scale models, nor does it systematically vary architectural components such as attention mechanisms or pretraining objectives.

Third, while linear probes and sparse autoencoders are well-established tools for interpretability, we did not fully explore the breadth of their configurations. In particular, hyperparameters for the sparse autoencoders were held constant across experiments, and we did not investigate the interpretability of the latent features with as much rigor as possible. Future work could apply linear probes directly to the autoencoder's latent space to examine whether disentangled features preserve or enhance truth selectivity.

Finally, this study focuses exclusively on internal representations and does not assess whether truth

encoding influences downstream behavior or generation. We do not claim that a model "believes" a statement it encodes as true. Future research could investigate the causal role of these internal truth representations in generation, calibration, or alignment settings, potentially by performing interventions or tracing activation flows during inference.

References

Guillaume Alain and Yoshua Bengio. 2017. Understanding intermediate layers using linear classifier probes.

Amos Azaria and Tom Mitchell. 2023. The internal state of an LLM knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976, Singapore. Association for Computational Linguistics.

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

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? . In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. 2023. Sparse autoencoders find highly interpretable features in language models.

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

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. 2022. Toy models of superposition.

Allyson Ettinger, Ahmed Elgohary, and Philip Resnik. 2016. Probing for semantic evidence of composition by means of simple classification tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 134–139, Berlin, Germany. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari,

Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International*

Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2024. Inference-time intervention: Eliciting truthful answers from a language model.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods.

Samuel Marks and Max Tegmark. 2024. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

Pasquale Minervini. 2024. True-false dataset. https://huggingface.co/datasets/pminervini/true-false.

Neel Nanda and Joseph Bloom. 2022. Transformerlens. https://github.com/TransformerLensOrg/TransformerLens.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI*. Accessed: 2024-11-15.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.

A Appendix

A.1 Streamlit-Based Tool

To conduct this work, we used the Python programming language and wrapped our code into an easy-to-use user interface with the Streamlit framework. This tool made varying the many parameters, hyperparameters, models, datasets, etc., extremely easy. It allowed for tabbed visualization and saving of previous runs (as well as metadata about them). And we configured it to run on either CUDA or MPS, meaning that we conducted some runs on cloud-based GPUs rather than an Apple Silicon MacBook Pro. We hope the tool can be used to further mechanistic interpretability work.

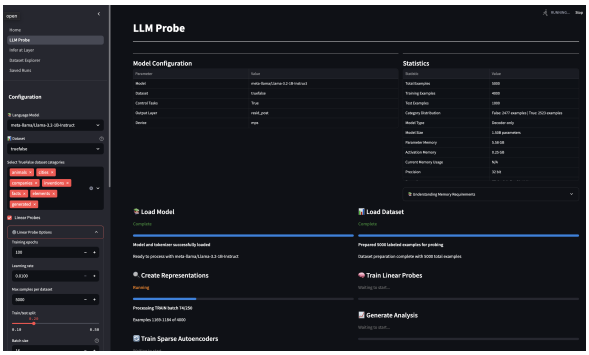

Figure 5: A tool to help run the experiments and analysis described in this paper.

A.2 Creation of logical datasets

```python
import csv
import random
import string


def generate_summation_dataset_csv(max_number, n=5000):
    with open(f"summation_{max_number}.csv", mode="w", newline="") as file:
        writer = csv.writer(file)
        writer.writerow(["statement", "label"])

        for i in range(n):
            a = random.randint(0, max_number)
            b = random.randint(0, max_number)

            if i % 2 == 0:
                correct_sum = a + b
                text = f"{a} + {b} = {correct_sum}"
                label = 1
            else:
                incorrect_sum = (
                    a + b + random.choice([i for i in range(-10, 11) if i != 0])
                )
                text = f"{a} + {b} = {incorrect_sum}"
                label = 0

            writer.writerow([text, label])


def generate_inequality_dataset_csv(max_number, n=5000):
    with open(f"inequality_{max_number}.csv", mode="w", newline="") as file:
        writer = csv.writer(file)
        writer.writerow(["statement", "label"])

        for i in range(n):
            a = random.randint(0, max_number)
            b = random.randint(0, max_number)

            # 50% chance of being correct
            if i % 2 == 0:
                if a == b:
                    a += 1  # ensure inequality
                statement = f"{a} > {b}" if a > b else f"{b} > {a}"
                label = 1
            else:
                if a == b:
                    b += 1
                statement = f"{a} > {b}" if a <= b else f"{b} > {a}"
                label = 0

            writer.writerow([statement, label])


def generate_even_odd_dataset_csv(max_number, n=5000):
    with open(f"even_odd_{max_number}.csv", mode="w", newline="") as file:
        writer = csv.writer(file)
        writer.writerow(["statement", "label"])
        for i in range(n):
            a = random.randint(0, max_number)
            if i % 2 == 0:
                statement = f"{a if a % 2 == 0 else a + 1} is even"
                label = 1
            else:
                statement = f"{a if a % 2 != 0 else a + 1} is even"
                label = 0
            writer.writerow([statement, label])


def generate_divisibility_dataset_csv(max_number, divisor=5, n=5000):
    with open(f"divisible_by_{divisor}_{max_number}.csv", mode="w", newline="") as
    ↪  file:
```

```
        writer = csv.writer(file)
        writer.writerow(["statement", "label"])
        for i in range(n):
            if i % 2 == 0:
                a = random.randint(0, max_number // divisor) * divisor
                statement = f"{a} is divisible by {divisor}"
                label = 1
            else:
                a = random.randint(0, max_number)
                while a % divisor == 0:
                    a = random.randint(0, max_number)
                statement = f"{a} is divisible by {divisor}"
                label = 0
            writer.writerow([statement, label])


def generate_multiplication_dataset_csv(max_number, n=5000):
    with open(f"multiplication_{max_number}.csv", mode="w", newline="") as file:
        writer = csv.writer(file)
        writer.writerow(["statement", "label"])
        for i in range(n):
            a = random.randint(0, max_number)
            b = random.randint(0, max_number)
            if i % 2 == 0:
                correct = a * b
                statement = f"{a} * {b} = {correct}"
                label = 1
            else:
                incorrect = a * b + random.choice([j for j in range(-10, 11) if j !=
                ↪  0])
                statement = f"{a} * {b} = {incorrect}"
                label = 0
            writer.writerow([statement, label])


def generate_chained_inequality_dataset_csv(max_number, n=5000):
    with open(f"chained_inequality_{max_number}.csv", mode="w", newline="") as file:
        writer = csv.writer(file)
        writer.writerow(["statement", "label"])
        for i in range(n):
            if i % 2 == 0:
                a, b, c = sorted(random.sample(range(max_number), 3))
                statement = f"{a} < {b} < {c}"
                label = 1
            else:
                # force a false condition
                while True:
                    a = random.randint(0, max_number)
                    b = random.randint(0, max_number)
                    c = random.randint(0, max_number)
                    if not (a < b < c):
                        break
                statement = f"{a} < {b} < {c}"
                label = 0
            writer.writerow([statement, label])


def get_random_vars(n=2):
    """Get n unique random uppercase letters"""
    return random.sample(string.ascii_uppercase, n)


def generate_boolean_and_dataset_csv(n=5000):
    with open("boolean_and.csv", mode="w", newline="") as file:
        writer = csv.writer(file)
        writer.writerow(["statement", "label"])
        for i in range(n):
            var1, var2 = get_random_vars(2)
            val1 = random.choice(["true", "false"])
            val2 = random.choice(["true", "false"])
```

```
            label = 1 if val1 == "true" and val2 == "true" else 0
            statement = f"If {var1} is {val1} and {var2} is {val2}, then {var1} and {var2} is true"
            writer.writerow([statement, label])


def generate_boolean_or_dataset_csv(n=5000):
    with open("boolean_or.csv", mode="w", newline="") as file:
        writer = csv.writer(file)
        writer.writerow(["statement", "label"])
        for i in range(n):
            var1, var2 = get_random_vars(2)
            val1 = random.choice(["true", "false"])
            val2 = random.choice(["true", "false"])
            label = 1 if val1 == "true" or val2 == "true" else 0
            statement = f"If {var1} is {val1} and {var2} is {val2}, then {var1} or {var2} is true"
            writer.writerow([statement, label])


def generate_boolean_not_dataset_csv(n=5000):
    with open("boolean_not.csv", mode="w", newline="") as file:
        writer = csv.writer(file)
        writer.writerow(["statement", "label"])
        for i in range(n):
            var = random.choice(string.ascii_uppercase)
            val = random.choice(["true", "false"])
            label = 1 if val == "false" else 0
            statement = f"If {var} is {val}, then NOT {var} is true"
            writer.writerow([statement, label])


def generate_boolean_xor_dataset_csv(n=5000):
    with open("boolean_xor.csv", mode="w", newline="") as file:
        writer = csv.writer(file)
        writer.writerow(["statement", "label"])
        for i in range(n):
            var1, var2 = get_random_vars(2)
            val1 = random.choice(["true", "false"])
            val2 = random.choice(["true", "false"])
            label = 1 if (val1 == "true") != (val2 == "true") else 0
            statement = f"If {var1} is {val1} and {var2} is {val2}, then {var1} XOR {var2} is true"
            writer.writerow([statement, label])


def generate_boolean_implies_dataset_csv(n=5000):
    with open("boolean_implies.csv", mode="w", newline="") as file:
        writer = csv.writer(file)
        writer.writerow(["statement", "label"])
        for i in range(n):
            var1, var2 = get_random_vars(2)
            val1 = random.choice(["true", "false"])
            val2 = random.choice(["true", "false"])
            label = 1 if val1 == "false" or val2 == "true" else 0
            statement = f"If {var1} is {val1} and {var2} is {val2}, then {var1} implies {var2} is true"
            writer.writerow([statement, label])


def generate_boolean_iff_dataset_csv(n=5000):
    with open("boolean_iff.csv", mode="w", newline="") as file:
        writer = csv.writer(file)
        writer.writerow(["statement", "label"])
        for i in range(n):
            var1, var2 = get_random_vars(2)
            val1 = random.choice(["true", "false"])
            val2 = random.choice(["true", "false"])
            label = 1 if (val1 == "true") == (val2 == "true") else 0
```

```
            statement = f"If {var1} is {val1} and {var2} is {val2}, then {var1} if and only if {var2} is true"
            writer.writerow([statement, label])


def generate_boolean_nand_dataset_csv(n=5000):
    with open("boolean_nand.csv", mode="w", newline="") as file:
        writer = csv.writer(file)
        writer.writerow(["statement", "label"])
        for i in range(n):
            var1, var2 = get_random_vars(2)
            val1 = random.choice(["true", "false"])
            val2 = random.choice(["true", "false"])
            label = 0 if val1 == "true" and val2 == "true" else 1
            statement = f"If {var1} is {val1} and {var2} is {val2}, then {var1} NAND {var2} is true"
            writer.writerow([statement, label])


def generate_boolean_nor_dataset_csv(n=5000):
    with open("boolean_nor.csv", mode="w", newline="") as file:
        writer = csv.writer(file)
        writer.writerow(["statement", "label"])
        for i in range(n):
            var1, var2 = get_random_vars(2)
            val1 = random.choice(["true", "false"])
            val2 = random.choice(["true", "false"])
            label = 1 if val1 == "false" and val2 == "false" else 0
            statement = f"If {var1} is {val1} and {var2} is {val2}, then {var1} NOR {var2} is true"
            writer.writerow([statement, label])


def generate_digit_count_dataset_csv(n=5000):
    with open("digit_count.csv", mode="w", newline="") as file:
        writer = csv.writer(file)
        writer.writerow(["statement", "label"])
        for i in range(n):
            num = random.randint(1, 99999)
            correct_len = len(str(num))
            if i % 2 == 0:
                statement = f"The number {num} has {correct_len} digits"
                label = 1
            else:
                incorrect_len = correct_len + random.choice([-2, -1, 1, 2])
                incorrect_len = max(1, incorrect_len)
                statement = f"The number {num} has {incorrect_len} digits"
                label = 0
            writer.writerow([statement, label])


def generate_set_membership_dataset_csv(max_number, n=5000):
    with open(f"set_membership_{max_number}.csv", mode="w", newline="") as file:
        writer = csv.writer(file)
        writer.writerow(["statement", "label"])
        for i in range(n):
            the_set = sorted(random.sample(range(max_number), 5))
            if i % 2 == 0:
                x = random.choice(the_set)
                label = 1
            else:
                x = random.randint(0, max_number)
                while x in the_set:
                    x = random.randint(0, max_number)
                label = 0
            statement = f"{x} is in the set {the_set}"
            writer.writerow([statement, label])


generate_summation_dataset_csv(1000)
```

```
generate_summation_dataset_csv(10)
generate_inequality_dataset_csv(1000)
generate_inequality_dataset_csv(10)
generate_even_odd_dataset_csv(1000)
generate_even_odd_dataset_csv(10)
generate_divisibility_dataset_csv(1000, divisor=5)
generate_divisibility_dataset_csv(10, divisor=5)
generate_multiplication_dataset_csv(1000)
generate_multiplication_dataset_csv(10)
generate_chained_inequality_dataset_csv(1000)
generate_chained_inequality_dataset_csv(10)
generate_boolean_and_dataset_csv()
generate_boolean_or_dataset_csv()
generate_boolean_not_dataset_csv()
generate_boolean_xor_dataset_csv()
generate_boolean_implies_dataset_csv()
generate_boolean_iff_dataset_csv()
generate_boolean_nand_dataset_csv()
generate_boolean_nor_dataset_csv()
generate_digit_count_dataset_csv()
generate_set_membership_dataset_csv(10)
generate_set_membership_dataset_csv(1000)
```

A.3 Complete Probing Results

Model	Parameters	Truth Type	Layer with Maximum Selectivity	Maximum Selectivity	Final Layer Selectivity
LLaMA 3.2-1B	1B	Summation (Single-Digit)	10 / 15	0.44	0.44
		Summation (Multi-Digit)	15 / 15	0.23	0.23
		Multiplication (Single-Digit)	11 / 15	0.54	0.52
		Multiplication (Multi-Digit)	15 / 15	0.29	0.29
		Consensus: Fictional	8 / 15	0.35	0.28
		Consensus: Factual	9 / 15	0.36	0.27
		Parity (Single-Digit)	4 / 15	0.86	0.41
		Parity (Double-Digit)	4 / 15	0.48	0.37
LLaMA 3.2-3B	3B	Boolean AND	2 / 27	0.24	0.24
		Boolean NOT	0 / 27	0.70	0.51
		Chained Inequality (Single)	13 / 27	0.58	0.48
		Chained Inequality (Multi)	12 / 27	0.59	0.54
		Consensus: Factual	15 / 27	0.47	0.37
		Consensus: Fictional	23 / 27	0.40	0.35
		Inequality (Multi-Digit)	12 / 27	0.55	0.48
		Inequality (Single-Digit)	26 / 27	0.60	0.51
		Multiplication (Multi-Digit)	26 / 27	0.42	0.42
		Multiplication (Single-Digit)	5 / 27	0.56	0.53
		Parity (Single-Digit)	19 / 27	0.82	0.51
		Parity (Multi-Digit)	2 / 27	0.55	0.50
		Set Membership (Multi)	3 / 27	0.59	0.45
		Set Membership (Single)	3 / 27	0.63	0.49
		Summation (Multi-Digit)	25 / 27	0.54	0.49
		Summation (Single-Digit)	19 / 27	0.60	0.45
BERT-base	110M	Boolean AND	4 / 12	0.24	0.24
		Boolean NOT	1 / 12	0.63	0.46
		Chained Inequality (Multi)	11 / 12	0.42	0.25
		Chained Inequality (Single)	7 / 12	0.63	0.43
		Consensus: Factual	7 / 12	0.15	0.13
		Consensus: Fictional	10 / 12	0.28	0.24
		Digit Count	12 / 12	0.55	0.55
		Divisible by 5 (Multi)	11 / 12	0.58	0.38
		Divisible by 5 (Single)	10 / 12	0.58	0.54
		Inequality (Multi-Digit)	10 / 12	0.41	0.38
		Inequality (Single-Digit)	10 / 12	0.61	0.54
		Multiplication (Multi)	8 / 12	0.11	0.09
		Multiplication (Single)	8 / 12	0.36	0.21
		Parity (Single-Digit)	10 / 12	0.86	0.86
		Parity (Multi-Digit)	7 / 12	0.45	0.39
		Set Membership (Multi)	10 / 12	0.54	0.52
		Set Membership (Single)	9 / 12	0.56	0.41
		Summation (Multi-Digit)	10 / 12	0.01	0.02
		Summation (Single-Digit)	7 / 12	0.25	0.20
BERT-large	340M	Consensus: Factual	24 / 24	0.22	0.22
		Consensus: Fictional	14 / 24	0.30	0.29
		Multiplication (Single-Digit)	18 / 24	0.37	0.27
		Multiplication (Multi-Digit)	16 / 24	0.14	0.08
		Parity (Single-Digit)	22 / 24	0.82	0.45
		Parity (Multi-Digit)	2 / 24	0.48	0.33
		Summation (Multi-Digit)	22 / 24	0.02	0.02
		Summation (Single-Digit)	15 / 24	0.35	0.31
GPT-2-large	774M	Consensus: Factual	16 / 35	0.22	0.04
		Consensus: Fictional	23 / 35	0.33	0.08
		Digit Count	4 / 35	0.54	0.01
		Multiplication (Multi)	3 / 35	0.17	0.00
		Multiplication (Single)	10 / 35	0.44	0.01
		Summation (Multi-Digit)	8 / 35	0.03	0.00
		Summation (Single-Digit)	18 / 35	0.34	0.01
		Divisible by 5 (Single)	6 / 35	0.84	0.01
		Divisible by 5 (Multi)	7 / 35	0.73	0.00

Table 3: Linear probe selectivity results across all models and truth types. Values show layer with maximum selectivity (current/total layers), maximum selectivity achieved, and selectivity at the final layer.