# MULTI-SHOT AUTOINTERP: AGENTS CAN EXPLAIN COMPLEX FEATURES BY REFINING EXPLANATIONS

**Kamal Maher** *
Independent
kmaher@alum.mit.edu

**Simon Elias Schrader** *
University of Oslo
simonschrader@proton.me

**Kola Ayonrinde** †
UK AI Security Institute
koayon@gmail.com

## ABSTRACT

Interpretability studies often show that some features of language models are easily interpretable by humans while others are difficult to explain. If such features are truly beyond human grasp, the field of interpretability may have fundamental limits. We notice, however, that previous human and automated interpretability studies typically seek single-shot explanations, and we hypothesise that the lack of interaction may be a cause for features seeming uninterpretable. To add interaction to feature interpretability, we introduce Multi-Shot AutoInterp (MSA), an agentic framework where LLM agents iteratively refine feature explanations by forming hypotheses, performing targeted experiments on new sequences, and incorporating feedback from feature activations. We find that MSA agents achieve improved AutoInterp performance on difficult-to-interpret sparse autoencoder features with 7.7% greater prediction accuracy than current AutoInterp methods. In addition, MSA agents produced insightful explanations for features that initially appeared opaque to the authors, for example illuminating the meaning of a feature that is activated on tokens representing conceptual parallelism. Our results suggest that features that were previously thought of as uninterpretable may instead be simply not amenable to single-shot explanation, which increases our estimation of how many features are ultimately human-understandable. We argue that through multi-shot approaches humans may be able to understand more of the elusive safety-relevant features in a language model or learn novel, AI-native scientific concepts that were not previously salient to humans.

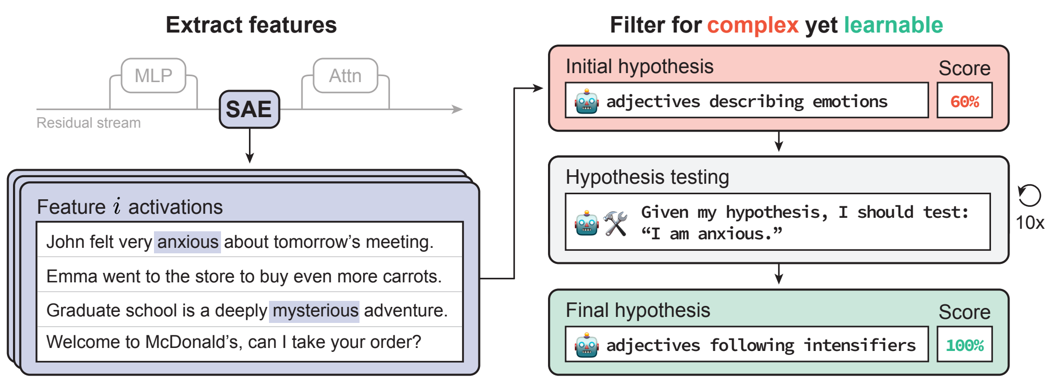


Figure 1: **Feature explanation refinement using our Multi-Shot AutoInterp Agent.** An SAE extracts features from an LLM (blue). Some of these features yield low prediction accuracy using one-shot automated interpretability (red). Our Multi-Shot AutoInterp (MSA) agent iteratively refines its explanations by generating hypotheses, testing these hypotheses against sample sequences, and observing whether the resulting activations fit the agent's prediction. We show that the MSA explanations often outperform single-shot explanations on difficult-to-interpret features (green).

*Equal contribution
†Corresponding author

# 1 INTRODUCTION

Understanding the internal representations of large language models (LLMs) is a central goal of mechanistic interpretability (Bereska & Gavves, 2024; Ayonrinde & Jaburi, 2025). Sparse autoencoders (SAEs) have emerged as a promising tool for decomposing model activations into human-interpretable features (Cunningham et al., 2023; Bricken et al., 2023), but a substantial fraction of SAE features are often reported as uninterpretable (Rajamanoharan et al., 2024; Thasarathan et al., 2025). Features that have been written off as uninterpretable can nonetheless causally influence model behaviour. Without understanding these features, researchers are likely to have an incomplete understanding of the factors that impact model behaviour, and in particular we may overlook mechanisms that are important to the safety of foundation models and critical AI systems. Beyond safety, there is increasing evidence that understanding AI representations may be an accelerant to the sciences. Simon & Zou (2025) and Adams et al. (2025) show that the representations of protein language models can encode biologically meaningful structure, including insights that may be novel to scientists.

To understand hard-to-interpret features, humans may need to learn AI-native concepts from models—a process we refer to as *Concept Enrichment* (Ayonrinde, 2025). One inspiring example of Concept Enrichment comes from Schut et al. (2023) who demonstrate that novel chess strategies from the superhuman chess-playing model AlphaZero (Silver et al., 2017) can be extracted using interpretability tools and then taught to human chess grandmasters. While Schut et al. (2023) show the potential of using interpretability tools to learn from machines, their approach is focused on a narrow task (chess) with a relatively narrow AI model (AlphaZero). LLMs, however, are much more general-purpose AI systems and have been shown to contain rich internal representations across diverse domains (Engels et al., 2025; Gurnee et al., 2025). As such, learning new concepts from the internals of LLMs remains an important open problem that requires scalable methods to identify novel, learnable concepts.

**Our contribution.** To automate the search for novel, learnable concepts, we introduce **Multi-Shot AutoInterp (MSA)**: an agentic framework where LLM agents hypothesise, test, and refine explanations of hard-to-interpret features using feedback from feature activations (Section 3). Applied to complex SAE features, MSA yields statistically significant improvements in detection accuracy and produces explanations that capture structure not recovered by the one-shot explanations of standard AutoInterp methods (Section 5). Furthermore, agents uncover latent structure that was opaque to both the initial explanation and the authors. These results demonstrate that AI agents can learn to explain complex features whose behavior is difficult for humans to predict, offering a concrete step toward extracting novel concepts from machines.

# 2 BACKGROUND

**Sparse Autoencoders.** Sparse Autoencoders (SAEs) decompose a model's internal activations into a sparse, overcomplete set of features (Cunningham et al., 2023; Bricken et al., 2023). Let $\boldsymbol{x} \in \mathbb{R}^d$ be an activation vector at a fixed layer, where $d$ is the layer dimensionality. SAEs approximate $\boldsymbol{x}$ as a sparse linear combination of $K \ll d$ features:

$$\boldsymbol{x} \approx \boldsymbol{W}\boldsymbol{z} + \boldsymbol{b}, \tag{1}$$

where $\boldsymbol{W} \in \mathbb{R}^{d \times K}$ is the decoder matrix, $\boldsymbol{b} \in \mathbb{R}^d$ is a bias term, and $\boldsymbol{z} \in \mathbb{R}^K$ is a sparse activation vector. We say feature $k$ is *activating* with strength $z_k$ when $z_k > 0$.

**Automated Interpretability.** Automated interpretability (AutoInterp) uses LLMs to generate and evaluate natural-language explanations of individual neurons or features (Bills et al., 2023; Paulo et al., 2025a). Explanation quality is measured by *detection accuracy*: given an explanation, a model predicts whether the feature activates in held-out sequences that can be activating or non-activating.

**Concept Enrichment.** *Concept Enrichment* refers to the process of learning novel, model-native concepts from AI systems (Ayonrinde, 2025). A leading example is the work of Schut et al. (2023), who extracted complex, novel chess strategies from a strong model. They then successfully taught these strategies to chess grandmasters.

## 3 METHODS

A feature is *complex* if it yields near-chance detection accuracy using one-shot AutoInterp,[1] suggesting it encodes patterns too intricate for a single explanation to capture.[2] To identify and interpret such features, we use MSA agents to filter for features with low detection accuracy and then iteratively refine feature explanations through targeted experimentation and feedback from activations.

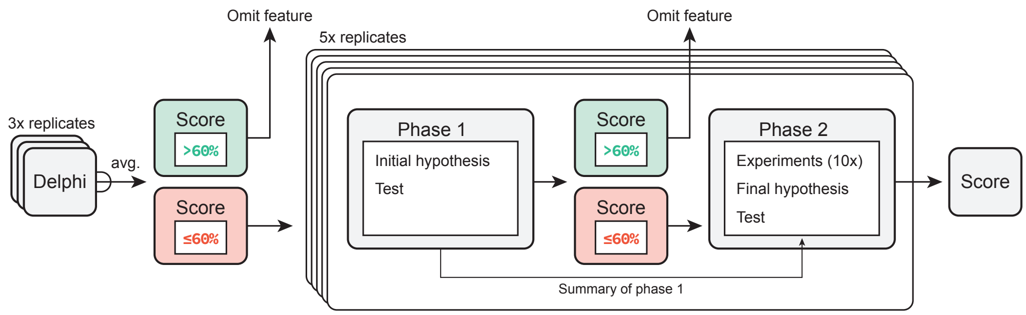


Figure 2: **The MSA agent workflow.** First, we select features with a low Delphi detection accuracy and pass them to the initial phase of the agent study, where an LLM agent is provided with example sequences about the feature and generates an initial explanation. The agent is then scored on a held-out test set based on this explanation. We then pass those features whose detection accuracies remain near-chance to the refinement phase, where the agent refines its explanation by generating its own probe sequences and observing their activations. Finally, we score the agent on a fresh test set based on its refined explanation.

For each SAE feature, we apply the following sequence: Delphi pre-filter $\rightarrow$ initial phase explanation generation and evaluation $\rightarrow$ initial phase filter $\rightarrow$ Refinement phase refinement and re-evaluation. We report the final learnability score $\Delta\mathrm{Acc}$ for features that reach the refinement phase (that is *complex features*).

**Delphi Pre-filter.** As a low-cost initial filter for complex features, we use "Delphi", an automated interpretability method by Paulo et al. (2025a), with a relatively weak LLM explainer. We select features with an average detection accuracy below an upper threshold and pass them to the initial phase.

**Initial Phase: Initial Explanation Generation and Evaluation.** The agent receives activating and non-activating example sequences together with the feature's activation values, and generates an initial explanation $H_1$. The agent is then tested on held-out sequences: given a set of new sequences (both activating and non-activating), the agent predicts which sequences activate the feature, based on $H_1$. We denote the resulting detection accuracy (computed per agent replicate) as $\mathrm{Acc}_1$. This is similar to Delphi, but uses a more capable model, different prompting strategies, and provides the agent with example sequences and their activation values during testing, not just the explanation $H_1$.[3]

**Initial Phase Filter.** We retain only features whose initial phase detection accuracy remains near chance (i.e. $\sim 50\%$ for balanced test sets) for each agent replicate, confirming that the feature resists straightforward explanation. These remaining features are deemed *complex* and proceed to the final phase.

**Refinement Phase: Probe-Guided Hypothesis Refinement and Re-evaluation.** In a new context window, the agent is provided with (1) the example sequences from the initial phase, (2) the

[1] Here, "complex" denotes low one-shot detection accuracy, not intrinsic difficulty.

[2] Following Bricken et al. (2023); Paulo et al. (2025a); Cunningham et al. (2023), we use an LLM's ability to explain a feature as an (imperfect) proxy for how human-understandable a feature is.

[3] A detailed comparison between the initial phase and Delphi is provided in Section C.

test sequences from initial phase with their feature activation values revealed, and (3) the initial hypothesis $H_1$. The agent then repeatedly generates a probe sequence for which it is provided with the feature activation values. For each probe sequence, the agent uses the activation values to update its explanation of the feature accordingly. After generating a fixed number of probe sequences, the agent produces a final refined hypothesis $H_2$ and is evaluated on a fresh test set. We denote the resulting detection accuracy (computed per agent replicate) as $\mathrm{Acc}_2$. Full prompts and implementation details are provided in section F.

**Learnability Metric.** We quantify learnability $\Delta\mathrm{Acc}$ by the improvement in average detection between the refinement phase and the initial phase, $\Delta\mathrm{Acc} = \mathrm{Acc}_2 - \mathrm{Acc}_1$.

## 4 Experimental Setup

**Models and Dataset.** We use Gemma-2-2B (Riviere et al., 2024) as our base model to be explained, leveraging the Gemma Scope SAEs (Lieberum et al., 2024) through SAELens (Bloom et al., 2024). We use SAE with $K = 16{,}384$ features trained on layer 13 residual stream activations (post-MLP) with an average sparsity of $84$.[4] All sequences are sampled from the SmolLM2 corpus (Allal et al., 2025), from which we sample a total of 10M tokens.

**Complex Feature Selection.** We select the first $500$ features by feature ID that have an average Delphi detection accuracy below $0.60$ for the initial phase of the MSA agent, using Qwen3-4B-Instruct-2507 (Yang et al., 2025) as the Delphi explainer model. As SAE features are unordered, this yields a diverse set of features that activate on a variety of different patterns. We select those features for which no agent obtained a initial phase detection accuracy above $0.60$ for the refinement phase of the MSA agent.

**MSA Agent.** We use Claude 4.5 Haiku (Anthropic, 2025) as the LLM agent.[5] We provide agents with $10$ activating and $10$ non-activating sequences for the initial explanation generation and for the tests in both the initial phase and the refinement phase. During the refinement phase, the agent generates $10$ exploratory sequences. For each feature, we sample non-activating sequences randomly, while activating sequences are drawn from the top $100$ sequences by activation, sampling $10$ per stage without replacement. We average agent detection accuracy scores over $5$ different agents with different random seeds, ensuring each agent receives different sequences.

## 5 Results

**Detection Accuracy Improvement.** Figure 3 shows the distribution of improvements in detection accuracy from the initial phase test to the refinement phase test, averaged across agents, for the $45$ features which went through the refinement phase. The mean detection accuracy over all features increased from $51\%$ to $59\%$. In particular, the effective average improvement, i.e. the improvement beyond random guessing (50%), was 7% $\pm$ 1% (SE), corresponding to higher accuracy in detecting the presence of feature activations in sequences after iterative refinement. The distribution is skewed towards positive improvements, with $38$ features showing an increase in detection accuracy, $6$ features showing a decrease, and $1$ feature showing no change. A sign test[6] applied to the agent-averaged mean detection accuracies shows that the predominance of improved over degraded features is statistically significant ($p = 10^{-6}$).

**Example of Complex Feature Learning.** The aggregate results show that most complex features improve under iterative refinement, and here we zoom in for an illustrative example. Feature 30 was initially uninterpretable by the authors but learnable by the agents. This feature had a mean initial detection accuracy of $46\%$ across agents, which improved to $63\%$ after iterative refinement. The

[4] `https://huggingface.co/google/gemma-scope-2b-pt-res/tree/main/layer_13/width_16k/average_l0_84`

[5] See Appendix E for a comparison with GPT-4o-mini.

[6] A sign test is a nonparametric test that assesses whether the median of paired differences differs from zero, using only the sign of differences, with no distributional assumptions. For more details, see (Conover, 1999).

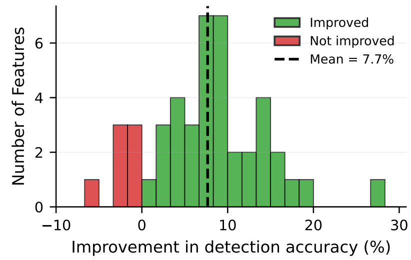


Figure 3: **Iterative refinement of explanations improves detection accuracy across features.** The histogram shows the change in detection accuracies between the initial phase test based on the initial explanation and the refinement phase test for the $45$ complex features considered for the refinement phase, demonstrating that most features saw an improvement. The dashed line shows the average improvement of $7.7\% \pm 1.0\%$ (SE).

maximum initial score across agents was $60\%$, and the maximum final score was $80\%$, indicating that hypothesis refinement was necessary to learn the feature's meaning. Based on its top activating sequences (see Figure 4), feature 30 appears to activate on prepositions such as "on" and "of". However, the agent's initial explanation achieved only $60\%$ accuracy, suggesting a deeper pattern.

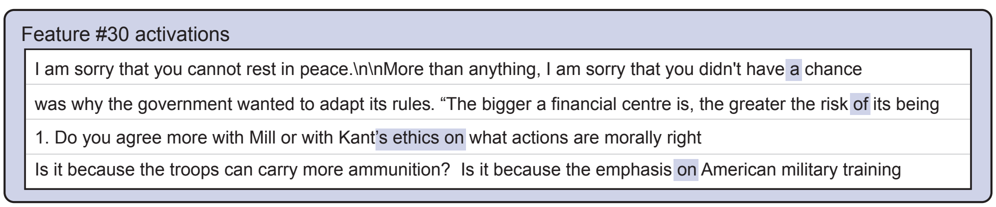


Figure 4: **Example activations for feature 30.** Four out of the top $100$ activating sequences for feature 30. The feature activates on the highlighted tokens.

Figure 5 shows excerpts from the reasoning process of the agent that achieved the highest final detection accuracy. The agent initially hypothesizes that the feature activates on prepositions, achieving $60\%$ detection accuracy. After receiving feedback, it recognizes that it missed structural repetition patterns. The agent then experiments with sequences containing repeated prepositional phrases, yet observes no activations. It notes that the original activating examples contained more surrounding context, so it generates longer sequences and observes activations. Finally, the agent synthesizes these observations into a refined explanation: the feature activates on prepositions and conjunctions appearing within grammatically or conceptually parallel structures. This refined explanation achieves $80\%$ detection accuracy.

Returning to Figure 4, we can verify that the highlighted tokens do indeed appear within parallel structures. The first sequence contains the repeated phrase "I am sorry that you..."; the second contains the parallel comparatives "the bigger" and "the greater"; the third contains the parallel "with Mill" and "with Kant"; and the fourth contains the repeated "Is it because the". These examples validate the agent's refined explanation.

We present additional complex yet learnable features in Section B.

## 6 Discussion and Conclusion

Our results indicate that the Multi-shot AutoInterp (MSA) agent allows LLM agents to produce more accurate explanations of concepts underlying SAE features. This is reflected in an average increase in detection accuracy of $7.7$ percentage points compared to explanations produced after a single exposure to activating sequences, with $84\%$ of features showing some improvement and $34\%$ showing an improvement of more than $10$ percentage points.

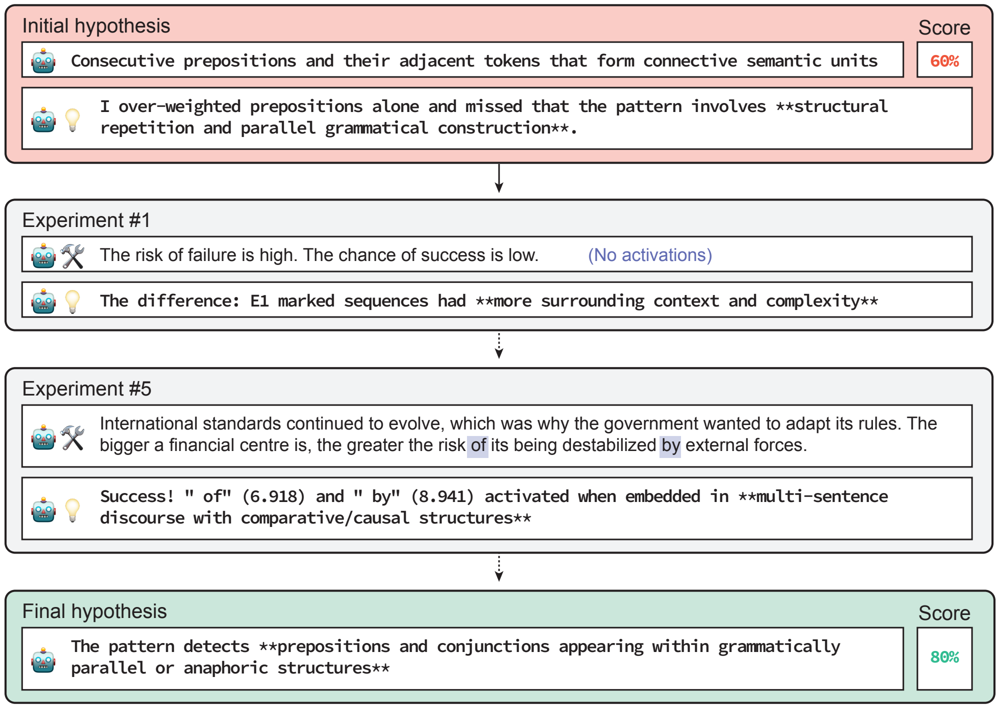


Figure 5: **Agent outputs while learning feature 30.** The agent iteratively refines its hypothesis through experimentation, improving detection accuracy from 60% to 80%. "E1" refers to the initial examples provided to the agent.

One implication of the agents' performance is the potential to improve understanding of features not only for agents, but also for humans. Firstly, there is some evidence that LLMs and humans perform cognitive tasks similarly (Binz & Schulz, 2023).[7] If both humans and AI understand similar explanations then (1) the features that had increased detection accuracy are likely also understandable by humans, and (2) the explanations our agents found have the potential to accelerate human understanding of these features.Secondly, using human learners in the place of our MSA agents could provide an effective way for humans to learn complex features. Together, these two mechanisms provide ways for humans to learn novel concepts from the internal representations of LLMs.

**Limitations.** Our learnability results use LLM agents as proxies for human learners, but agent learnability need not imply human learnability or vice versa. Another limitation is that negative examples are sampled randomly from non-activating sequences instead of using hard negative mining (Sohn, 2016; Liu et al., 2016). Random sampling can hence result in overly easy negatives, and coarse explanations may achieve high detection accuracy without capturing the feature's true boundary. Moreover, positive examples are drawn from high-activation contexts, which do not reflect the full activation distribution. Our claim that the MSA agent is a way to learn concepts in LLMs is also conditional on SAE features being a "real" representation of concepts in an LLM.

**Future work.** In addition to filtering for complex features, importance-based feature selection could help prioritize features that have the greatest impact on the base model's behaviour.[8] To assess whether the learnability candidates produced by the LLM agents are truly human-learnable, future work should carry out studies where humans are presented with approaches similar to the MSA agent we used for the LLM agents.

[7]Though see also Goodfellow et al. (2015) and Moravec's paradox (Moravec, 1988) for ways in which the human and AI skill profiles can be quite different.

[8]We attempted to implement importance filtering using direct logit attribution, and discuss the results in Section D.

## REFERENCES


Etowah Adams, Liam Bai, Minji Lee, Yiyang Yu, and Mohammed AlQuraishi. From mechanistic interpretability to mechanistic biology: Training, evaluating, and interpreting sparse autoencoders on protein language models, 2025. URL https://openreview.net/forum?id=zdOGBRQEbz.

Ajobi. Seer: A framework for agentic interpretability experiments. https://github.com/ajobi-uhc/seer, 2025.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model, 2025. URL https://arxiv.org/abs/2502.02737.

Anthropic. Claude haiku 4.5 model card. https://assets.anthropic.com/m/99128ddd009bdcb/Claude-Haiku-4-5-System-Card.pdf, 2025. Accessed 2025-12-14.

Kola Ayonrinde. Position: Interpretability is a bidirectional communication problem. In *ICLR 2025 Workshop on Bidirectional Human-AI Alignment*, 2025. URL https://openreview.net/forum?id=O4LaRH4zSI.

Kola Ayonrinde and Louis Jaburi. A mathematical philosophy of explanations in mechanistic interpretability – the strange science part i.i, 2025. URL https://arxiv.org/abs/2505.00808.

Leonard Bereska and Efstratios Gavves. Mechanistic interpretability for ai safety – a review, 2024. URL https://arxiv.org/abs/2404.14082.

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

Marcel Binz and Eric Schulz. Using cognitive psychology to understand gpt-3. *Proceedings of the National Academy of Sciences*, 120(6), February 2023. ISSN 1091-6490. doi: 10.1073/pnas.2218523120. URL http://dx.doi.org/10.1073/pnas.2218523120.

Mark Bissell, Michael Byun, and Daniel Balsam. You and your research agent: Lessons from using agents for interpretability research. https://www.goodfire.ai/blog/you-and-your-research-agent, 2025. Goodfire blog post.

Joseph Bloom, Curt Tigges, Anthony Duong, and David Chanin. Saelens. https://github.com/decoderesearch/SAELens, 2024.

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

Garyk Brixi, Matthew G. Durrant, Jerome Ku, Michael Poli, Greg Brockman, Daniel Chang, Gabriel A. Gonzalez, Samuel H. King, David B. Li, Aditi T. Merchant, Mohsen Naghipourfar, Eric Nguyen, Chiara Ricci-Tam, David W. Romero, Gwanggyu Sun, Ali Taghibakshi, Anton Vorontsov, Brandon Yang, Myra Deng, Liv Gorton, Nam Nguyen, Nicholas K. Wang, Etowah Adams, Stephen A. Baccus, Steven Dillmann, Stefano Ermon, Daniel Guo, Rajesh Ilango, Ken Janik, Amy X. Lu, Reshma Mehta, Mohammad R.K. Mofrad, Madelena Y. Ng, Jaspreet Pannu, Christopher Ré, Jonathan C. Schmok, John St. John, Jeremy Sullivan, Kevin Zhu, Greg Zynda, Daniel Balsam, Patrick Collison, Anthony B. Costa, Tina Hernandez-Boussard, Eric Ho, Ming-Yu Liu, Thomas McGrath, Kimberly Powell, Dave P. Burke, Hani Goodarzi, Patrick D. Hsu, and Brian L. Hie. Genome modeling and design across all domains of life with Evo 2, 2025. URL https://www.biorxiv.org/content/10.1101/2025.02.18.638918v1.

W. J. Conover. *Practical Nonparametric Statistics*. John Wiley & Sons, New York, 3 edition, 1999.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models, 2023. URL https://arxiv.org/abs/2309.08600.

Joshua Engels, Eric J. Michaud, Isaac Liao, Wes Gurnee, and Max Tegmark. Not all language model features are one-dimensionally linear, 2025. URL https://arxiv.org/abs/2405.14860.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015. URL https://arxiv.org/abs/1412.6572.

Wes Gurnee, Emmanuel Ameisen, Isaac Kauvar, Julius Tarng, Adam Pearce, Chris Olah, and Joshua Batson. When models manipulate manifolds: The geometry of a counting task. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/linebreaks/index.html.

Jiaojiao Han, Wujiang Xu, Mingyu Jin, and Mengnan Du. Sage: An agentic explainer framework for interpreting sae features in language models, 2025. URL https://arxiv.org/abs/2511.20820.

Been Kim, John Hewitt, Neel Nanda, Noah Fiedel, and Oyvind Tafjord. Because we have LLMs, we can and should pursue agentic interpretability, 2025. URL https://arxiv.org/abs/2506.12152.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2, 2024. URL https://arxiv.org/abs/2408.05147.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. *SSD: Single Shot MultiBox Detector*, pp. 21–37. Springer International Publishing, 2016. ISBN 9783319464480. doi: 10.1007/978-3-319-46448-0_2. URL http://dx.doi.org/10.1007/978-3-319-46448-0_2.

Hans Moravec. *Mind Children: The Future of Robot and Human Intelligence*. Harvard University Press, Cambridge, USA, 1988.

Gonçalo Paulo, Alex Mallen, Caden Juang, and Nora Belrose. Automatically interpreting millions of features in large language models, 2025a. URL https://arxiv.org/abs/2410.13928.

Gonçalo Paulo, Stepan Shabalin, and Nora Belrose. Transcoders beat sparse autoencoders for interpretability, 2025b. URL https://arxiv.org/abs/2501.18823.

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. Improving dictionary learning with gated sparse autoencoders. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2404.16014.

Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris

Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Cogan, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

Lisa Schut, Nenad Tomasev, Tom McGrath, Demis Hassabis, Ulrich Paquet, and Been Kim. Bridging the human-ai knowledge gap: Concept discovery and transfer in alphazero, 2023. URL https://arxiv.org/abs/2310.16410.

Tamar Rott Shaham, Sarah Schwettmann, Franklin Wang, Achyuta Rajaram, Evan Hernandez, Jacob Andreas, and Antonio Torralba. MAIA: A multimodal automated interpretability agent, 2024. URL https://arxiv.org/abs/2404.14394.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017. URL https://arxiv.org/abs/1712.01815.

Elana Simon and James Zou. Interplm: discovering interpretable features in protein language models via sparse autoencoders. *Nature Methods*, 22(10):2107–2117, Oct 2025. ISSN 1548-7105. doi: 10.1038/s41592-025-02836-7. URL https://doi.org/10.1038/s41592-025-02836-7.

Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/6b180037abbebea991d8b1232f8a8ca9-Paper.pdf.

Harrish Thasarathan, Julian Forsyth, Thomas Fel, Matthew Kowal, and Konstantinos Derpanis. Universal sparse autoencoders: Interpretable cross-model concept alignment, 2025. URL https://arxiv.org/abs/2502.03714.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small, 2022. URL https://arxiv.org/abs/2211.00593.

Nicholas Wang, Christoforos Nalmpantis, Ching Fang, Pouya Niki, Mark Bissell, Pooja Kathail, Dron Hazra, Andrey Karailiev, Michael Pearce, Javkhlan-Ochir Ganbat, Archa Jain, Luca Giacomoni, Daniel Balsam, Jonathan Wan, and Ravi Solanki. Using interpretability to identify a novel class of alzheimer's biomarkers, 2026. URL https://www.goodfire.ai/research/interpretability-for-alzheimers-detection.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models, 2022. URL https://arxiv.org/abs/2206.07682.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

# A Related Work

## A.1 Agentic Interpretability

Standard automated interpretability is one-shot (Paulo et al., 2025a): an LLM is provided with examples, produces an explanation, and is evaluated based on the explanation. This works well for many features but not for those whose activation patterns are not apparent from a small sample. One solution is to let agents iteratively explore feature behavior, an approach that has been pursued by recent work at varying levels of automation.

**Interactive and human-in-the-loop systems.** One approach keeps the human in the loop, using agents to accelerate rather than replace the researcher. *Scribe* (Bissell et al., 2025) is an interpretability research agent that can autonomously find interesting SAE features, analyze activations, and run steering experiments, but the research direction remains human-driven. *Seer* (Ajobi, 2025) provides sandboxed environments in which agents execute interpretability experiments iteratively on behalf of a researcher who defines the investigation. Kim et al. (2025) propose a multi-turn conversational approach in which an LLM proactively assists human understanding by building a model of the user's knowledge, requiring continuous human engagement. These systems reduce manual effort but still depend on a human to set goals and judge results.

**Fully automated agentic interpretability.** Fully automated systems remove the human from the loop entirely. MAIA (Shaham et al., 2024) equips LLM agents with tools for generating and testing hypotheses about neuron behavior in vision models. SAGE (Han et al., 2025) iteratively refines SAE feature explanations, improving average explanation quality.

Rather than improving explanations *on average*, our work evaluates the question of whether agentic refinement helps agents learn *complex* features specifically, those that resist one-shot explanation, and whether the resulting explanations help humans understand novel concepts.

## A.2 SAE-based Scientific Discovery

AI systems trained on scientific data may learn representations that capture scientific concepts unknown to human researchers. Extracting such concepts could lead to new scientific insight, but it requires methods that surface interpretable, novel features. In protein biology, Simon & Zou (2025) and Adams et al. (2025) show that applying SAEs to protein language models recovers features corresponding to known functional motifs, demonstrating that SAE features encode meaningful structure. In genomics, Brixi et al. (2025) and Wang et al. (2026) use SAEs to identify biologically relevant features in DNA and genomic foundation models. A key open question in this line of work is whether the novel (previously unknown to humans) concepts these methods uncover are human-understandable. While we have not applied the MSA agent to assess human learnability of novel features, it provides one possible pathway for evaluating such feature learnability.

# B Results for Additional Features

## B.1 Feature 368

At first glance, the pattern in the top activations may seem simple: this feature appears to activate on commas in fronted dependent clauses (e.g. "From the airport, local taxis are available" rather than "Local taxis are available from the airport"). The agent notices this pattern in its initial explanation but achieves only a 55% detection accuracy. It tests a simple fronted clause but observes no activations, hypothesizing that the pattern may depend on preceding context. Once the agent adds preceding context to its previous probe sequence, it observes activations. Its final explanation is that this feature activates on commas following dependent clauses, but only if there is preceding context. Thus, while the activations shown may have suggested a simple one-shot explanation, the agent was able to identify a subtle hidden pattern via multi-shot experimentation.

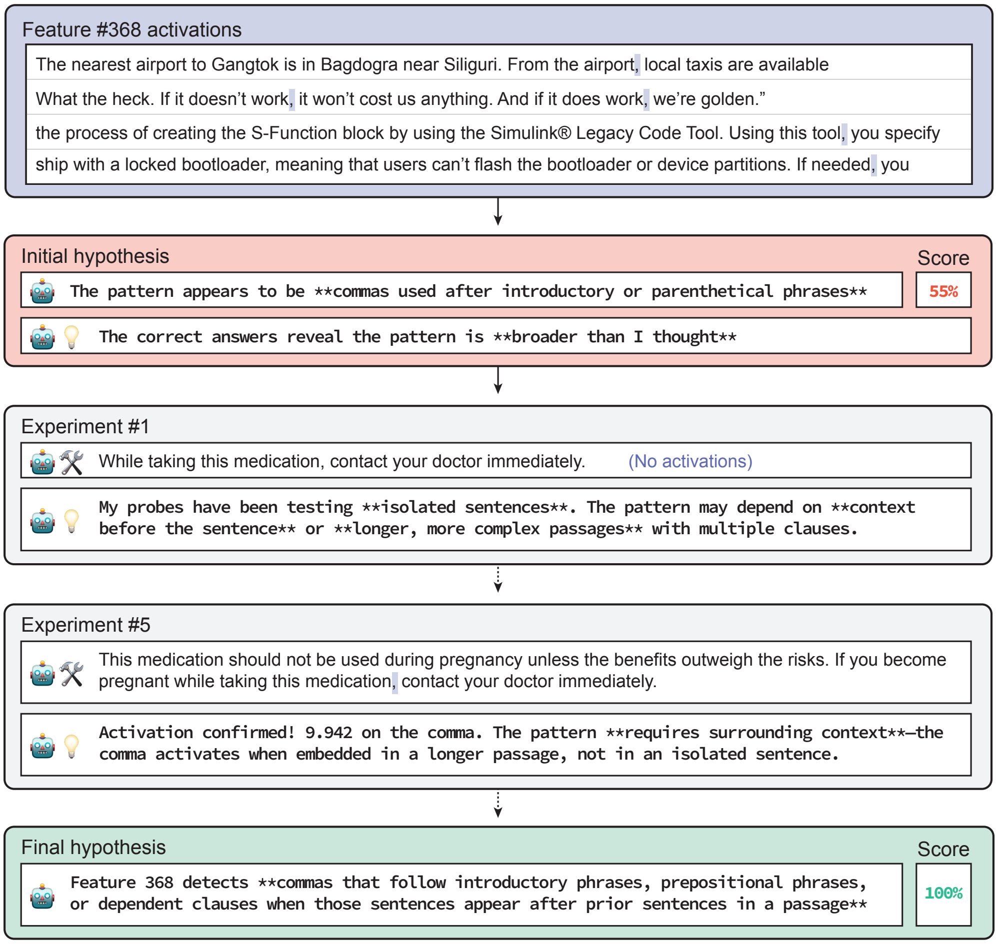


Figure 6: **Example activations and agent output for feature 368.** Four out of the top 50 activating sequences for feature 368 (top/blue). The feature activates on the highlighted tokens. The agent iteratively refines its hypothesis through experimentation, improving detection accuracy from 55% to 100%.

### B.2 FEATURE 505

Due to the activations on a wide variety of nouns and adjectives surrounded by diverse contexts, this feature appeared unclear to the authors. The agent also seemed unsure of the pattern, as it broadly hypothesized that the feature encodes "semantically unified multi-token phrases". However, it eventually tested a paraphrased positive sequence from phase 1 and surprisingly observed no activations. The agent then double-checked by probing with the original sequence and found that it indeed activated. It reasoned that the only difference between its probe and the original sequence was the exact structure—specifically "repetition with elaboration". This explanation ultimately yielded 85% detection accuracy, suggesting that the agent learned the key pattern. Top activating sequences support this pattern. For instance, "negative stereotype" activates when followed by examples like "Rednecks" or "Ignorant". On the other hand, adjectives like "sarcastic" activate when contrasted with "hysterical" within an explanation. Ultimately, by reading over the agent's explanations, the authors were able to better understand the feature's behavior.

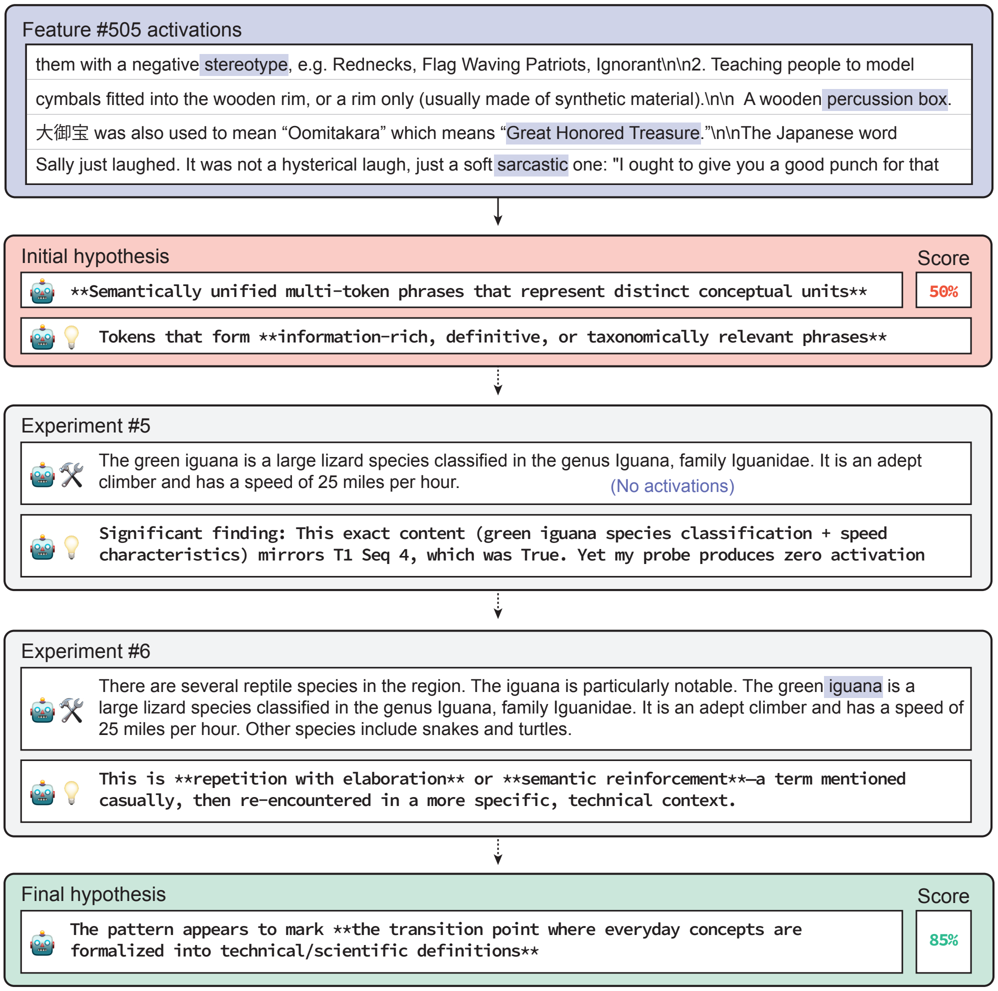


Figure 7: **Example activations and agent output for feature 505.** Four out of the top 50 activating sequences for feature 505 (top/blue). The feature activates on the highlighted tokens. The agent iteratively refines its hypothesis through experimentation, improving detection accuracy from 50% to 85%.

### B.3 Feature 490

The agent begins by probing with a paraphrasing of a positive sequence from phase 1. However, it observes that it surprisingly does not activate. The agent then probes with a more exact copy of the original positive sequence and observes an activation. Comparing the two probes, the agent realizes that the main difference is the structure: activation "depends on comparison/contrast structure". While this explanation explains top activating sequences, the agent continues to iterate and eventually generates a probe that does not activate as expected. The agent then quickly pivots explanations toward "prepositions functioning as connectors", leading to a final explanation of "discourse-coherence markers", which deviates from the original explanation. Nevertheless, with the context of all its tests in mind, the agent is still able to achieve 85% detection accuracy. Despite its tangent, the intermediate explanations helped the authors better understand this feature's behavior.

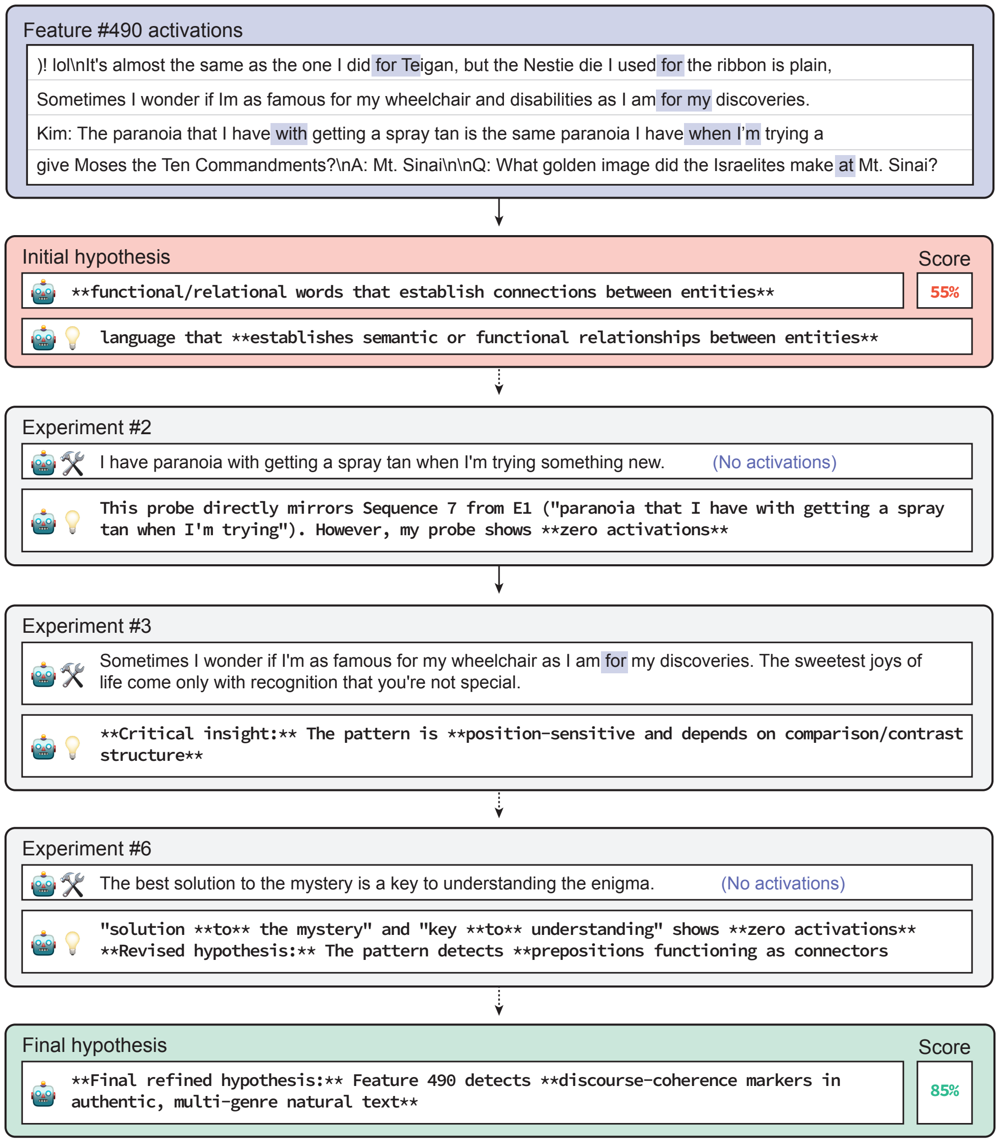


Figure 8: **Example activations and agent output for feature 490.** Four out of the top 50 activating sequences for feature 490 (top/blue). The feature activates on the highlighted tokens. The agent iteratively refines its hypothesis through experimentation, improving detection accuracy from 55% to 85%.

## C Comparison with Delphi

Our implementation of phase 1 in Section 3 is similar to Delphi Paulo et al. (2025b) in that they both generate explanations and then score them using detection accuracy. However, there are a few differences that make phase 1 and Delphi distinct:

- Delphi samples activating sequences across quantiles by default, whereas we sample the top activating sequences.
- Delphi uses an explainer model to generate a feature explanation, which is then handed off to a separate scorer model to predict activations. We use a single model to explain and predict within the same context window.

We suspect that these methodological differences cause detection accuracy in phase 1 to be higher on average than in Delphi.

# D IMPORTANCE CRITERION

We originally sought features that were both complex and *important*. However, we found that these features often encoded relatively simple grammatical concepts. Here, we provide our definition of importance and visualize the relationship between complexity and importance.

## D.1 IMPORTANCE DEFINITION

For an input sequence $\boldsymbol{x}$ and token position $i$, let

$$z_{k,i}(\boldsymbol{x}) \in \mathbb{R}_{\geq 0} \tag{2}$$

denote the activation of SAE feature $k$ at position $i$.

Let $v_j(\boldsymbol{x}, i)$ be the $j$-th top-$M$ predicted next token at position $i$. Let $\boldsymbol{d}_k \in \mathbb{R}^d$ denote the $k$-th column of the SAE decoder matrix $\boldsymbol{W}$, and let $\tilde{\boldsymbol{d}}_k$ and $\tilde{\boldsymbol{u}}_v$ denote the $\ell_2$-normalized SAE decoder and unembedding vectors, respectively. Define the *direct logit attribution* (Wang et al., 2022)

$$a_k(\boldsymbol{x}, i, j) = \tilde{\boldsymbol{d}}_k \cdot \tilde{\boldsymbol{u}}_{v_j(\boldsymbol{x}, i)}, \tag{3}$$

which is a linear approximation of how strongly feature $k$ contributes to increasing the logit of the predicted token $v_j(\boldsymbol{x}, i)$ at position $i$ in input $\boldsymbol{x}$.

We define the *weighted attribution* as

$$w_k(\boldsymbol{x}, i, j) = z_{k,i}(\boldsymbol{x})\, a_k(\boldsymbol{x}, i, j), \tag{4}$$

i.e., it scales the direct logit attribution by the feature's activation strength.

In order to compute feature importance, we consider the expected weighted attribution and expected activation of feature $k$, where the expectation values are taken over dataset samples $\boldsymbol{x}$, token positions $i$, and top-$M$ predicted tokens $j$:

$$\mathbb{E}[w_k] := \mathbb{E}_{\boldsymbol{x},i,j}\left[w_k(\boldsymbol{x}, i, j)\right], \qquad \mathbb{E}[z_k] := \mathbb{E}_{\boldsymbol{x},i}\left[z_{k,i}(\boldsymbol{x})\right]. \tag{5}$$

We define the importance $I_k$ of feature $k$ as

$$I_k = \frac{\mathbb{E}[w_k]}{\mathbb{E}[z_k] + \varepsilon}, \tag{6}$$

where $\varepsilon = 10^{-8}$ is a small constant to prevent division by zero. The importance $I_k$ therefore measures how strongly feature $k$ tends to push the model toward its own preferred next-token predictions, conditional on the feature being active.

This filtering of importance parallels Schut et al. (2023)'s approach, but while they measured importance by presence of a feature in stronger, but not weaker models, we operationalize importance directly via a feature's impact on next-token prediction.

## D.2 IMPORTANCE SETUP AND RESULTS

For our importance calculations, we used the same dataset as described in Section 4. We set $M = 10$ for the top-$M$ predicted tokens to consider in Equation (6).

Figure 9 shows feature importance versus Delphi scores for all 16,384 features, illustrating the four quadrants defined by importance and complexity. Features 7759 and 6213 exemplify how importance filters for grammatical concepts: Feature 7759 activates on the word "should", while feature 6213 activates on nouns following intensifying adjectives (e.g., "very", "sought-after").

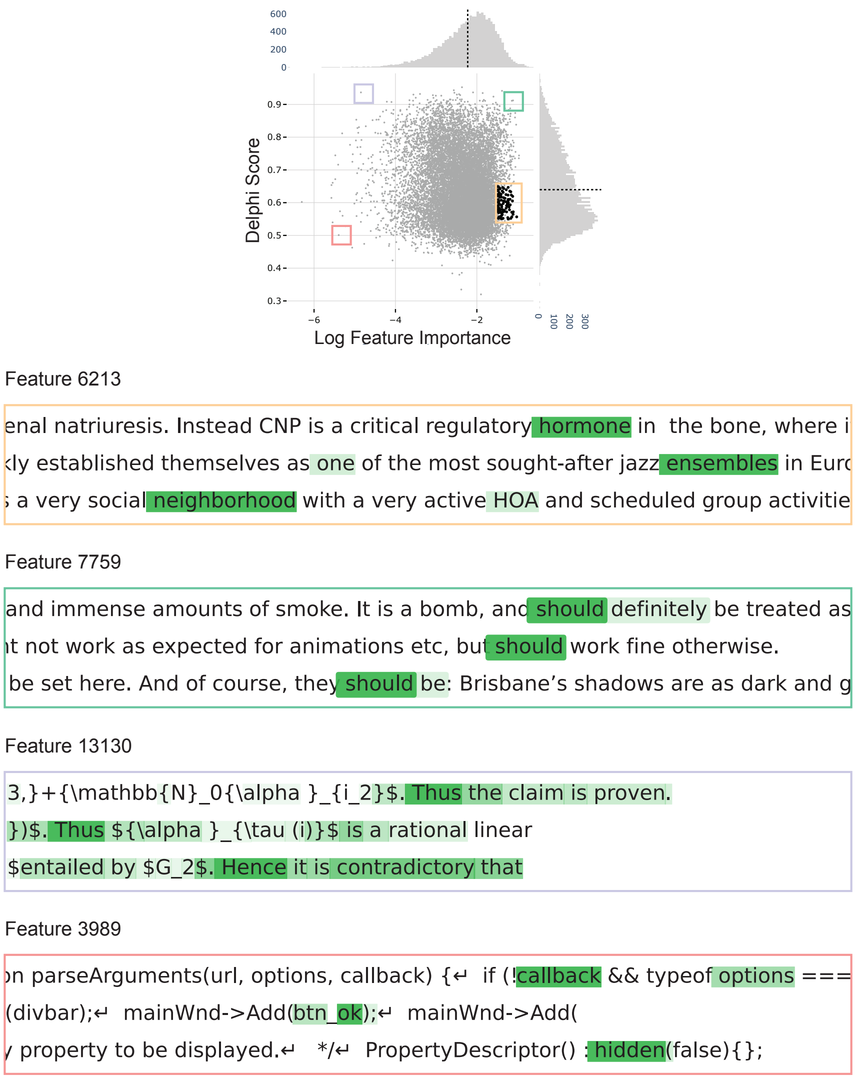


Figure 9: **Feature importance vs. automated interpretability detection accuracy scores for all $16,384$ features of the SAE.** Histograms are shown on the top and right axes for importance (calculated via direct logit attribution) and Delphi's autointerp detection accuracy, respectively. Dashed lines indicate the mean detection accuracy ($0.64$) and mean importance ($0.008$). Each of the four coloured points exemplifies a feature in one of the four quadrants of importance vs. detection accuracy.

## E WEAKER LLM EXPLAINERS

To assess whether feature learnability under the MSA protocol depends on agent capability, we repeated our learnability experiments using a weaker LLM agent, GPT-4o mini. For both LLM agents used (Haiku 4.5 and GPT-4o mini), at least four out of five agents produced correctly formatted output for each feature in the initial phase. Figure E shows the distribution over agent-averaged initial detection accuracies for both LLM agents, showing that GPT 4o-mini produces a much lower average initial score (mean $64\%$) compared to Haiku 4.5 (mean $70\%$).

There are many more features in the lower score range (50-60%) for GPT-4o mini, indicating that there are more features that are challenging for this LLM agent. When running the MSA protocol on the $170$ selected features, we find that the agents struggled with tool use, and there are only $97$ features where each of the five agents produced correctly formatted output in the refinement phase.

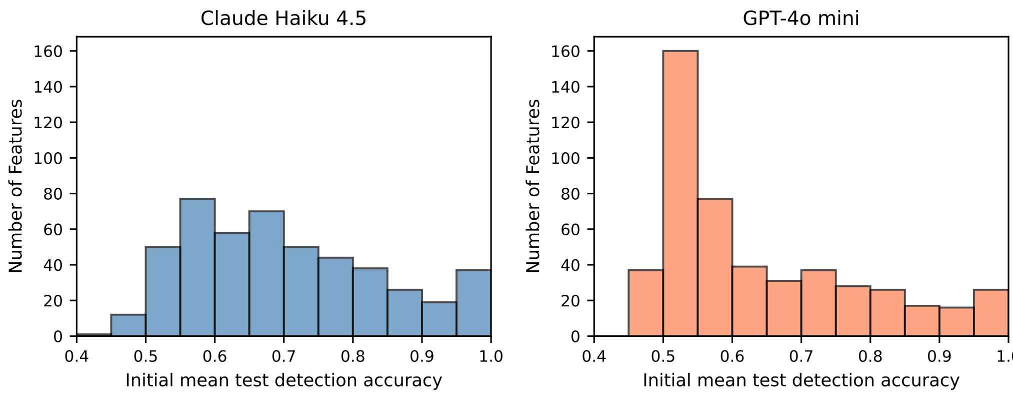


Figure 10: **Initial detection accuracy increases with LLM agent capability.** Distribution of initial detection accuracy scores for Claude Haiku 4.5 and GPT-4o mini. The distribution for Claude Haiku 4.5 is shifted to the right, indicating higher one-shot explanation performance relative to GPT-4o mini. GPT-4o mini therefore treats more features as complex, even when those features are readily explained by Haiku 4.5.

For those $97$ features, we find that the average increase in detection accuracy is only $1.9\% \pm 0.3\%$ (SE), which is significantly lower than the increase observed for Haiku 4.5 ($7.7\% \pm 1.0\%$). There are only $2/97$ features with an increase of more than $10\%$ in detection accuracy, as opposed to $16/45$ features for Haiku 4.5. However, a sign test still indicates that the predominance of $54$ improved over $28$ degraded features is statistically significant ($p = 0.005$).

These results indicate that the choice of LLM agent has a strong effect on the learnability of features through the MSA protocol. This suggests a capability threshold: as explainer models improve, iterated automated interpretability and agentic refinement become increasingly practical research directions for features that previously appeared intractable. This mirrors how few-shot prompting became broadly useful only once models were good enough at in-context learning (Wei et al., 2022).

## F Example Agent Transcript and Tool Definitions

### F.1 Example Agent Transcript

We present the conversation flow of our agent study, showing system prompts, user messages, agent responses, and tool interactions in their original order. While individual messages are truncated for brevity, no steps in the interaction sequence are omitted; truncations are indicated with "...". The prompts and example text sequences are illustrative, the concrete text excerpts relate to feature 30.

# Phase 1

System Prompt

```
You are a creative and analytical thinker focused on identifying
patterns in sequences of text.  You will go through phases of
exploration and testing to understand a single hidden pattern.

In phase E1, you will be shown multiple different sequences, some
containing the hidden pattern and some not.  Those that contain
the pattern will have markings like <<this>> surrounding important
tokens relating to the hidden pattern.  IMPORTANT: The <<angle
```

brackets>> and activation values are annotations to help you|they are NOT the pattern itself. Your hypothesis should describe what the marked tokens have in common, not simply that they are marked. Consecutive important tokens appear together: <<just like this>>. Each sequence is formatted with a header ('===== Sequence i ====='), followed by the sequence itself. A divider ('-----') separates the sequence from numerical values corresponding to each marked token and how representative it is of the hidden pattern. If the sequence does not contain the pattern, this space will say 'None'. You will form a hypothesis about what the pattern is based on these marked sequences. Patterns may involve semantic, grammatical, positional, or other concepts. Pay more attention to high-importance tokens than low-importance tokens when forming hypotheses. Also pay attention to the tokens surrounding the marked tokens, as they too can influence whether the pattern is present.

In phase T1, you will be given new sequences without markings to test your understanding of the pattern. You will use the provided `submit_predictions` tool to submit your True/False predictions for which of these sequences contain the pattern.

Be concise in your answers. Avoid listing examples, and prioritize conveying your hypotheses clearly and succinctly.

User

```
It is now Phase E1.
Follow your instructions given the following sequences:
========== Sequence 0: ==========
but let's divide it into chunks and start understanding it piece by
piece.

Go to line 106 (github gist)

In our try
--------------------
Token activations:
'  but': 0.000
' let': 0.000
''': 0.000
's': 0.000
' divide': 0.000
' it': 0.000
' into': 0.000
' chunks': 0.000
' and': 0.000
' start': 0.000
' understanding': 0.000
' it': 0.000
' piece': 0.000
' by': 0.000
' piece': 0.000
'.': 0.000
'
n
n': 0.000
'Go': 0.000
' to': 0.000
' line': 0.000
' ': 0.000
'1': 0.000
```

```
'0': 0.000
'6': 0.000
' (': 0.000
'github': 0.000
' gist': 0.000
')': 0.000
'
n
n': 0.000
'In': 0.000
' our': 0.000
' try': 0.000
(No activations above threshold)
...
```

Agent

Phase E1 Analysis
Hypothesis: The hidden pattern is: **Preposition-based phrasal or multi-token sequences where specific prepositions appear together with certain surrounding tokens that form meaningful semantic units.**
...

User

```
It is now Phase T1.
Follow your instructions given the following sequences:
========== Sequence 0: ========== , and why does it matter? Where
does it come from? What does it look like in practice? How do
people use it to find religious meaning? Why
...
```

Tool Call

```
submit_predictions("0": false, "1": false, "2": true, "3":
false, ... )
```

Tool Result

```
"0": true, "1": false, "2": true, "3": false,...
```

# Phase 2

System Prompt

```
You will now continue with additional phases to further refine your
understanding of the pattern.
In phase E2, you will design and run experiments to probe the
boundaries of your hypothesis. You will run a single experiment
at a time. In each experiment, you will generate a single
probe sequence and then submit a single tool call using the
`get_activations` tool to see whether the pattern is detected in
that sequence. After each experiment, reflect on the results and
update your hypothesis accordingly, if at all.
Keep in mind all previous results from phases E1, T1, and E2 as
you refine your hypothesis; it should be consistent with all known
data.
```

```
In phase T2, you will again predict the presence of the pattern in
new sequences based on your refined hypothesis.  You will use the
`submit_predictions` tool to submit your True/False predictions.

Be concise in your answers.  Avoid listing examples, and prioritize
conveying your hypotheses clearly and succinctly.
```

User

```
=== RECAP FROM PREVIOUS PHASES ===

In Phase E1, you were shown these LABELED sequences (with
<<markers>> indicating the pattern):
========== Sequence 0:  ========== but let's divide it into chunks
and start understanding it piece by piece.

Go to line 106 (github gist)

In our try
--------------------
Token activations:
'  but':  0.000
' let':  0.000
''':  0.000
's':  0.000
' divide':  0.000
' it':  0.000
' into':  0.000
' chunks':  0.000
' and':  0.000
' start':  0.000
' understanding':  0.000
' it':  0.000
' piece':  0.000
' by':  0.000
' piece':  0.000
'.':  0.000
'
n
n':  0.000
'Go':  0.000
' to':  0.000
' line':  0.000
' ':  0.000
'1':  0.000
'0':  0.000
'6':  0.000
' (':  0.000
'github':  0.000
' gist':  0.000
')':  0.000
'
n
n':  0.000
'In':  0.000
' our':  0.000
' try':  0.000
(No activations above threshold)
...
Your E1 hypothesis was:
# Phase E1 Analysis
## Hypothesis
```

```
The hidden pattern is: **Preposition-based phrasal or multi-token
sequences where specific prepositions appear together with certain
surrounding tokens that form meaningful semantic units.** ...
In Phase T1, you were tested on these UNLABELED sequences:
========== Sequence 0: ========== , and why does it matter? Where
does it come from? What does it look like in practice? How do
people use it to find religious meaning? Why
...
Your predictions and the correct answers were:
Sequence 0: predicted False, actual True
Sequence 1: predicted False, actual False
Sequence 2: predicted True, actual True
Sequence 3: predicted False, actual False
...
Your T1 score was 60%.

=== END RECAP ===

Based on this recap, briefly analyze what you got wrong and refine
your hypothesis. Do NOT make any tool calls yet - Phase E2 will
begin with a separate prompt.
```

Agent

## Analysis of T1 Errors

I made **8 false negatives** (missed true patterns) and **1 false positive**:

**False Negatives (predicted False, was True):**
- Seq 0: "why does it matter? Where does it come from?"
...

User

```
It is now Phase E2.
Follow your instructions to probe your hypothesis.
The `feature_id` argument you must use is 30. Number of probes
remaining: 10.
```

Tool Call

```
get_activations( ===== Probe sequence ===== Do you like apples? Do
you like oranges? Do you like bananas?
--------------------
Token activations:
'Do': 0.000
' you': 0.000
' like': 0.000
' apples': 0.000
'?': 0.000
' Do': 0.000
' you': 0.000
' like': 0.000
' oranges': 0.000
'?': 0.000
' Do': 0.000
' you': 0.000
' like': 0.000
' bananas': 0.000
'?': 0.000
(No activations above threshold)
```

```
...)
```

User

```
Reflect on your probe results. Be concise.
```

Agent

**Reflection:**

My hypothesis about anaphora/repetition was wrong. . .

*[Probing continues for remaining iterations. . . ]*

User

```
It is now Phase T2.
Follow your instructions given the following sequences:
========== Sequence 0: ==========
my energy in answering to the \why" question. Why should people
share? Why should they care about the LoS? Why should they spend
their time on
...
```

Tool Call

```
submit_predictions( "predictions": "0": true, "1": false, ...
)
```

Tool Result

```
"0": true, "1": false, ...
```

### F.2 Tool Definitions

The agent has access to two tools during the study:

Tool Call

```
submit_predictions(predictions: dict[str, bool]) -> str
```

Submit predictions for whether each sequence contains the hidden pattern. The `predictions` argument maps sequence IDs (as strings) to boolean values.

Tool Call

```
get_activations(feature_id: int, sequence: str) -> str
```

Probe a custom sequence to see which tokens activate the pattern. Returns the sequence with activated tokens marked and their activation values.