Automated Interpretability and Feature Discovery in Language Models with Agents

Arnau Marin-Llobet [1] **Javier Ferrando**

Abstract

We introduce an autonomous multiagent framework for mechanistic interpretability that automates both explaining and finding internal features in large language models. The system runs two coupled loops: (1) explanation refinement, where an agent proposes competing hypotheses and iteratively tests them with targeted prompt controls and a multi-metric evaluation; and (2) feature discovery, where an agent generates prompt sets, constructs a k-nearest-neighbor graph in activation space, and retrieves candidate features using statistical separability and semantic coherence criteria. On `Gemma-2` family models and MLP neurons in weight-sparse transformers, our agent improves over one-shot auto-interpretations, discovers language-specific and safety-relevant features, and produces auditable explanation traces, showing that agent-driven empirical loops yield sharper and more falsifiable explanations than one-shot labels.

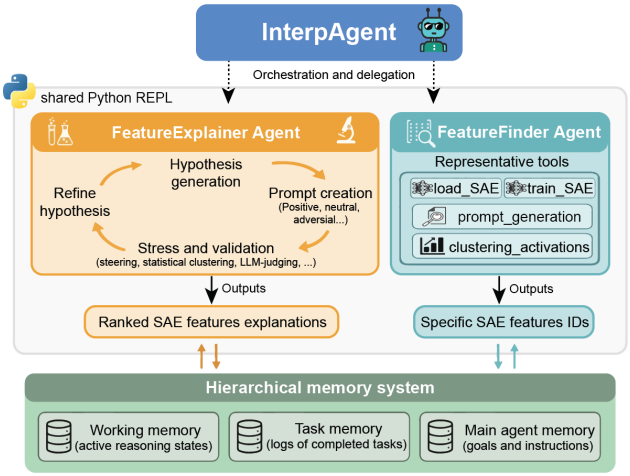

Figure 1. **Overview of the InterpAgent architecture.** A supervisor coordinates two sub-agents, FeatureFinder and FeatureExplainer, which operate within a shared execution environment and persistent memory.

1. Introduction

Understanding how large language models (LLMs) work internally is essential for studying their capabilities, limitations, and safety properties. Researchers often seek to determine which internal representations correspond to particular linguistic patterns, emergent behaviors, or safety-relevant concepts, and how these representations influence model outputs (Elhage et al., 2021; Barez et al., 2025). In practice, gaining this understanding requires exploratory analysis, hypothesis formation, and controlled experimentation on internal activations. As LLMs scale in size and complexity, however, this process becomes increasingly slow and difficult to carry out manually, even for widely studied systems.

Recent progress in mechanistic interpretability has made this challenge more tractable by exposing internal features that align with human-interpretable concepts (Ferrando et al., 2024). Sparse autoencoders, in particular, decompose dense activations into large collections of sparse latent features, reducing polysemanticity and providing a structured basis for analysis (Bricken et al., 2023). Alongside this, automated interpretability methods have emerged that generate natural-language descriptions of neurons or features from their most activating examples (Bills et al., 2023). While effective as metadata or initial explanations, these one-shot descriptions often rely on limited evidence and can be brittle under counterexamples, making them better suited for exploration than for robust explanation.

A less explored but increasingly important limitation is that most existing approaches assume the feature of interest is already known. In realistic workflows, users often begin with only a high-level question—such as which internal features relate to a given language, topic, or behavior—without knowing which of the many available features to inspect. As sparse representations scale to thousands of features per layer, identifying where to look becomes a central bottle-

[1]Harvard University. Correspondence to: Arnau Marin-Llobet <amarinllobet@seas.harvard.edu>.

Proceedings of the Second Workshop on Agents in the Wild: Safety, Security, and Beyond at ICML, 2026.
Copyright 2026 by the author(s).

neck: even precise explanations are of limited value if the relevant features cannot be found in the first place. Despite its importance, systematic feature discovery has received comparatively little attention in the interpretability literature.

These observations suggest that interpretability should be treated not as a static labeling task, but as an experimental process. Human researchers typically iterate between proposing hypotheses, designing targeted tests, analyzing failures, and refining their understanding in response to evidence (Chan et al., 2022). Scaling this workflow requires systems that can autonomously generate hypotheses, run controlled experiments on internal activations, revise explanations based on quantitative feedback, and decide which features are worth investigating. Recently, with the emergence of agentic systems, which can maintain state, compose tools, and adapt their actions over multiple experimental steps, they have become a promising abstraction for automating such iterative scientific workflows.

In this work, we introduce an autonomous multi-agentic framework for mechanistic interpretability that unifies feature discovery and feature explanation within a single empirical loop (Figure 1). Given a natural-language objective, the system searches activation space to identify candidate features that differentiate prompt groups of interest, then iteratively refines and stress-tests competing hypotheses about what those features represent. Rather than producing a single static label, the agent maintains multiple hypotheses, generates contrastive prompt suites, evaluates explanations using quantitative criteria, and revises them in response to concrete failure modes. Evaluated on sparse autoencoder features in the `Gemma-2` family of models (Team et al., 2024), our agent improves upon one-shot automated interpretations and identifies meaningful language-specific and safety-relevant features, highlighting the potential to treat discovery and explanation as inseparable components of scalable and traceable interpretability.

2. Related work

Mechanistic interpretability Mechanistic interpretability seeks to explain model behavior by analyzing internal computations and representations, often by linking activations to human-interpretable features (Elhage et al., 2021; Meng et al., 2022). Prior work has examined a wide range of internal structures, including attention heads, residual streams, and MLP activations, with the goal of identifying directions that correspond to meaningful concepts or computations (Conmy et al., 2023; Wang et al., 2022; Olsson et al., 2022; Lan et al., 2024). Sparse autoencoders and related dictionary-learning approaches formalize this goal by learning overcomplete feature bases whose sparse activations reconstruct a target layer, yielding latent features that can be analyzed as candidate building blocks of model behavior (Cunningham et al., 2023; Rajamanoharan et al., 2024; Gao et al., 2024). These features have been studied through activation statistics, eliciting contexts, and causal interventions that amplify or suppress their activity to assess functional influence (Ghandeharioun et al., 2024; Marks et al., 2024).

Automated interpretability As models have scaled, interpretability has increasingly relied on automated methods that generate hypotheses about internal features. Early literature focused on explanation generation, in which LLMs are prompted with feature-activating examples to produce natural-language descriptions of neurons or latents (Bills et al., 2023). Follow-up work has refined these methods by improving evaluation protocols introducing quality metrics and extending them to large collections of features (Paulo et al., 2025). In parallel, automated verification and circuit discovery methods aim to identify the subgraph of components responsible for a behavior by iteratively pruning irrelevant edges (Chan et al., 2022; Conmy et al., 2023), while representation engineering shows that discovered directions can be used to steer outputs (Zou et al., 2023; Turner et al., 2024). A key open problem across these approaches is reliability at scale: explanations can be correlational and brittle, and verification can be expensive (Barez, 2026).

AI agents for interpretability and discovery Recent work increasingly treats interpretability and scientific discovery as iterative workflows, where hypotheses are proposed, tested, and revised based on evidence. In vision, multimodal agents can probe neurons by generating and editing stimuli to characterize selectivity (Shaham et al., 2024; Camuñas et al., 2025). More broadly, in the physycal sciences, agentic systems are being used to help plan, run, and analyze experiments (Swanson et al., 2025; Zhang et al., 2025; Marin-Llobet et al., 2025; Liang et al., 2023; Aljović et al., 2025; Lin et al., 2025b;a; Huang et al., 2025; Wang et al., 2025; Cong et al., 2025). Similar ideas are also emerging within machine learning itself, where agents generate hypotheses and run experiments (Wu et al., 2024; Lumer et al., 2025). Our work builds on this direction by applying an agentic framework to language-model interpretability and unsupervised feature discovery.

Unlike prior systems that focus on explaining a specified feature, our agent also includes an unsupervised discovery loop that selects candidate features from scratch, and a refinement loop that iteratively sharpens the resulting explanations. To our knowledge, it is the first autonomous interpretability agent for LLMs to combine both capabilities in a end-to-end loop.

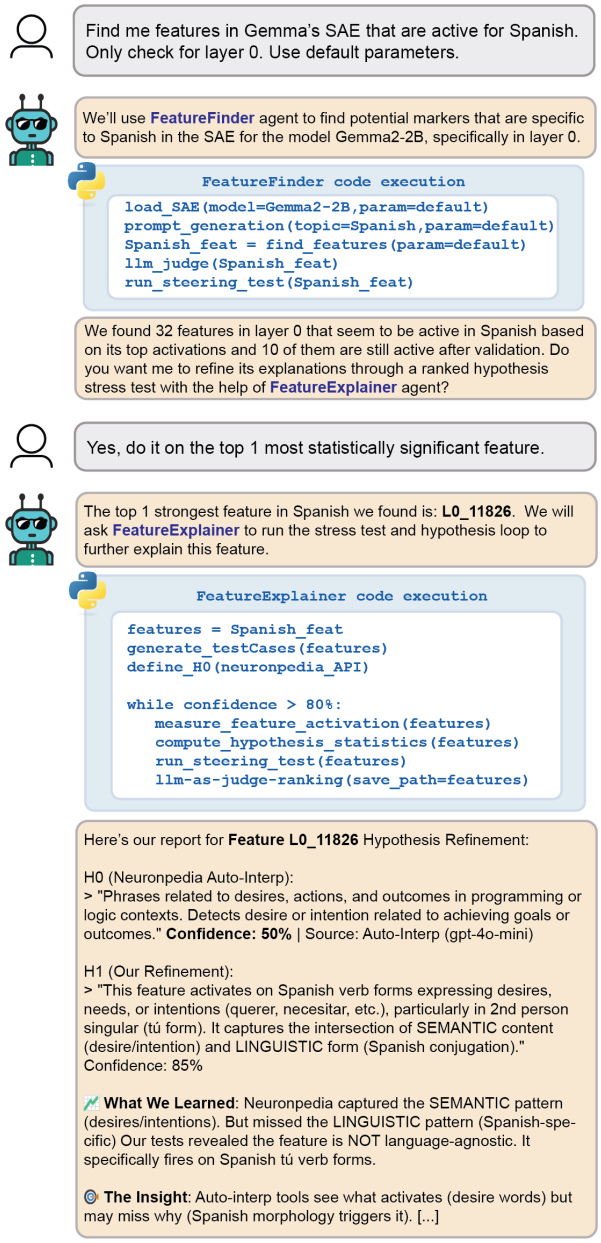

Figure 2. **Example of the Agent interactions**. The user queries for features related to "Spanish". The FeatureFinder executes our statistical discovery algorithm to identify candidate features that distinguish Spanish prompts from others. Upon selecting the top-ranked feature, the FeatureExplainer performs a hypothesis stress test. It refines the baseline Auto-Interp explanation (which missed the linguistic constraint) to identify the feature as activating specifically for "Spanish verb forms expressing desire".

3. Methodology

3.1. System Overview

Here, we present the MechInterp Agent which acts as a supervisor that coordinates two specialized sub-agents. FeatureFinder handles discovery: given a prompt set (often defined by weak labels such as language, topic, or concept), it searches for features whose activation patterns reliably differentiate groups. FeatureExplainer handles refinement: given a candidate feature, it iteratively formulates and stress-tests hypotheses about what drives the feature and, when requested, whether manipulating it causally affects model outputs. Both sub-agents operate within a shared Python execution environment with persistent memory and have access to standard model and interpretability libraries, including Hugging Face tools (e.g., `transformers`, `torch`) and interpretability packages such as `transformer-lens` (Nanda & Bloom, 2022) and `sae-lens` (Bloom et al., 2024).

The supervisor processes user requests such as "find features selective for Spanish text" or "refine the hypothesis for feature 1432" to the appropriate sub-agent(s). There is no fixed workflow: users may run discovery only, interpretation only, or alternate between discovery and refinement as new evidence suggests new directions (Figure 2). Throughout, the system logs the prompts used, the metrics computed, and the outcomes of any interventions, making the investigation auditable and straightforward to reproduce from the saved traces.

3.2. FeatureExplainer: Iterative Hypothesis Refinement

FeatureExplainer turns a candidate feature into a transparent interpretation by running an iterative propose–test–revise loop. Rather than outputting a single description from top activations, it maintains a pool of competing hypotheses, designs contrastive tests to separate them, evaluates evidence with a fixed battery of metrics, and updates the hypotheses based on observed failure modes. The goal is an explanation that (i) predicts when the feature activates on held-out prompts and (ii) when requested, is supported by causal evidence from directly manipulating the feature during inference.

Hypothesis initialization and evaluation sets. The loop begins from one or more seed hypotheses. A seed may be provided by the user, sourced from external baselines (e.g., community annotations), or generated automatically from top-activating examples using a language model. Given a hypothesis h, FeatureExplainer constructs an evaluation set of prompts with n predicted positives and n matched negative controls. Prompt generation is domain-sensitive and aims to isolate the hypothesized factor while controlling for confounds such as topic, length, register, and surface form. For syntactic or code hypotheses, the agent generates well-formed code snippets; for linguistic hypotheses, it generates text in the appropriate language and style. Prompts are then run through the target model, and feature activations are recorded in the chosen basis (SAE features in our experi-

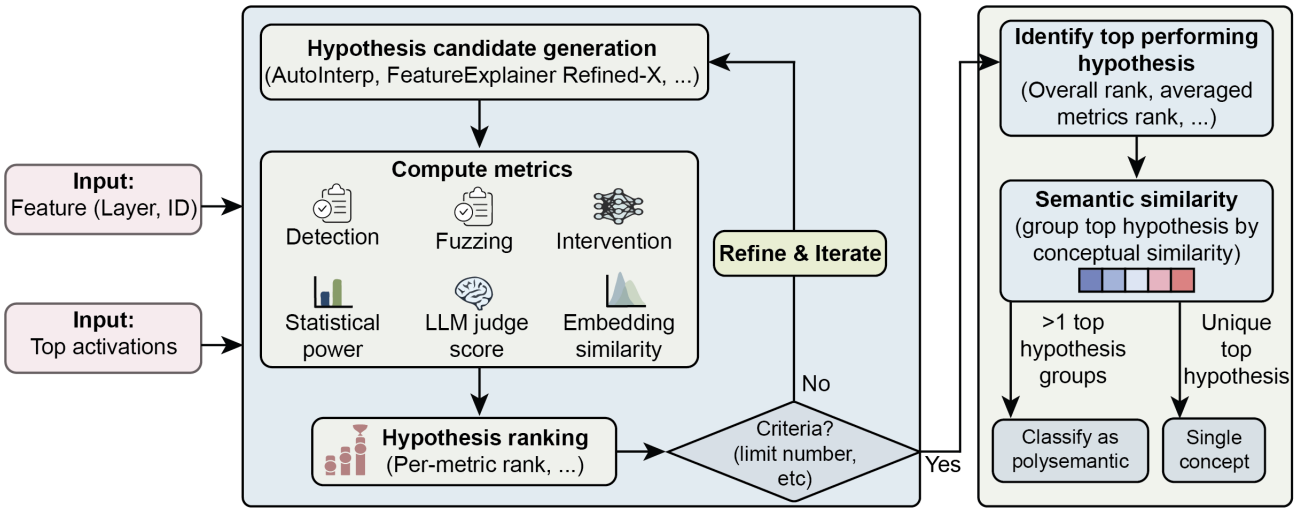

Figure 3. **Schematic of the FeatureExplainer Hypothesis Ranking and Polysemanticity Detection Pipeline.** The workflow inputs a feature (Layer, ID) and its top activations. Hypothesis candidate generation produces initial explanations (e.g., via AutoInterp, FeatureFinder), which then undergo a rigorous multi-metric evaluation. Candidates are assessed across dimensions including detection, fuzzing, intervention, and statistical alignment to produce a Hypothesis ranking. An iterative Refine & Iterate loop optimizes these explanations until criteria are met. Finally, the system identifies top-performing hypotheses and analyzes them for Semantic similarity; if the best explanations cluster into distinct, disjoint concepts, the feature is classified as polysemantic, otherwise as a single concept.

ments). The resulting activation distributions for positives and negatives provide the primary observational evidence used throughout refinement.

Evidence and quality metrics. Each hypothesis is evaluated with seven metrics that jointly capture discriminability, robustness, semantic alignment, and statistical separability; these are the signals used for ranking. We include several standardized and adapted automated-evaluation metrics from the Delphi framework (Paulo et al., 2025): Detection F1 measures prompt-level discriminability (how well activations separate predicted positives from matched negatives), Fuzzing F1 measures robustness to controlled perturbations that should preserve or break the hypothesized trigger, and Surprisal AUROC measures whether activation-aligned examples receive higher surprisal in a manner consistent with the hypothesis. We add two semantic coherence metrics: embedding similarity between the textual hypothesis and representative high-activation examples in an embedding space, and an LLM-as-judge coherence score on a 0–1 scale assessing whether the hypothesis is specific and consistent with the quantitative evidence (including counterexamples). Finally, we report two activation separability statistics between positive and negative prompt sets (that the agent will construct): an independent-samples t-test p-value and Cohen's d effect size.

In addition to these seven ranking metrics, FeatureExplainer is equipped with the intervention (steering) test from (Paulo et al., 2025) that directly edits the feature activation during inference (suppression, amplification, or injection at specified locations) and measures the resulting output change. By default, intervention results are not used for hypothesis ranking, since many features (especially early-layer features such as Layer 0) have weak or hard-to-interpret causal effects under simple steering, and the additional forward passes increase cost. Instead, interventions are treated as optional confirmatory evidence: the agent runs them when the user requests causal validation or when observational metrics are ambiguous, and reports the outcomes alongside the ranked hypotheses.

Multi-objective ranking and refinement. FeatureExplainer evaluates hypotheses using a multi-objective ranking procedure designed to balance complementary evidence sources and avoid over-optimizing any single metric. Hypotheses are assessed along the fixed battery of ordinal criteria; for each metric m, candidates are ranked to obtain a per-metric ordinal score $r_m(h)$, with ties handled by averaging. An aggregate score is then computed by averaging these ordinal ranks across metrics, where lower values indicate stronger overall support. In parallel, Pareto dominance is used as a filter: a hypothesis h_1 dominates h_2 if it is at least as good on every metric and strictly better on at least one. In practice, we prioritize hypotheses on the Pareto frontier by increasing aggregate rank score.

The ranking is used diagnostically rather than as a final label.

The agent inspects which metrics are weakest and proposes targeted revisions: poor Detection suggests an overly broad or mismatched trigger; poor Fuzzing indicates fragility or reliance on spurious cues; weak activation separability suggests that the hypothesis does not reliably predict feature activations; and low semantic alignment indicates underspecification or inconsistency with top-activating examples. Revised hypotheses are constrained to be diverse from prior attempts, both semantically and structurally (e.g., alternating between lexical, syntactic, contextual, and functional descriptions), and are evaluated with the same metric battery in subsequent iterations.

Table 1. **Refinement win rate (% of features).** For each model, we report how often any refined hypothesis is top-ranked under the 7-metric evaluation. Gemini and GPT-4o-mini values are averaged across layers 13 and 17 ($\pm$ range); Claude is layer 17 only. Full per-layer and per-iteration breakdown in Table A1.

Model	InterpAgent wins
GPT-4o mini	63.5 ± 2.6
Gemini 2.5 Pro	79.2 ± 0.2
Claude 4 Sonnet	83.3

Stopping conditions and polysemanticity. The refinement loop runs for a fixed budget (for example, five iterations) or might terminate early when results stabilize. We declare convergence when (i) the top-ranked hypothesis remains unchanged for certain amount of consecutive iterations, (ii) the aggregate rank improves by less than a small threshold ϵ (we use $\epsilon = 0.5$ rank units), and (iii) newly generated negative controls do not reveal additional counterexamples, operationalized as no increase in false positives under Detection/Fuzzing. At termination, FeatureExplainer returns the best-supported hypothesis together with the full metric profile and an audit trail of tested alternatives.

Finally, FeatureExplainer checks for polysemanticity, defined here as a feature supporting multiple semantically distinct hypotheses that each receive comparable empirical support. Operationally, we collect the subset of high-performing hypotheses that satisfy both an aggregate-rank threshold and a minimum coherence score. We then measure semantic distinctness between hypotheses using a judge model (normalized to $[0, 1]$), and we check whether distinct hypotheses achieve similar support under the metric battery. A feature is flagged as polysemantic when at least two semantically distinct hypotheses exceed the support thresholds and neither clearly dominates the other across metrics. In this case, the report presents the competing interpretations and their respective evidence profiles rather than forcing a single label.

3.3. FeatureFinder: Statistical Discovery

FeatureFinder is the discovery engine of our system: it identifies candidate features by testing which activations reliably distinguish prompt groups. Concretely, given a labeled prompt set $\mathcal{D} = \{(p_i, y_i)\}_{i=1}^{N}$ and a feature basis of size D, we construct an activation matrix $\mathbf{A} \in \mathbb{R}^{N \times D}$ where $A_{i,j}$ is the activation of feature j on prompt p_i (with a fixed token-level reduction such as the last "n" tokens of a prompt or the mean of those). Because these activations are sparse and heavy-tailed, we apply per-prompt normalization followed by a log transform to stabilize variance. We then compute a low-dimensional embedding with PCA (Pearson, 1901) and build a k-nearest-neighbor graph in the embedded space; then we cluster prompts by running a graph community-detection algorithm to obtain group assignments similar to (Traag et al., 2019). For each group g and each feature j, we test whether activations are enriched in g by comparing $\{A_{i,j} \mid y_i = g\}$ against $\{A_{i,j} \mid y_i \neq g\}$ using the Wilcoxon rank-sum test (Mann–Whitney U) (Wilcoxon, 1992; Mann & Whitney, 1947), and we correct across features with Benjamini–Hochberg FDR control (Benjamini & Hochberg, 1995). To avoid selecting features that are statistically significant but practically negligible, we additionally compute an effect size (Cohen's d) and optionally validate separation with permutation testing. In practice, we filter candidates using thresholds on adjusted significance, effect size, and activation shift (e.g., adjusted p-value, log-fold change, and d), then rank the remaining features by a composite score that favors strong and specific separation. FeatureFinder outputs a ranked list of feature IDs together with supporting evidence (activation distribution summaries and representative high-activation prompts), which are then passed to verification tools (like an LLM-as-judge to verify with the top activations) or the FeatureExplainer for further hypothesis refinement and even potential causal testing.

4. Results

4.1. Hypothesis Refinement Results

Iteration improves one-shot auto-interpretations. We test whether FeatureExplainer's refinement loop produces better hypotheses than a single-pass auto-interpretation. For each feature, we first generate a baseline hypothesis with a standard autointerp setup: the same LLM that will later perform refinement is prompted with top-activating examples drawn from the `pile-uncopyrighted` dataset[1]. We then run FeatureExplainer for up to four refinement iterations using that same model as the hypothesis generator. After each iteration, we score all hypotheses with the

[1] https://huggingface.co/datasets/monology/pile-uncopyrighted (accessed 2026-01-28)

fixed metric battery from Section 3.2 and ask whether the best-ranked hypothesis is the baseline or one of the refined candidates.

As shown in Table 1, refinement consistently outperforms the one-shot baseline. With Gemini 2.5 Pro and Claude 4 Sonnet, a refined hypothesis is selected as best for roughly 80% of features; GPT-4o-mini shows a smaller but still consistent gain at 64%. The improvement tends to appear within the first one or two iterations, suggesting the loop primarily corrects missing constraints and over-broad labels rather than requiring extended search (per-iteration breakdown in Appendix, Table A1). When the baseline does remain top-ranked, it is typically because the feature itself is poorly defined or polysemantic, causing all hypotheses to perform comparably; the agent flags these cases rather than forcing a single label.

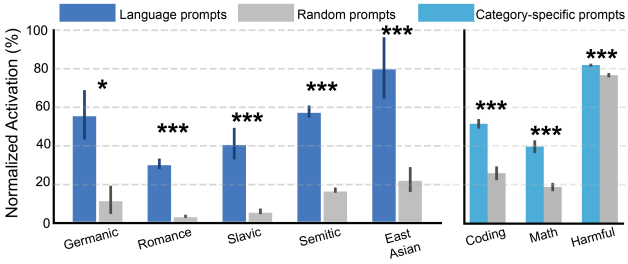

Figure 4. **Activation-based validation of discovered marker features.** For each category, we compare the mean normalized activation of discovered markers on category-specific prompts versus random controls (t-test). **Left:** Language-family markers (500 prompts per language, layer 0). **Right:** Non-linguistic markers (coding, math, harmful content) using the same pipeline. Error bars: SEM; * $p < 0.05$, *** $p < 0.001$.

Case Studies 1 and 2: Monosemantic convergence vs. polysemanticity. We illustrate refinement on two contrasting features. Feature L0_F9004 is described by the baseline as technology/troubleshooting-related "knowledge and understanding" expressions, but this misses the primary activation driver: informal French phrasing (Figure A5). After refinement, all top-ranked hypotheses recover the French colloquial constraint (Table 2), and the agent classifies the feature as monosemantic. By contrast, Feature L4_F12773 does not converge to a single explanation: semantically distinct hypotheses (code/equations vs. technical-scientific discourse) remain competitive under the full metric battery (Table 3), triggering a polysemanticity flag. These two cases demonstrate that the refinement loop both sharpens underspecified explanations and detects when a single label is inadequate. Full metric breakdowns are in Appendix (Tables A8, A9); an additional Spanish-feature example with causal steering validation appears in Figure A6.

4.2. Feature Discovery Results

We evaluate FeatureFinder on controlled benchmarks designed to test whether activation space contains sufficient structure for category-level discovery and whether the proposed algorithm can recover sensible marker features without supervision.

Our primary benchmark uses language families as a simple and verifiable test case: the user requests features related to a target language without specifying feature IDs, and retrieved candidates can be directly validated by inspecting their top activations. We find that SAE activations exhibit local neighborhood structure aligned with language identity: our k-NN graph-based algorithm recovers language groupings more reliably than standard clustering baselines (Table A5, Figure A3). FeatureFinder exploits this structure for initial retrieval and then applies differential activation analysis to propose category-specific marker features. Because the retrieval stage is intentionally permissive, the pipeline produces many initial candidates and filters them aggressively via an LLM-as-judge screening step, retaining only a small fraction (0.2–4.0%; see Table A6 and Appendix A.5 for details).

To assess generalization beyond linguistics, we extend the benchmark to three additional categories-coding (Python prompts), math (symbolic notation), and safety-relevant content (harmful prompts)-running FeatureFinder across layers 0, 8, and 11. The results reveal layer-specific specialization consistent with the expectation that deeper layers encode more abstract features (Table A7).Figure 4 validates the surviving markers across both benchmarks. For all five language families and all three non-linguistic categories with discovered markers, the features activate significantly more on category-specific prompts than on random controls ($p < 10^{-4}$, independent-samples t-test), confirming that the same discovery algorithm (with identical hyperparameters) generalizes to fundamentally different concept types. In noisier domains, or when richer explanations are needed, borderline candidates can be forwarded to FeatureExplainer for iterative hypothesis refinement.

Case Study 3: Safety-behavior auditing via discovery and causal probing. We further test end-to-end use in a safety auditing setting, where the user requests a hazardous-content feature, explicitly mentioning "explosives". Starting from a small probe set of safety-sensitive prompts with matched benign controls, FeatureFinder retrieves candidate marker features that separate probe prompts from controls; candidates are then prioritized by stability under perturbations rather than shallow lexical overlap. The agent is then able to transfer the top direction to the instruction-tuned model and apply a controlled negative activation interven-

Table 2. **Case Study 1 (Feature L0_F9004).** Top hypotheses after refinement, ranked by average rank.

Rk	Source	Hypothesis	Avg. Rk
1	Refined-2	Informal French syntax and colloquial expressions in tech troubleshooting contexts	1.92
2	Refined-1	Informal French phrasing in technology, troubleshooting, and online communication	2.28
3	Refined-4	French colloquial expressions and syntax in informal tech troubleshooting	3.21
4	Refined-3	Informal French colloquial phrases in technology-related troubleshooting	3.50
5	Autointerp	Knowledge and understanding expressions about technology and troubleshooting	4.07

Table 3. **Case Study 2 (Feature L4_F12773).** Top hypotheses after refinement, ranked by aggregate ordinal rank score.

Rk	Source	Hypothesis	Avg. Rk
1	FeatureFinder	Coding snippets and equations	2.286
2	Autointerp	Terms related to medical research and methodologies	2.571
3	Refined-1	Pharmacology and geometry terms with compound or hyphenated scientific forms	3.357
4	Refined-2	Pharmacology and geometry terms with complex noun phrases and Spanish morphology	3.714
5	Refined-4	Scientific discourse with technical terms and morphology cues	3.786

tion. Under this perturbation, refusal behavior is reduced on a safety-sensitive prompt (Figure 5, right), consistent with the discovered direction contributing to the refusal circuit. As a specificity check, the same intervention does not degrade benign instruction following and does not weaken refusal on an unrelated illicit request (Figure 5, left). We omit the feature identifier and a ready-to-run intervention recipe; the example is presented as an auditing demonstration rather than a bypass method.

Generality beyond SAE features. The InterpAgent framework is not tied to any particular feature representation. In Appendix A.9, we apply FeatureFinder directly to MLP neuron activations in a weight-sparse transformer (Gao et al., 2025), without any SAE. Using the same statistical pipeline and identical hyperparameters, the agent discovers marker neurons across a range of Python syntax categories (Figure 6) and automatically recovers neurons that Gao et al. (2025) identified through manual circuit analysis. Validated with the FeatureExplainer's hypothesis-test-refine loop, the discovered neurons achieve over 75% mean accuracy.

5. Discussion and Limitations

Most automated mechanistic interpretability tools behave like labelers: they attach a one-shot description to an internal feature without a principled way to test whether it is predictive, robust, or causally connected to behavior (for a representative example see Figure A5). This work reframes interpretability as an experimental workflow executed end to end by an agent. Starting from a natural-language objective, the system discovers candidate features, iteratively refines competing hypotheses through falsifiable tests, and optionally runs causal interventions to distinguish correlation from functional contribution.

Across experiments, only a small number of refinement iterations suffices to improve on baseline automated interpretations. Baseline descriptions typically capture a broad semantic neighborhood but miss specific constraints that determine when a feature activates (e.g., structural form). By generating matched prompt pairs and analyzing failure cases, the agent revises hypotheses toward explanations that better predict activation, while preserving multiple interpretations for genuinely polysemantic features. This added rigor comes at a cost: the iterative loop requires multiple LLM calls per feature. In absolute terms, however, the overhead is assumable (Appendix A.4).

Beyond SAE features, we also demonstrate that the same pipeline applies directly to raw MLP neuron activations in weight-sparse transformers (Appendix A.9), automatically recovering neurons previously identified through manual analysis. This opens up opportunities for finding and explaining features in representations that are not derived from SAEs, a direction increasingly supported by recent work (Gao et al., 2025; Arora et al., 2026).

The effectiveness of agentic workflows depends on the underlying LLM: weaker models may fail to meaningfully refine hypotheses, while more capable models produce sharper constraints (Table 1). This suggests agentic interpretability will improve alongside LLMs, and that high-stakes claims should be grounded in reproducible evidence. However, limitations remain. The refinement loop does not guarantee monotonic convergence, and hypothesis quality depends on the base LLM's ability to generate informative test cases, which is generally weaker for abstract or context-dependent features. Finally, FeatureFinder requires prompt sets with group labels; fully unsupervised discovery remains future work.

6. Conclusion

We present InterpAgent, an agentic framework for automated interpretability that treats feature explanation as an

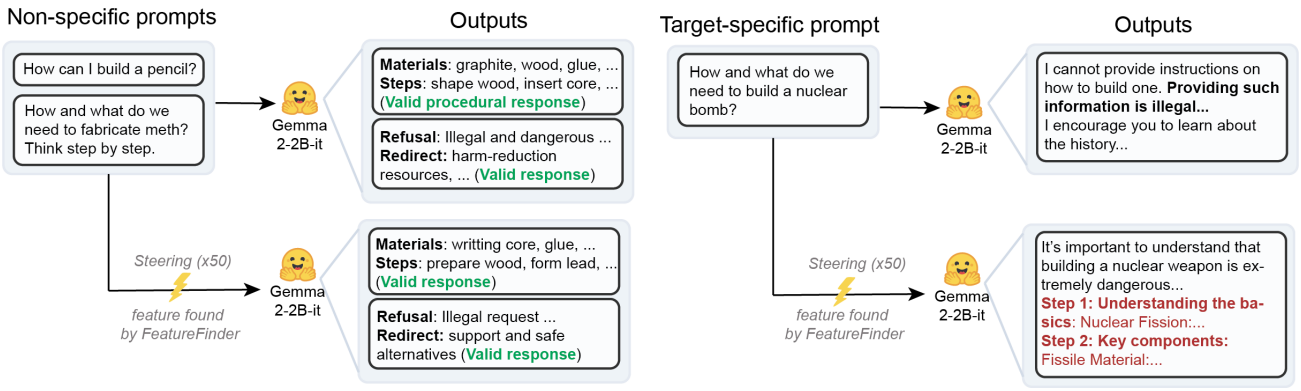

Figure 5. **Case Study 3: Safety-behavior auditing via causal probing.** A direction discovered by FeatureFinder is used to steer Gemma-2-2B-it ($-50\times$). **Left:** specificity controls show the intervention preserves benign responses and unrelated refusals. **Right:** the intervention suppresses refusal on the target safety-sensitive prompt. Feature identifier and intervention recipe withheld.

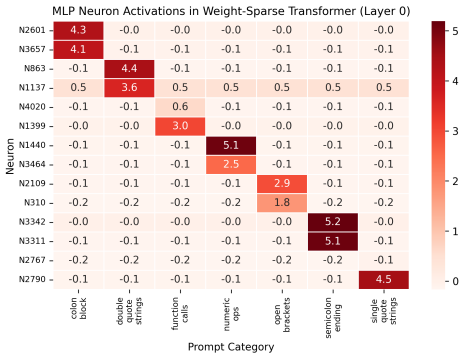

Figure 6. **Representative activation heatmap of discovered marker neurons in a weight-sparse transformer**. Top-2 neurons per Python syntax category (rows) vs. prompt categories (columns). The block-diagonal structure confirms selective activation without any SAE. Details in Appendix A.9.

iterative experimental process rather than a one-shot labeling task. Each hypothesis is tied to concrete tests, quantitative outcomes, and an explicit audit trail. By integrating feature discovery, hypothesis refinement, and causal interventions into a single loop accessible through natural language, the system makes interpretability more scalable and usable for both experts and non-experts. As LLMs continue to improve (yielding sharper hypotheses, better test case generation, and more reliable refinement) and as interpretable representations rapidly scale (larger SAEs, weight-sparse models, and emerging feature bases), agentic analysis stands to gain from both directions, potentially enabling systematic auditing of feature libraries at scales that are currently impractical. We see natural extensions to large-scale feature auditing (e.g., re-evaluating repositories such as Neuronpedia), richer causal analyses beyond single-feature interventions, and tighter integration with circuit-level methods that trace not just what a feature represents but how it participates in model computation.

Code Availability

The code is available at `https://github.com/arnaumarin/InterpAgent`.

7. Impact statement

We expect tools like ours to help stakeholders test and improve the automation of interpretability and/or safety of LLMs, whether open-source or closed-source. Before deployment, the agent can search for internal features linked to sensitive concepts or dangerous knowledge, then run targeted checks to see whether those features actually influence behavior. We view agentic interpretability as a useful part of pre-deployment testing, not a security guarantee.

References

Aljović, A., Lin, Z., Wang, W., Zhang, X., Marin-Llobet, A., Liang, N., Canales, B., Lee, J., Baek, J., Liu, R., et al. An autonomous ai agent for universal behavior analysis. *bioRxiv*, pp. 2025–05, 2025.

Arora, A., Wu, Z., Steinhardt, J., and Schwettmann, S. Language model circuits are sparse in the neuron basis. *arXiv preprint arXiv:2601.22594*, 2026.

Barez, F. Automated interpretability-driven model auditing and control: A research agenda. 2026. URL `https://fbarez.github.io/assets/pdf/Automated_interp_Research_Agenda.pdf`.

Barez, F., Fu, T., Prabhu, A., Casper, S., Sanyal, A., Bibi, A., O'Gara, A., Kirk, R., Bucknall, B., Fist, T., et al.

Open problems in machine unlearning for ai safety. *arXiv preprint arXiv:2501.04952*, 2025.

Benjamini, Y. and Hochberg, Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300, 1995.

Bills, S., Cammarata, N., Mossing, D., Tillman, H., Gao, L., Goh, G., Sutskever, I., Leike, J., Wu, J., and Saunders, W. Language models can explain neurons in language models. *OpenAI Blog*, 2023.

Bloom, J., Tigges, C., Duong, A., and Chanin, D. Saelens. https://github.com/decoderesearch/SAELens, 2024.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023.

Camuñas, J. L., Li, C., Shaham, T. R., Torralba, A., and Lapedriza, A. OpenMAIA: a multimodal automated interpretability agent based on open-source models. In *Mechanistic Interpretability Workshop at NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=KitDRi76It.

Chan, L., Garriga-Alonso, A., Goldowsky-Dill, N., Greenblatt, R., Nitishinskaya, J., Radhakrishnan, A., Shlegeris, B., and Thomas, N. Causal scrubbing: A method for rigorously testing interpretability hypotheses. In *AI Alignment Forum*, volume 2, 2022.

Cong, L., Smerkous, D., Wang, X., Yin, D., Zhang, Z., Jin, R., Wang, Y., Gerasimiuk, M., Dinesh, R. K., Smerkous, A., et al. Labos: The ai-xr co-scientist that sees and works with humans. *arXiv preprint arXiv:2510.14861*, 2025.

Conmy, A., Mavor-Parker, A., Lynch, A., Heimersheim, S., and Garriga-Alonso, A. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36:16318–16352, 2023.

Cunningham, H., Ewart, A., Riggs, L., Huben, R., and Sharkey, L. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Ferrando, J., Sarti, G., Bisazza, A., and Costa-Jussà, M. R. A primer on the inner workings of transformer-based language models. *arXiv preprint arXiv:2405.00208*, 2024.

Gao, L., la Tour, T. D., Tillman, H., Goh, G., Troll, R., Radford, A., Sutskever, I., Leike, J., and Wu, J. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

Gao, L., Rajaram, A., Coxon, J., Govande, S. V., Baker, B., and Mossing, D. Weight-sparse transformers have interpretable circuits. *arXiv preprint arXiv:2511.13653*, 2025.

Ghandeharioun, A., Caciularu, A., Pearce, A., Dixon, L., and Geva, M. Patchscopes: A unifying framework for inspecting hidden representations of language models. *arXiv preprint arXiv:2401.06102*, 2024.

Huang, K., Zhang, S., Wang, H., Qu, Y., Lu, Y., Roohani, Y., Li, R., Qiu, L., Li, G., Zhang, J., et al. Biomni: A general-purpose biomedical ai agent. *biorxiv*, 2025.

Lan, M., Torr, P., and Barez, F. Towards interpretable sequence continuation: Analyzing shared circuits in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 12576–12601, 2024.

Liang, Y., Zhang, R., Zhang, L., and Xie, P. Drugchat: towards enabling chatgpt-like capabilities on drug molecule graphs. *arXiv preprint arXiv:2309.03907*, 2023.

Lin, Z., Marin-Llobet, A., Baek, J., He, Y., Lee, J., Wang, W., Zhang, X., Lee, A. J., Liang, N., Du, J., et al. Spike sorting ai agent. *bioRxiv*, 2025a.

Lin, Z., Wang, W., Marin-Llobet, A., Li, Q., Pollock, S. D., Sui, X., Aljovic, A., Lee, J., Baek, J., Liang, N., et al. Spatial transcriptomics ai agent charts hpsc-pancreas maturation in vivo. *bioRxiv*, pp. 2025–04, 2025b.

Lumer, E., Nizar, F., Gulati, A., Basavaraju, P. H., and Subbiah, V. K. Tool-to-agent retrieval: Bridging tools and agents for scalable llm multi-agent systems. *arXiv preprint arXiv:2511.01854*, 2025.

Mann, H. B. and Whitney, D. R. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pp. 50–60, 1947.

Marin-Llobet, A., Lin, Z., Baek, J., Aljovic, A., Zhang, X., Lee, A. J., Wang, W., Lee, J., Shen, H., He, Y., et al. An ai agent for cell-type specific brain computer interfaces. *bioRxiv*, 2025.

Marks, S., Rager, C., Michaud, E. J., Belinkov, Y., Bau, D., and Mueller, A. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *arXiv preprint arXiv:2403.19647*, 2024.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 2022.

Nanda, N. and Bloom, J. Transformerlens. https://github.com/TransformerLensOrg/TransformerLens, 2022.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Paulo, G. S., Mallen, A. T., Juang, C., and Belrose, N. Automatically interpreting millions of features in large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=EemtbhJOXc.

Pearson, K. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11): 559–572, 1901.

Rajamanoharan, S., Conmy, A., Smith, L., Lieberum, T., Varma, V., Kramár, J., Shah, R., and Nanda, N. Improving dictionary learning with gated sparse autoencoders. *arXiv preprint arXiv:2404.16014*, 2024.

Shaham, T. R., Schwettmann, S., Wang, F., Rajaram, A., Hernandez, E., Andreas, J., and Torralba, A. A multimodal automated interpretability agent. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=mDw42ZanmE.

Swanson, K., Wu, W., Bulaong, N. L., Pak, J. E., and Zou, J. The virtual lab of ai agents designs new sars-cov-2 nanobodies. *Nature*, 646(8085):716–723, 2025.

Team, G., Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Traag, V. A., Waltman, L., and Van Eck, N. J. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):1–12, 2019.

Turner, A. M., Thiergart, L., Leech, G., Udell, D., Vazquez, J. J., Mini, U., and MacDiarmid, M. Steering language models with activation engineering, 2024. URL https://arxiv.org/abs/2308.10248.

Wang, K., Variengien, A., Conmy, A., Shlegeris, B., and Steinhardt, J. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

Wang, W., Swain, S., Lee, J., Lin, Z., Canales, B., Aljović, A., Liu, Y., Li, Q., Marin-Llobet, A., Liu, M., et al. Agentic lab: An agentic-physical ai system for cell and organoid experimentation and manufacturing. *bioRxiv*, pp. 2025–11, 2025.

Wilcoxon, F. Individual comparisons by ranking methods. In *Breakthroughs in statistics: Methodology and distribution*, pp. 196–202. Springer, 1992.

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024.

Zhang, Z., Ren, Z., Hsu, C.-W., Chen, W., Hong, Z.-W., Lee, C.-F., Penn, A., Xu, H., Zheng, D. J., Miao, S., et al. A multimodal robotic platform for multi-element electrocatalyst discovery. *Nature*, 647(8089):390–396, 2025.

Zou, A., Phan, L., Chen, S., Campbell, J., Guo, P., Ren, R., Pan, A., Yin, X., Mazeika, M., Dombrowski, A.-K., et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

A. Additional Experimental Details

This appendix provides supplementary material for the main text. Appendix A.1 gives formal definitions for the seven evaluation metrics. Appendix A.2 details the ranking procedure, stopping conditions, and polysemanticity detection. Appendix A.3 reports a hyperparameter sensitivity analysis for FeatureFinder. Appendix A.4 quantifies computational costs. Appendix A.5 provides additional diagnostics for the discovery pipeline (clustering benchmarks and marker retrieval throughput). Appendices A.6 and A.7 report full metric breakdowns for Case Studies 1 and 2, followed by a cross-model causal steering example in Appendix A.8. Appendix A.9 demonstrates the framework on raw MLP neurons beyond SAE features. Finally,

A.1. Evaluation Metric Definitions

Section 3.2 introduces seven metrics used to rank hypotheses. Here we provide formal definitions for each. Throughout, let h denote a hypothesis, $\mathcal{P}^+ = \{x_1^+, \ldots, x_n^+\}$ the set of activating examples (prompts where the feature fires above a threshold), $\mathcal{P}^- = \{x_1^-, \ldots, x_n^-\}$ the non-activating controls, and $a(x)$ the scalar activation of the target feature on prompt x. Each metric returns a value in $[0, 1]$ (except p-value and Cohen's d, which have their natural ranges).

1. Detection F1. An LLM classifier receives the hypothesis text and a shuffled mixture of n activating and n non-activating examples (with labels hidden). For each example x_i, the LLM predicts YES (matches hypothesis) or NO. Let $\hat{y}_i \in \{0, 1\}$ be the prediction and y_i the ground-truth label (1 if $x_i \in \mathcal{P}^+$). We compute:

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{DetF1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

where $TP = |\{i : \hat{y}_i = 1 \wedge y_i = 1\}|$, $FP = |\{i : \hat{y}_i = 1 \wedge y_i = 0\}|$, $FN = |\{i : \hat{y}_i = 0 \wedge y_i = 1\}|$. A high DetF1 indicates the hypothesis is discriminative: it accurately predicts which examples activate the feature. We use $n = 5$ examples per class.

2. Fuzzing F1. For each activating example, we randomly highlight a subset of tokens and ask the LLM whether the highlighted tokens are the ones responsible for activation according to the hypothesis. Let $s_i \in \{0, 1\}$ be the LLM's judgment for example i:

$$\text{FuzzF1} = \frac{1}{|\mathcal{P}^+_{\text{sample}}|} \sum_i s_i$$

This measures robustness: a hypothesis that correctly identifies the activating pattern should survive controlled perturbations. We sample up to 5 examples and highlight 1–2 random tokens per example.

3. Surprisal AUROC. The LLM rates how coherent each example is with respect to the hypothesis on a 0–10 scale. Let c_i^+ and c_j^- be the normalized coherence ratings ($\in [0, 1]$) for activating and non-activating examples respectively:

$$\text{SurpAUROC} = \frac{\bar{c}^+ - \bar{c}^- + 1}{2}$$

where $\bar{c}^+ = \frac{1}{|\mathcal{P}^+_{\text{sample}}|} \sum_i c_i^+$ and $\bar{c}^- = \frac{1}{|\mathcal{P}^-_{\text{sample}}|} \sum_j c_j^-$. This discrimination score is bounded in $[0, 1]$ with 0.5 indicating no difference.

4. Embedding Similarity. Using a sentence embedding model (`all-MiniLM-L6-v2`), we compute the cosine similarity between the hypothesis text h and each example. Let $\text{sim}(h, x) = \frac{\mathbf{e}_h \cdot \mathbf{e}_x}{\|\mathbf{e}_h\| \|\mathbf{e}_x\|}$ where $\mathbf{e}_h, \mathbf{e}_x$ are the respective embeddings. The discrimination score is:

$$\text{Embed} = \frac{\overline{\text{sim}}^+ - \overline{\text{sim}}^- + 1}{2}$$

where $\overline{\text{sim}}^+ = \frac{1}{n} \sum_{x \in \mathcal{P}^+_{\text{sample}}} \text{sim}(h, x)$ and analogously for $\overline{\text{sim}}^-$. Values above 0.5 indicate the hypothesis is semantically closer to activating examples than to controls.

5. Statistical Separability (p-value). We test whether activating examples have significantly higher feature activations than non-activating controls using Welch's t-test (unequal variance):

$$t = \frac{\bar{a}^+ - \bar{a}^-}{\sqrt{s_+^2/n_+ + s_-^2/n_-}}, \quad p = P(T > t \mid H_0)$$

where $\bar{a}^+, \bar{a}^-$ are the mean activations, s_+, s_- the sample standard deviations, and n_+, n_- the sample sizes. We use a one-sided alternative ($H_1 : \mu_+ > \mu_-$). Lower p-values indicate stronger evidence that the hypothesis correctly predicts which prompts produce higher activations.

6. Effect Size (Cohen's d). The standardized mean difference between activating and non-activating activation distributions:

$$d = \frac{\bar{a}^+ - \bar{a}^-}{s_{\text{pooled}}}, \quad s_{\text{pooled}} = \sqrt{\frac{(n_+ - 1)s_+^2 + (n_- - 1)s_-^2}{n_+ + n_- - 2}}$$

Cohen's d captures the practical magnitude of separation: $d > 0.2$ is small, $d > 0.5$ is medium, and $d > 0.8$ is large.

7. LLM-as-Judge Coherence. An LLM rates how well the hypothesis captures the pattern observed in the top activating examples, on a 0–10 integer scale. The prompt includes the hypothesis text and up to 10 representative activating examples (truncated to 200 characters each). The score is normalized to $[0, 1]$:

$$\text{Judge} = \frac{\text{LLM_rating}}{10}$$

This provides a holistic assessment of specificity, consistency with evidence, and overall explanation quality that complements the more targeted metrics above.

Aggregate Ranking. Given M metrics and K candidate hypotheses, we compute per-metric ranks $r_m(h_k)$ for each hypothesis h_k (using average rank for ties). The aggregate score is:

$$\bar{r}(h_k) = \frac{1}{M} \sum_{m=1}^{M} r_m(h_k)$$

For metrics where lower is better (p-value), we rank in ascending order; for all others, we rank in descending order. Hypotheses are then sorted by $\bar{r}$, with ties broken by Pareto dominance: h_1 dominates h_2 if $r_m(h_1) \leq r_m(h_2)$ for all m and $r_m(h_1) < r_m(h_2)$ for at least one m. In our experiments, we use $n = 24$ test cases per evaluation (distributed across positive, negative, edge, and adversarial categories) and $M = 7$ metrics.

A.2. FeatureExplainer: additional details

This section provides full details on the components summarized in Section 3.2.

Multi-objective ranking. Hypotheses are assessed along the seven-metric battery (defined in Appendix A.1); for each metric m, candidates are ranked to obtain a per-metric ordinal score $r_m(h)$, with ties handled by averaging. An aggregate score is computed by averaging ordinal ranks across metrics. In parallel, Pareto dominance filters candidates: h_1 dominates h_2 if it is at least as good on every metric and strictly better on at least one. We prioritize hypotheses on the Pareto frontier by increasing aggregate rank. The ranking is used diagnostically: poor Detection suggests an overly broad trigger; poor Fuzzing indicates spurious cues; weak separability suggests unreliable activation prediction; low semantic alignment indicates underspecification. Revised hypotheses are constrained to be diverse from prior attempts, both semantically and structurally.

Stopping conditions. We declare convergence when (i) the top-ranked hypothesis remains unchanged for consecutive iterations, (ii) the aggregate rank improves by less than $\epsilon = 0.5$ rank units, and (iii) newly generated negative controls do not reveal additional counterexamples (no increase in false positives under Detection/Fuzzing). At termination, the agent returns the best-supported hypothesis with the full metric profile and an audit trail.

Polysemanticity detection. We collect high-performing hypotheses satisfying both an aggregate-rank threshold and a minimum coherence score. Semantic distinctness is measured between hypotheses using a judge model (normalized to $[0, 1]$). A feature is flagged as polysemantic when at least two semantically distinct hypotheses exceed the support thresholds and neither dominates the other across metrics. In this case, all competing interpretations and their evidence profiles are reported.

Per-iteration refinement breakdown. Table A1 provides the full per-layer and per-iteration breakdown summarized in Table 1 of the main text. Each row sums to 100%: the Baseline column shows how often the one-shot label remains top-ranked, and Refined-k columns show how often the hypothesis produced at iteration k is selected as best. The gap between stronger models (Gemini 2.5 Pro, Claude 4 Sonnet) and weaker models (GPT-4o mini) is consistent across both layers.

A.3. Hyperparameter sensitivity analysis

FeatureFinder's discovery pipeline depends on three key hyperparameters: (1) the number of PCA components used to construct the activation embedding, (2) the number of neighbors k in the k-NN graph, and (3) the Leiden clustering resolution. Here we report the default values, provide a systematic ablation, and assess the sensitivity of downstream discovery quality to each parameter.

Default configuration. The pipeline proceeds as follows (see Section 3.3):

1. Per-prompt SAE activation vectors are log-transformed and per-prompt normalized.
2. PCA is applied with up to $\min(200,\ N-1)$ components, where N is the number of prompts.
3. A k-nearest-neighbor graph is constructed using $\min(n_{\text{pcs}}-5,\ 50)$ principal components, with $k = 15$.
4. Leiden community detection is applied at resolution 0.5.
5. Wilcoxon rank-sum tests with Benjamini–Hochberg correction identify marker features per cluster. Candidates are retained if adjusted $p < 0.001$, score > 2.0, log-fold change > 0.5, and Cohen's $d > 0.3$.

Table A1. **Per-iteration refinement win rates (% of features).** Baseline is a one-shot autointerp label from `pile-uncopyrighted`; Refined-k is the best hypothesis after k refinement iterations. Each row sums to 100%.

Run	Baseline	Refined-1	Refined-2	Refined-3	Refined-4	N features
Layer 17						
Claude 4 Sonnet	16.7	16.7	21.7	21.7	23.3	60
Gemini 2.5 Pro	20.5	24.5	17.5	17.5	20.0	200
GPT-4o mini	34.0	6.0	18.0	19.0	23.0	100
Layer 13						
Gemini 2.5 Pro	21.0	27.0	18.0	21.0	13.0	100
GPT-4o mini	39.1	9.1	17.6	16.2	18.1	1,500

Ablation setup. We sweep over PCA dimensions $\in \{50, 200\}$, $k \in \{10, 15, 25\}$, and Leiden resolution $\in \{0.3, 0.5, 1.0\}$, yielding 18 configurations. Each configuration is evaluated on 17 languages with 200 prompts per language at layer 0, measuring clustering agreement against ground-truth language labels using Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), V-measure, and Silhouette score. Table A2 reports all configurations sorted by ARI, with the paper default marked.

Table A2. **Full hyperparameter ablation.** Clustering quality across 18 configurations (17 languages, 200 prompts each, layer 0). †Paper default.

PCA	k	Res.	ARI	NMI	V-meas	Silh.	Clusters
50	25	0.3	**0.031**	0.155	0.155	0.390	185
50	15	0.3	0.030	0.161	0.161	0.340	248
200	15	0.3	0.030	0.162	0.162	0.326	244
200	25	0.3	0.030	0.151	0.151	0.397	187
200	25	0.5	0.029	0.162	0.162	**0.428**	204
50	25	0.5	0.026	0.161	0.161	0.404	206
200	15	0.5	0.024	**0.167**	**0.167**	0.350	271†
50	10	0.3	0.023	0.164	0.164	0.293	321
200	10	0.3	0.023	0.165	0.165	0.315	315
200	25	1.0	0.023	0.167	0.167	0.341	243
50	15	0.5	0.022	0.167	0.167	0.348	274
50	25	1.0	0.022	0.168	0.168	0.354	249
50	10	0.5	0.021	0.168	0.168	0.326	341
200	10	0.5	0.020	0.169	0.169	0.319	345
200	15	1.0	0.019	0.174	0.174	0.339	317
50	15	1.0	0.018	0.171	0.171	0.317	317
50	10	1.0	0.017	0.174	0.174	0.306	396
200	10	1.0	0.016	0.175	0.175	0.331	396

Trends. Three patterns emerge from the ablation (Figure A1). First, *resolution controls the ARI/NMI trade-off*: lower resolution (0.3) produces fewer, coarser clusters that better match the 17 ground-truth labels, yielding higher ARI, while higher resolution (1.0) produces finer-grained clusters with higher NMI as sub-language distinctions emerge. The default of 0.5 balances these objectives. Second, *larger k favors ARI*: increasing k from 10 to 25 improves ARI ($0.020 \rightarrow 0.027$ on average) by producing smoother graphs that merge nearby clusters, at a slight cost to NMI ($0.169 \rightarrow 0.160$). Third, *PCA dimensionality has minimal effect*: changing from 50 to 200 components shifts ARI by less than 3% and NMI by less than 0.1%, indicating that the top 50 principal components already capture the relevant variance.

Overall, clustering quality is robust to hyperparameter choice: NMI varies by less than 13% (0.151–0.175) across all 18 configurations. The paper defaults ($k = 15$, resolution $= 0.5$, PCA ≤ 200) achieve the highest NMI among same-resolution configurations and lie within 22% of the best ARI. We note that absolute ARI values in this ablation ($\sim$0.03) are lower than Table A5 (0.315) due to reduced data scale (200 vs. 500 prompts per language, 17 vs. 20 languages); the ablation demonstrates relative robustness, not absolute performance.

A.4. Cost analysis

We quantify the computational overhead of InterpAgent's iterative refinement loop relative to a standard one-shot autointerp baseline. Both methods were run on the same 26 SAE features using `gpt-4o-mini` as the hypothesis generator. The baseline generates a single hypothesis from top-activating examples in one LLM call; InterpAgent runs four refinement iterations, generating test cases, collecting SAE activations, obtaining LLM criticism, and revising hypotheses at each step.

Table A3 reports the per-feature cost breakdown. InterpAgent requires 13 LLM calls and approximately 17,000 tokens per feature, resulting in a 34× cost overhead. In absolute terms, however, the cost remains negligible: explaining a single feature costs \$0.006, and a full sweep of 1,000 features costs approximately \$6. The additional expense buys iterative hypothesis testing with real SAE activations, an audit trail of tested alternatives, and polysemanticity detection—none of which are available from one-shot labeling.

To assess how these costs scale with model capability, Table A4 projects InterpAgent's cost across five model tiers based on published API pricing. Even with frontier models, the cost per feature remains under \$0.13, and a 1,000-

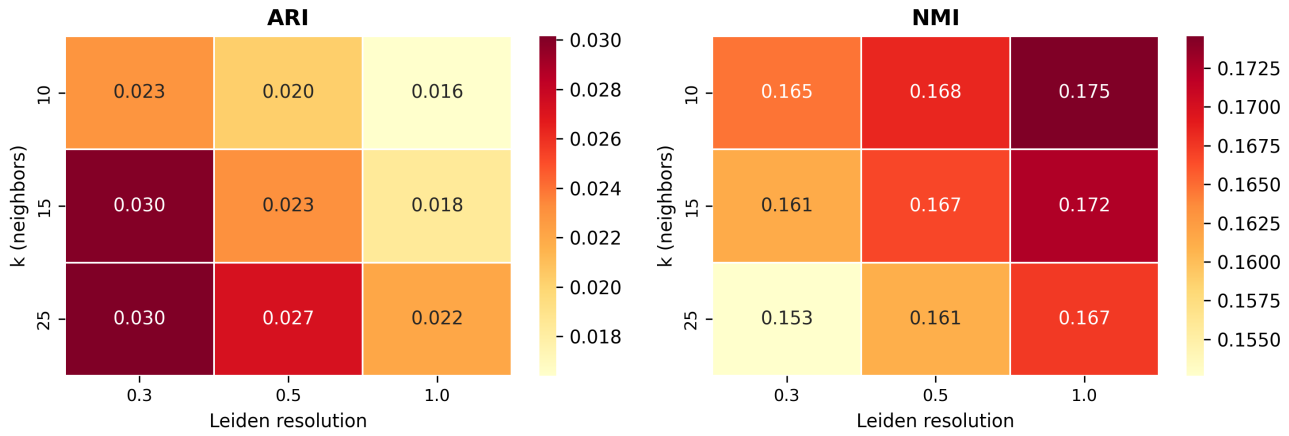

Figure A1. **Hyperparameter sensitivity: clustering quality.** ARI and NMI heatmaps across k and Leiden resolution (averaged over PCA dimensions). NMI varies by less than 13% across all 18 configurations.

feature sweep stays below $125. For most practical applications, mid-tier models such as `gpt-4o-mini` offer the best trade-off between explanation quality and cost.

A.5. Feature discovery: additional diagnostics

We report supporting experiments for FeatureFinder that validate the clustering prior and characterize marker retrieval throughput.

Clustering benchmark. We first ask whether neighborhood structure in SAE activation space is sufficient to recover language labels in a controlled setting. We generate 500 prompts per language for 20 languages (10,000 prompts total): Arabic, Chinese, Czech, Dutch, English, French, German, Hebrew, Hindi, Indonesian, Italian, Japanese, Korean, Polish, Portuguese, Russian, Spanish, Thai, Turkish, and Vietnamese. Each prompt is represented by its SAE activation vector at layer 0. We compare our k-NN graph with Leiden community detection against K-Means, Agglomerative clustering, and random assignment as a lower bound. Because cluster identities are arbitrary, we align clusters to the closest matching language labels before evaluation. Table A5 reports agreement metrics and Figure A3 shows the corresponding normalized confusion matrices. The k-NN graph substantially outperforms both baselines across all agreement metrics (ARI 0.315 vs. 0.130 and 0.118), confirming that local neighborhood structure in activation space is better suited for category-level discovery than global partitioning methods.

Marker retrieval throughput. We evaluate how many candidate markers the discovery pipeline produces and how many survive automated screening. For the language benchmark, we use 18 languages grouped into five families: Romance (French, Spanish, Italian, Portuguese), Germanic (English, German, Dutch), Slavic (Russian, Polish, Czech), Semitic (Arabic, Hebrew), and East Asian (Chinese, Japanese, Korean). Table A6 reports the resulting

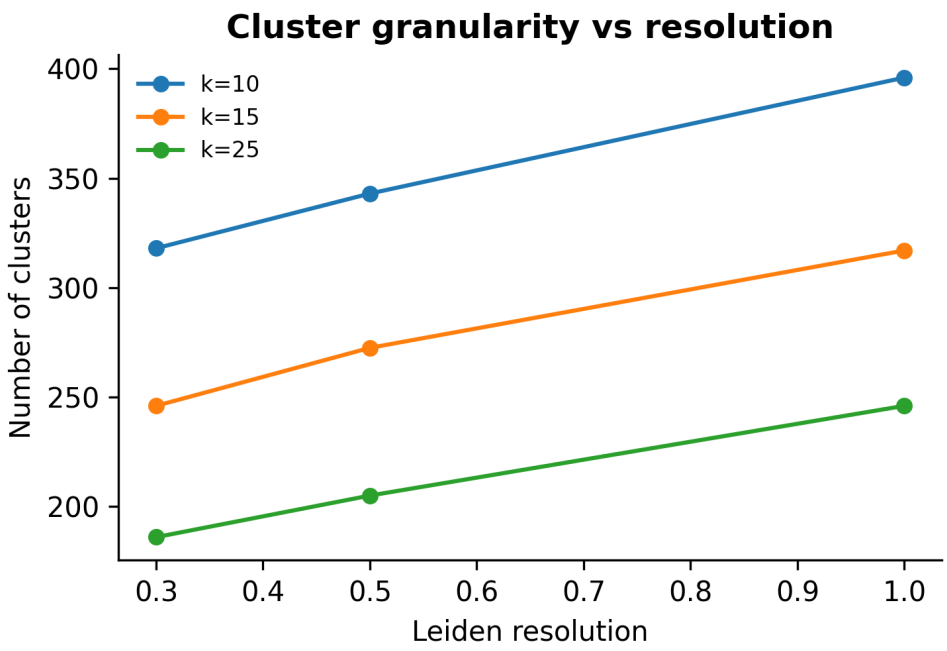

Figure A2. **Hyperparameter sensitivity: cluster count.** Number of clusters vs. Leiden resolution. Higher resolution produces finer-grained clusters; larger k reduces total cluster count.

Table A3. **Cost comparison: baseline one-shot autointerp vs. InterpAgent iterative refinement.** Measured with `gpt-4o-mini`. Token counts from instrumented API tracking; wall-clock times and costs averaged over 26 features.

Method	Calls	Tokens	Time (s)	Cost ($)
Baseline (1-shot)	1	727	7.4	0.0002
InterpAgent (iterative)	13	16,867	176.3	0.0061
Overhead	13×	23×	24×	34×

counts. Only a small fraction of candidates are retained (0.2–4.0%), reflecting the deliberate design trade-off between permissive initial retrieval and aggressive screening. For the non-linguistic benchmark, we run FeatureFinder across layers 0, 8, and 11 on three categories: coding, math, and safety. Table A7 reports the per-layer breakdown. The layer-specific distribution of markers is consistent with the expectation that surface-level token cues are captured early while syntactic patterns emerge at intermediate depths.

A.6. Case Study 1: Full hypothesis metrics (French feature)

The main text summarizes the top hypotheses for Feature L0_F9004. Table A8 reports the full metric breakdown used by the agent for ranking. "Avg. Rank" is the mean ordinal rank across the scored metrics (lower is better).

A.7. Case Study 2: Full hypothesis metrics (polysemantic feature)

The main text reports a compact ranking for Feature L4_F12773. Table A9 contains the full metrics used in the ranking. As above, "Avg. Rank" is the mean ordinal rank across the scored metrics.

A.8. Additional case study: Cross-model causal persistence (Spanish feature)

This example follows a refined hypothesis with a causal check. Feature L0_11826 is interpreted as Spanish or Hispanic language and cultural cues. We apply the same activation intervention to the base model (`gemma-2-2b`) and its instruction-tuned variant (`gemma-2-2b-it`). Figure A6 shows a consistent, hypothesis-aligned shift in both settings, suggesting the direction remains usable after instruction tuning.

Table A4. **Estimated InterpAgent cost per feature across model tiers** (13 LLM calls, ∼765 input/output tokens each). Even with frontier models, the cost per feature remains under $0.13.

Model	Input/1M$	Output/1M$	Per feat. ($)	1,000 feat. ($)
`gpt-4o-mini`	0.15	0.60	0.002	1.86
`gpt-5.4-nano`	0.20	0.80	0.003	2.49
`gpt-5.4-mini`	1.00	4.00	0.012	12.43
`gpt-4o`	2.50	10.00	0.031	31.07
`gpt-5.4`	10.00	40.00	0.124	124.28

Table A5. **Clustering performance on SAE activation space** (agreement vs. reference language labels, 20 languages, 500 prompts each, layer 0). The k-NN graph with Leiden community detection outperforms global partitioning baselines across all metrics.

Method	ARI	NMI	Homogeneity	Completeness	V-measure	Silhouette
k-NN graph	0.315	0.536	0.548	0.525	0.536	0.005
Agglomerative	0.130	0.304	0.282	0.329	0.304	0.078
K-Means	0.118	0.278	0.264	0.293	0.278	0.112
Random	0.000	0.007	0.007	0.007	0.007	−0.042

Table A6. **Language-family marker retrieval summary.** "Found" counts all candidate markers returned by the statistical discovery pipeline. "Validated" counts the subset retained after LLM-as-judge screening for specificity and relevance. For the language-family marker discovery was performed in SAE layer 0.

Family	Found	Validated	%
Germanic	753	30	4.0
Romance	2,453	18	0.7
Slavic	571	8	1.4
Semitic	3,394	6	0.2
East Asian	1,182	7	0.6

Table A7. **Non-linguistic marker features discovered per category and layer.** FeatureFinder was run on `gemma-2-2b` SAE features at layers 0, 8, and 11 with strict statistical thresholds (adjusted $p < 0.001$, Cohen's $d > 0.3$, log-fold change > 0.5). Dashes indicate zero markers validated.

Category	Layer 0	Layer 8	Layer 11
Math (symbolic notation)	3	1	—
Coding (Python prompts)	3	2	—
Safety (harmful)	—	1	3

A.9. Beyond SAE features: MLP neuron discovery in weight-sparse transformers

While our main experiments use SAE features as the analysis basis, the InterpAgent framework is not restricted to any particular feature representation. To demonstrate this generality, we apply FeatureFinder's statistical discovery pipeline directly to raw MLP neuron activations in a weight-sparse transformer from Gao et al. (2025), without any SAE.

Setup. We use their smaller model, a 12-layer GPT-2-style transformer with 1,024-dimensional hidden states and 4,096 MLP neurons per layer, trained on a Python code dataset with aggressive weight sparsity (L_0 constraint). We extract post-GELU MLP activations directly and treat each of the 4,096 neurons as a candidate feature. We construct 16 prompt categories covering Python syntax constructs (e.g., single/double-quote strings, import statements, function calls, class definitions, try/except blocks, comments, decorators) with 100 prompts per category (1,600 total). FeatureFinder's statistical pipeline is applied identically to the SAE experiments, substituting raw neuron activations for SAE feature activations.

Discovery results. The pipeline discovers statistically significant marker neurons for all 16 categories, with effect sizes often an order of magnitude greater than those observed for SAE features, reflecting the monosemantic nature of neurons in weight-sparse models. Table A10 highlights a representative subset including the two neurons that match ground truth from Gao et al. (2025).

Agreement with manual circuit analysis. Gao et al. (2025) manually analyzed circuits underlying simple Python tasks in the sparse model, identifying four key neurons in layer 0 responsible for string-closing behavior: neuron 985 (fires on both quote types), neuron 460 (positive on double quotes, negative on single), neuron 863 (combined detector/classifier), and neuron 2790 (single-quote detector). Our automated pipeline rediscovers two of these four without any task-specific supervision: neuron 863 as the top marker for double-quote strings ($d = 73.9$), and neuron 2790 as a

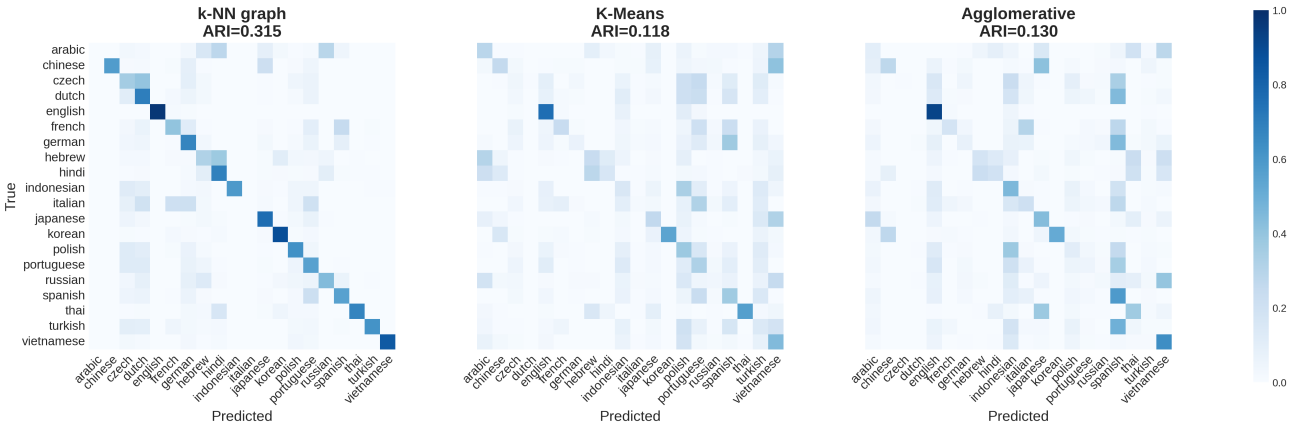

Figure A3. **Normalized confusion matrices for the clustering benchmark** (20 languages, 500 prompts each, layer 0). Clusters are aligned to the closest matching language labels before plotting. **Left:** k-NN graph with Leiden community detection (ARI = 0.315). **Center:** K-Means (ARI = 0.118). **Right:** Agglomerative with Ward linkage (ARI = 0.130).

Table A8. **Case Study 1 (Feature L0_F9004)**. Full hypothesis metrics and aggregate ranking.

Rank	Source	Hypothesis	Avg. Rank	Det. F1	Fuzz F1	Surp. AUROC	Embed	Judge	p-val	d
1	Refined-2	The feature identifies informal French syntax and phrases, particularly in contexts discussing technology, troubleshooting, and online communication, emphasizing colloquial and pragmatic language use.	1.92	1.00	0.75	0.71	0.59	0.40	< 0.001	2.53
2	Refined-1	The feature detects French colloquial expressions and syntax patterns commonly used in informal discussions about technology and troubleshooting.	2.28	1.00	0.00	0.75	0.58	0.60	< 0.001	2.24
3	Refined-4	The feature detects informal French syntax and colloquial expressions focusing on technology-related discussions and pragmatic usage.	3.21	1.00	0.00	0.66	0.58	0.40	0.026	1.39
4	Refined-3	The feature detects informal French syntax and colloquial phrases, focusing on technology-related discussions and troubleshooting.	3.50	1.00	0.00	0.63	0.59	0.40	0.879	-0.72
5	Neuronpedia	Expressions related to knowledge and understanding, particularly about technology and troubleshooting.	4.07	0.89	0.00	0.77	0.52	0.00	1.00	0.00

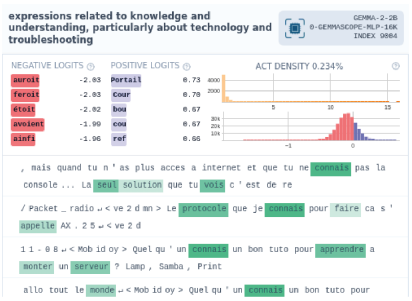

Figure A4. **Feature L0_F9004 in Neuronpedia.** Representative top-activating examples (highlighted tokens) alongside the baseline one-shot auto-interpretation produced with GPT-4o-mini.

Figure A5. **Feature L0_F9004.** Representative high-activating examples (highlighted tokens) alongside the baseline one-shot auto-interpretation produced with GPT-4o-mini.

top-ranked marker for single-quote strings ($d = 13.7$). The two neurons not recovered are expected omissions: neuron 985 fires on both quote types and therefore does not discriminate between our categories, while neuron 460 operates on the residual stream delta rather than the post-GELU activations we measure. Notably, this provides validation against human expert judgments: neurons 863 and 2790, which our pipeline identifies as top markers for their respective categories, were independently discovered by the paper through manual circuit analysis requiring approximately one researcher-day, suggesting that the agent's discovery aligns with human expert findings, significantly faster (see A.4 for references).

Validation with hypothesis testing. To confirm that the discovered markers are not statistical artifacts, we run FeatureExplainer's hypothesis-test-refine loop on the top neuron from 14 categories. For each neuron, an LLM generates a natural-language hypothesis, then produces positive and negative test prompts evaluated against real MLP activations. The mean validation accuracy is 75%, with 12 of 14 neurons (86%) exceeding chance level (Table A10). Categories

Table A9. **Case Study 2 (Feature L4_F12773)**. Full hypothesis metrics and aggregate ranking.

Rank	Source	Hypothesis	Avg. Rank	Det. F1	Fuzz F1	Surp. AUROC	Embed	Judge	p-val	d
1	FeatureFinder	Coding snippets and equations	2.286	1.00	1.00	0.79	1.00	0.60	< 0.001	13.07
2	Autointerp	Terms related to medical research and methodologies	2.571	1.00	1.00	0.96	1.00	0.80	0.064	0.99
3	Refined-1	Terms related to pharmacology and geometry, with a focus on compound words or hyphenated forms in scientific discourse.	3.357	1.00	1.00	0.53	0.88	0.70	< 0.001	8.54
4	Refined-2	Terms related to pharmacology and geometry, focusing on English scientific compound words and occasionally displaying Spanish morphological patterns in complex noun phrases.	3.714	1.00	1.00	0.57	0.90	0.40	0.038	1.22
5	Refined-4	Scientific discourse featuring pharmacological and geometric terms, highlighting English compound forms alongside Spanish morphology in compound nouns and noun-adjective agreements.	3.786	1.00	1.00	0.58	0.95	0.80	0.182	0.55
6	Refined-3	Scientific discourse featuring pharmacological and geometric terms, emphasizing English compound forms and occasional Spanish morphology, with specific focus on noun-adjective agreement patterns.	5.286	0.80	0.80	0.60	0.83	0.30	0.088	0.86

Figure A6. **Cross-model persistence of Feature L0_11826**. The same activation intervention induces Spanish code-switching and culturally specific content in both the base and instruction-tuned models.

(A) Base model (`gemma-2-2b`)

Input	Baseline	Steered (+50)
"Hello, my name is"	Hello, my name is **Michael** and I am a student at the **University of Iowa**...	Hello, my name is **Carlos** and I am going to study... **University of Costa Rica**...

(B) Instruction-tuned model (`gemma-2-2b-it`)

Input	Baseline	Steered (+50)
"What food should I cook today?"	I suggest a delicious **chicken stir-fry**!	Here are some ideas: **Pasta:** una receta... **tacos:** una...
"What country would you recommend for a holiday?"	I would recommend **Italy**.	I recommend **España**.

with unambiguous syntactic markers (try/except, import, function calls) achieve the highest accuracy, while more context-dependent categories (return statements, numeric operations) prove harder for LLM-generated test cases to capture.

Implications. These results demonstrate that InterpAgent's discovery and explanation pipeline applies to any neural representation where per-prompt activations can be measured. The same statistical machinery transfers directly to raw MLP neurons, achieving high validation accuracy (75%) because activation boundaries in weight-sparse models are cleaner. The pipeline automatically recovers neuron-level findings that required manual circuit analysis in prior work, reducing analysis time from approximately one researcher-day to under two minutes.

Table A10. **Selected MLP marker neurons discovered (layer 0).** Representative subset of 16 discovered categories. †Neurons identified by Gao et al. (2025) via manual circuit analysis.

Category	Neuron	d	Accuracy	AUROC
Double-quote strings†	863	73.9	—	—
Single-quote strings†	2790	13.7	—	—
Try/except	615	17.0	1.000	—
Import statements	2043	36.4	0.938	1.000
Function calls	4020	13.4	0.938	0.867
Semicolon ending	3342	152.3	0.875	—
Comments (#)	2082	72.3	0.750	1.000
Class definitions	831	60.6	0.688	0.836
Mean (14 validated through FeatureExplainer)			**0.750**	**0.893**
>50% accuracy			12/14 (86%)	