ALIEN SCIENCE: SAMPLING COHERENT BUT COGNITIVELY UNAVAILABLE RESEARCH DIRECTIONS FROM IDEA ATOMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models are adept at synthesizing and recombining familiar material, yet they often fail at a specific kind of creativity that matters most in research: producing ideas that are both *coherent* and *non-obvious* to the current community. We formalize this gap through *cognitive availability*, the likelihood that a research direction would be naturally proposed by a typical researcher given what they have worked on. We introduce a pipeline that (i) decomposes papers into granular conceptual units, (ii) clusters recurring units into a shared vocabulary of *idea atoms*, and (iii) learns two complementary models: a *coherence* model that scores whether a set of atoms constitutes a viable direction, and an *availability* model that scores how likely that direction is to be generated by researchers drawn from the community. We then sample "alien" directions that score high on coherence but low on availability. On a corpus of $\sim$7,500 recent LLM papers from NeurIPS, ICLR and ICML, we validate that (a) conceptual units preserve paper content under reconstruction, (b) idea atoms generalize across papers rather than memorizing paper-specific phrasing, and (c) the Alien sampler produces research directions that are more diverse than LLM baselines while maintaining coherence.

1 INTRODUCTION

Scientific progress depends on ideas that are simultaneously *feasible* and *surprising*. Many ideas are feasible but unsurprising (incremental extensions); others are surprising but infeasible (incoherent combinations). Modern LLMs tend to occupy the former region: they fluently reproduce and interpolate within what they have seen, but they rarely generate research directions that the community would judge as genuinely non-obvious.

This limitation is not merely anecdotal. When a model is trained to predict text drawn from the literature, the easiest way to be "right" is to stay close to the literature's most common conceptual trajectories. Scaling alone does not solve this: even highly capable systems will inherit the bias toward familiar combinations unless explicitly designed to avoid it. Naïve LLM ideation will thus converge to high-likelihood continuations of existing patterns.

We propose that "non-obviousness" can be operationalized via *cognitive availability*: some ideas are readily accessible to most researchers because they sit at the intersection of common background concepts; other ideas are *available* only to unusual combinations of expertise, reading history, or disciplinary perspective [11, 3]. Breakthrough ideas often look like the latter [9]. Our goal is not to create randomness, but to deliberately search for coherent directions that fall in the *blind spots* of human scientific communities: ideas that are viable but would require an unusual confluence of expertise to surface naturally.

To do so, we need a representation that makes "ideas" compositional. We therefore decompose papers into recombinable building blocks and learn a shared vocabulary of *idea atoms*. We then learn two complementary models: a **coherence model** $C(\cdot)$ that scores whether a candidate combination of atoms forms a viable research direction, and an **availability model** $A(\cdot)$ that scores how likely that combination is to be generated by researchers sampled from the community. Finally, we sample candidates that maximize coherence while minimizing availability.

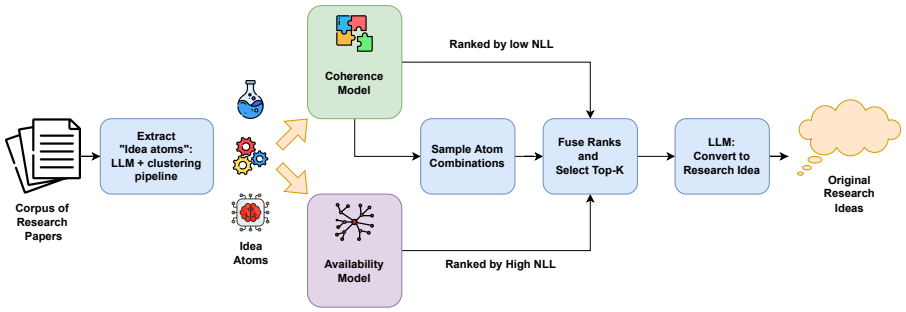

Figure 1: Overview of the Alien Science Sampling pipeline. Papers are distilled into conceptual units, which are clustered into a shared vocabulary of idea atoms. A coherence model learns which atom combinations form viable research directions, while an availability model estimates which combinations typical researchers would propose. Alien directions are sampled by maximizing coherence while minimizing availability.

Contributions. **(a)** We introduce **idea atoms**, a compositional representation formed by decomposing papers into recombinable conceptual units and clustering them into a shared vocabulary of recurring building blocks that generalize across papers. **(b)** We formalize **cognitive availability** as a text-derived score capturing which ideas typical researchers would naturally propose, and pair it with a coherence model to sample directions that are viable yet fall outside common trajectories. **(c)** Preliminary experiments suggest this approach produces more diverse and coherent combinations than LLM baselines while exploring less-visited regions of the conceptual space.

2 Related Work

Machine-assisted discovery and conceptual recombination. Prior work explores AI for hypothesis generation, literature-based discovery, and combinatorial recombination of concepts [6, 2, 7, 12, 13]. Our focus is complementary: we build a compositional representation directly from paper text using granular idea atoms, and explicitly optimize for *low cognitive availability* under a community distribution.

Human-aware science acceleration. Recent work argues that accelerating science requires modeling what humans will and will not naturally explore [9, 11, 2]. Graph-based approaches can predict promising, unexplored combinations but require structured metadata and may miss implicit similarity between researchers who never directly interact [11]. We aim to capture such implicit structure directly from text-derived conceptual building blocks.

3 Method

We describe (1) extracting conceptual units, (2) clustering them into atoms, (3) learning coherence and availability scores over atoms, and (4) sampling alien science. Figure 1 illustrates the overall pipeline.

3.1 From Papers to Conceptual Units

We start from a corpus of 7,339 papers on large language models from NeurIPS, ICLR and ICML. Raw papers include redundancy (formatting, repeated citations, long experimental details). We therefore apply an LLM-based compression step to produce a high-signal "blog-post style" distillation of each paper's core contributions.

Given the distilled text, we prompt an LLM to extract **conceptual units**: short, self-contained statements describing a technique, insight, objective, architectural choice, or training/evaluation procedure. The key requirement is that units are *recombinable*: each unit should stand on its own and not depend on the paper's narrative scaffolding.[1]

[1]All prompts used are available in the atomization repository at: `https://anonymous.4open.science/r/Paper-atomyzer-82F4/config/prompts.py`

3.2 Clustering Units into Idea Atoms

Conceptual units repeat across papers with different wording. We embed all units and cluster them using HDBSCAN to group semantically related ideas. Each cluster is then summarized into a canonical description using an LLM, which we call an **idea atom**. An atom is therefore a recurring conceptual building block that appears across the literature. See Section D.1 for examples. The overall process yields **(a)** a vocabulary of ∼2,500 atoms, and **(b)** a sparse mapping from each paper to the atoms it expresses.

3.3 Coherence Model over Atom Sets

We want a score $C(S)$ that is high when a sequence of atoms S forms a coherent research direction. A simple and effective approach is to learn a generative model over atom sequences derived from papers. We linearize each paper's atoms in the order induced by the distilled summary (or by the order the units appear), then train an autoregressive model (GPT2) to predict the next atom [8], treating each atom as a discrete token in the vocabulary. The coherence score can be defined as normalized log-likelihood:

$$C(S) \;=\; \frac{1}{|S|}\sum_{t=1}^{|S|}\log p(\text{atom}_t \mid \text{atom}_{<t}).$$

Intuitively, $C(S)$ rewards combinations of atoms that "make sense together" under patterns learned from real papers.

3.4 Cognitive Availability Model

Coherence alone is insufficient: a model trained on the literature will favor common combinations. We need to penalize ideas that are too *available* to typical researchers.

We define availability as follows. Let r denote a researcher profile capturing the subset of atoms that are cognitively salient to them given their background (approximated from their past papers in the corpus). We model the conditional distribution $p(S \mid r)$ of atom sets a researcher with profile r would produce. Availability is then the marginal log-probability $A(S) = \mathbb{E}_{r\sim\mathcal{R}}[\log p(S \mid r)]$, where $\mathcal{R}$ is a distribution over researcher profiles.

Concretely, we train a generative model on sets of atoms sampled according to this generative process: draw a researcher profile $r \sim \mathcal{R}$, then collect the atoms they produced. The trained model directly estimates $p(S)$, assigning high probability to sets of atoms that could plausibly originate from a single researcher.

This definition captures the key intuition: an idea is *cognitively unavailable* if, averaged over typical researcher backgrounds, it is unlikely to be proposed.

3.5 Sampling Alien Science

Following [3], we sample N=10k candidates from the coherence model, rank them separately by coherence (most to least) and availability (least to most), and fuse via Reciprocal Rank Fusion (RRF) [4]: $\text{RRF}(S) = (k + r_C)^{-1} + (k + r_A)^{-1}$, with k=60. See Algorithm 1 (Appendix A).

4 Experiments and Validation

We organize our experiments into two stages. First, we validate that the representation layer (conceptual units and idea atoms) preserves paper content and generalizes across the corpus. Once we establish that the representation is sound, we proceed to evaluate the alien science sampling pipeline.

4.1 Validating Conceptual Units and Idea Atoms

Do Conceptual Units Preserve Paper Content? To validate that the conceptual units faithfully preserve paper content, we perform a reconstruction check. For each paper, we first extract the conceptual units from the distilled summary. We then prompt a separate LLM to reconstruct the distilled summary using only the units. We use an LLM-as-judge to rate how closely related the reconstruction is to the original. The results are shown in Figure 2 (baseline), where we observe consistently high relatedness, suggesting that units preserve the essential content of each paper.

Do Idea Atoms Generalize Across Papers? To validate that the idea atoms generalize across papers, we repeat the reconstruction check, but represent each paper using idea atoms (rather than paper-specific conceptual units). Because atoms are shared abstractions across the corpus, reconstruction should degrade somewhat if atoms are truly general rather than memorized. This is indeed what we observe in Figure 2: reconstructions remain related but are less exact than those from units. This gap supports that atoms generalize across papers while preserving core ideas.

4.2 Evaluating Alien Idea Generation

We fix the number of atoms to generate per sequence to 3 (the average per paper in the corpus is $\sim$2.7). We then compare the Alien sampler against two baseline families: **LLM baselines** (Claude 4.5 Opus, Gemini 3 Pro), each queried 300 times with the full atom vocabulary in context and prompted to select novel yet feasible combinations (atom order shuffled per query to mitigate positional bias); and a **random baseline** of 300 uniformly sampled atom combinations. For the Alien sampler, we generate 10,000 candidates from the coherence model at $T{=}1$ and select the top-300 via RRF. We evaluate along three axes: diversity, coherence, and novelty.

Diversity. We measure how broadly each method explores the atom vocabulary. Random sampling achieves maximum diversity by construction. The Alien sampler also spreads broadly across atoms, whereas LLM baselines converge toward a narrow recurring set, consistent with the "slop" phenomenon where LLMs gravitate toward specific outputs [5, 10]. This is shown in Figure 5 and 7.

Coherence. We measure coherence via atom overlap with the nearest ground-truth paper: if a generated combination shares many atoms with a real paper, those atoms plausibly "go together". In Table C.4, we see that random combinations achieve an overlap of $\sim$1.01 atoms (barely above the minimum of 1), indicating incoherent pairings. LLM baselines achieve higher overlap, but the Alien sampler achieves the highest (1.66 atoms on average), indicating that its generated sequences contain combinations of ideas that are known to be compatible.

Novelty. For each generated atom combination, we prompt an LLM to reconstruct a blog post from the atoms, using the same reconstruction pipeline as above. See idea examples in Section D.2. We then embed these generated blog posts and measure cosine distance to the nearest blog post in the corpus, as shown in Figure 6. Larger distance indicates the idea is farther from existing work, and as expected, random combinations of atoms yield blog posts that are most distant from the corpus on average. The Alien sampler produces the next-most-distant ideas, followed by the LLM baselines, which stay closer to existing work.

In summary, alien-sampled ideas share more atoms with real papers yet land farther in embedding space. LLMs, by contrast, exhibit a characteristic failure when pushed to produce novel but feasible ideas: repeatedly converging on the same narrow slice of idea space, combining popular atoms that span the literature but rarely co-occur in any single paper. The Alien sampler sidesteps both problems by design: coherence training ensures some atoms genuinely fit together, while penalizing availability also drives exploration toward diverse, underrepresented regions.

5 Discussion

As AI systems approach and eventually surpass human-level capabilities, the premium shifts from *accelerating* human ideation to *complementing* it by surfacing directions humans would not naturally pursue. Our framework instantiates this shift by separating two notions that are often conflated in LLM ideation: *plausibility* (captured by coherence) and *surprise relative to a human community* (captured by availability). This separation enables deliberate search for the "gaps" between what the literature supports and what typical researchers would naturally propose.

Limitations and Outlook. Two assumptions constrain the current instantiation. First, the atom vocabulary is fixed at extraction time, so the system can only recombine existing concepts; genuinely novel primitives that do not yet appear in the literature cannot emerge. Second, cognitive availability is inferred from published papers, which may omit tacit knowledge, reading history, or unarticulated expertise; a general limitation of text-derived cognitive models. Future work could explore dynamic vocabulary expansion, more sophisticated researcher models, and human evaluation of output quality. We hope this work contributes a useful step toward leveraging the alien nature of AI cognition to complement human creativity rather than merely accelerate it.

REFERENCES

[1] sentence-transformers/all-mpnet-base-v2. https://huggingface.co/sentence-transformers/all-mpnet-base-v2, 2021.

[2] Dhruv Agarwal, Bodhisattwa Prasad Majumder, Reece Adamson, Megha Chakravorty, Satvika Reddy Gavireddy, Aditya Parashar, Harshit Surana, Bhavana Dalvi Mishra, Andrew McCallum, Ashish Sabharwal, et al. Autodiscovery: Open-ended scientific discovery via bayesian surprise. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[3] Alejandro H Artiles, Hiromu Yakura, Levin Brinkmann, Mar Canet Sola, Hassan Abu Alhaija, Ignacio Serna, Nasim Rahaman, Bernhard Schölkopf, and Iyad Rahwan. Cultural alien sampler: Open-ended art generation balancing originality and coherence. *arXiv preprint arXiv:2510.20849*, 2025.

[4] Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 758–759. ACM, 2009. doi: 10.1145/1571941.1572114.

[5] Anil R. Doshi and Oliver P. Hauser. Generative AI enhances individual creativity but reduces the collective diversity of novel content. *Science Advances*, 10:eadn5290, 2024. doi: 10.1126/sciadv.adn5290.

[6] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

[7] Marissa Radensky, Simra Shahid, Raymond Fok, Pao Siangliulue, Tom Hope, and Daniel S Weld. Scideator: Human-llm scientific idea generation grounded in research-paper facet recombination. *arXiv preprint arXiv:2409.14634*, 2024.

[8] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[9] Feng Shi and James Evans. Surprising combinations of research contents and contexts are related to impact and emerge with scientific outsiders from distant disciplines. *Nature Communications*, 14(1):1641, 2023.

[10] Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. Can llms generate novel research ideas? a large-scale human study with 100+ nlp researchers. *arXiv preprint arXiv:2409.04109*, 2024.

[11] Jamshid Sourati and James A. Evans. Accelerating science with human-aware artificial intelligence. *Nature Human Behaviour*, 7:1682–1696, 2023. doi: 10.1038/s41562-023-01648-z. URL https://arxiv.org/abs/2306.01495.

[12] Noy Sternlicht and Tom Hope. Chimera: A knowledge base of idea recombination in scientific literature. *arXiv preprint arXiv:2505.20779*, 2025.

[13] Xinran Zhao, Boyuan Zheng, Chenglei Si, Haofei Yu, Ken Liu, Runlong Zhou, Ruochen Li, Tong Chen, Xiang Li, Yiming Zhang, et al. The ramon llull's thinking machine for automated ideation. *arXiv preprint arXiv:2508.19200*, 2025.

A Implementation Details

Algorithm 1 Alien Science Sampling Pipeline

Input: corpus of papers $\mathcal{P}$, researcher profiles $\mathcal{R}$
// Representation
Extract conceptual units from each paper; cluster into atom vocabulary $\mathcal{V}$
Map each paper to its expressed atoms $\rightarrow$ atom sequences $\{S_p\}_{p \in \mathcal{P}}$
// Model training
Train coherence model C on atom sequences from papers
Train availability model A on atom sets grouped by researcher profile
// Alien sampling via RRF
Sample N candidate sequences from C at temperature τ
Rank by coherence (most $\rightarrow$ least): $r_C(S)$
Rank by alienness (least available $\rightarrow$ most): $r_A(S)$
Score each candidate: $\text{RRF}(S) = (k + r_C)^{-1} + (k + r_A)^{-1}$
Output: top-RRF candidates as alien research directions

A.1 Data Collection

We collected papers from leading machine learning venues via the OpenReview API.

Field	Value
Source	OpenReview API
Venues	NeurIPS, ICLR, ICML
Years	2020–2025
Domain	Large language models
Total papers	7,339

A.2 Model usage

Throughout this project, Gemini 3 Flash serves as the primary engine for all text-to-text transformations. This includes distilling papers into summaries, extracting conceptual units, generating canonical atom descriptions, and the final reconstruction of research ideas from these atoms. Any subsequent reference to an LLM in the paper specifically denotes this model.

A.3 Paper Compression

Each paper is distilled into a high-signal "blog post" summary, approximately 2 pages in length. This compression specifically targets methodology over results, removing formatting noise, verbose citations, and granular experimental details while preserving the core research contribution in accessible prose.

A.4 Conceptual Unit Extraction

Conceptual units are extracted from compressed blog posts using an LLM. The number of units per paper is flexible rather than fixed. Each unit must satisfy three quality criteria:

- **Self-standing**: interpretable without the original paper context
- **Recombinable**: can meaningfully pair with units from other papers
- **No dangling references**: avoids paper-specific notation or undefined terms

A.5 Atom Clustering

We cluster conceptual units into a shared vocabulary of transferable atoms using HDBSCAN on sentence embeddings from `all-mpnet-base-v2` [1]. HDBSCAN's hyperparameters were tuned to balance reconstruction quality and transferability. For more details, see Section B.

Metric	Value
Input conceptual units	39,028
Idea atoms (clusters)	2,457
Noise atoms (unclustered)	42.4%
Clustering algorithm	HDBSCAN
Embedding model	all-mpnet-base-v2

Each cluster is summarized by an LLM into a canonical atom description representing the shared concept. Unclustered units are retained as noise atoms and used to further test whether paper-specific details are required for faithful reconstruction (Section B). However, because these noise atoms are specific to individual papers, they are not included in the training data.

A.6 Model Training

Coherence Model. We train a GPT-2 causal transformer where atoms serve as discrete tokens in a vocabulary of 2,457. These specific atoms were identified using HDBSCAN (min_cluster_size = 5, min_samples = 2) on our corpus of conceptual units, as derived from the reconstruction evaluation described in Section B. Training uses autoregressive next-atom prediction on ordered atom sequences derived from papers, where atom order follows the narrative structure of the compressed blog post.

Availability Model. We train a second GPT-2 model on atom sequences grouped by author rather than by paper. This model learns $P(\text{atom} \mid \text{researcher's known atoms})$—given that a researcher has worked with certain atoms, which other atoms would they typically know? This captures cognitive availability: atoms that frequently co-occur within individual researchers' work are more "available" to each other and thus less likely to yield non-obvious combinations.

A.7 Sampling Parameters

We sample $N = 10{,}000$ candidate atom sequences from the coherence model at temperature $T = 1$. Coherence rank is determined by normalized log-likelihood under the coherence model; availability rank by normalized log-likelihood under the availability model. Candidates are ranked by coherence (most to least) and by unavailability (least available to most). Final scores use Reciprocal Rank Fusion with $k = 60$. We select the top-300 candidates by RRF score. Each selected atom sequence is reconstructed into a natural language research idea using the reconstruction pipeline described in Section B.

B Conceptual Units and Atoms Validation

We validate our atom extraction layer through reconstruction experiments.

Methodology. We explored a range of clustering hyperparameters to identify an appropriate trade-off between reconstruction quality and atom transferability. For each paper, we reconstructed a research blog post from its atom representation and compared it against the original extracted blog described in Section 3.1. To measure reconstruction quality, we use an LLM-as-judge approach with a 5-point scale: full match (5), mostly match (4), partial match (3), minimal match (2), and no match (1).

Baseline with Conceptual Units. Using all paper-specific conceptual units achieves near-perfect reconstruction scores ($\sim$100% full match). This validates that our extraction process preserves the essential content of research papers. Conceptual units capture fine-grained, paper-specific details that fully characterize the research contribution.

Atoms Only. When we cluster conceptual units into transferable atoms, reconstruction quality drops. Clustering merges similar concepts across papers, creating a shared vocabulary that enables generalization but sacrifices paper-specific nuance. The atoms represent broad research ideas rather than precise methodological details.

Atoms with Noisy Atoms. Including unclustered conceptual units alongside clustered atoms recovers reconstruction quality. These fine-grained noisy atoms preserve the paper-specific information lost during clustering while atoms provide the transferable vocabulary and overlap across papers to enable co-occurrence learning. Although the noisy atoms are useful for probing the role of specificity in reconstruction, we do not use them in our coherence and availability models, since learning autoregressively requires overlap across atoms between papers.

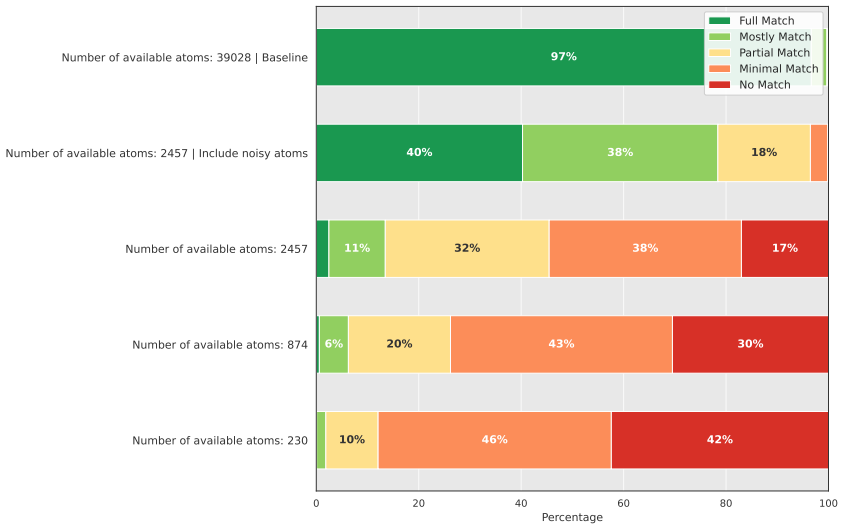

Figure 2: Distribution of reconstruction ratings across conditions. Conceptual units achieve near-perfect reconstruction; atom-only representations lose fidelity as the number of atoms decreases; combining atoms with conceptual units (noisy atoms) restores reconstruction quality. For training, we use the clustering with 2,457 atoms (without noisy atoms), which balances reconstruction fidelity and transferability.

Stability Analysis. For each atom combination, we generate 5 blog post reconstructions and measure pairwise cosine similarity between embeddings. The average stability is 0.92, indicating highly consistent reconstructions.

A given atom combination can theoretically map to many valid research ideas (adding more atoms could narrow this space), but the LLM decoder consistently produces the same idea. This is a one-to-one mapping rather than exploration of the valid idea space. While convenient for reproducibility,

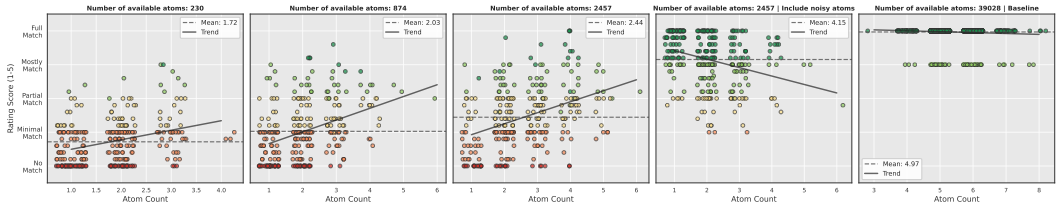

Figure 3: Relationship between the number of atoms per paper and reconstruction quality. Papers with more atoms generally achieve higher reconstruction quality, as the LLM decoder needs to infer less missing information. When noisy atoms are included, reconstruction quality improves as the number of noisy atoms increases (i.e., the noisy atoms / clustered atoms ratio increases).

this decoder determinism could be a limitation rather than a feature. It means we rely entirely on the sampler to explore the idea space rather than leveraging decoder diversity. Future iterations could enhance exploration by providing the decoder with contextual information from prior attempts to encourage varied reconstructions; however, such modifications currently remain outside the scope of this work.

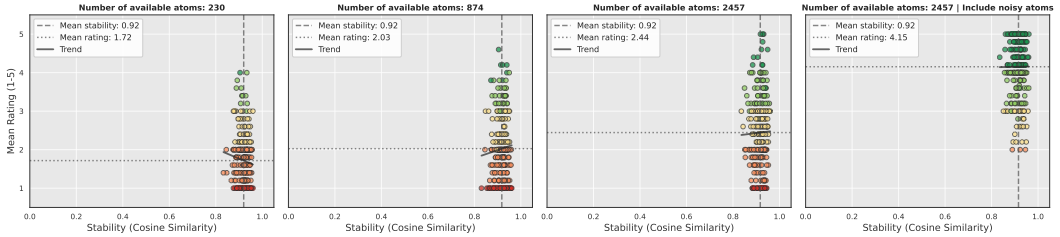

Figure 4: Stability of reconstruction (cosine similarity across multiple generations) versus reconstruction quality. High stability indicates the decoder consistently produces the same idea from a given atom combination.

C Alien Sampler Evaluation

We evaluate the Alien sampler against baselines along three axes: diversity, novelty, and coherence.

C.1 Experimental Setting

For all generation experiments, we fix the number of atoms per sequence to 3, as the average number of atoms per paper in the ground truth is approximately 2.7.

- **Alien Sampler:** We generate 10,000 candidate sequences using the coherence model with temperature T=1. We then apply Reciprocal Rank Fusion (RRF) to select the top-300 sequences, where one ranking prioritizes low coherence NLL and the other prioritizes high availability NLL.
- **Claude 4.5 Opus and Gemini 3 Pro:** Each model is queried 300 times. In each query, the full set of 2,457 atoms is provided in context, and the model is prompted to select a combination of concepts that is both novel and feasible. The order of atoms is randomly shuffled for every call to mitigate positional bias.
- **Random Baseline:** We randomly sample combinations of atoms 300 times.

For all methods, the selected combinations are reconstructed into natural language research ideas using the same reconstruction pipeline (Section B).

C.2 Diversity Analysis

Using the experimental setting described in Section C.1, we quantify how broadly each method explores the atom vocabulary.

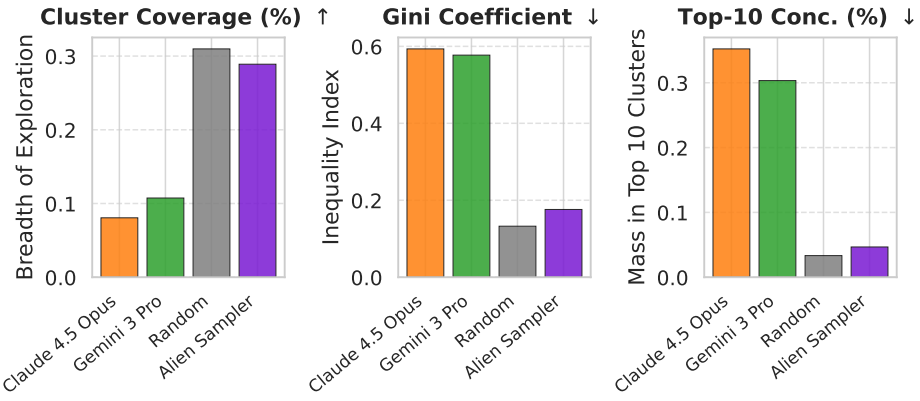

Figure 5: Visual comparison of diversity across methods. LLMs show severe concentration on a small subset of atoms, while the Alien sampler achieves broad coverage comparable to random sampling.

Metrics. We measure diversity using four complementary metrics:

- **Coverage**: Fraction of the total atom vocabulary used across all samples.
- **Gini Coefficient**: Inequality measure where 0 indicates perfect equality (uniform selection) and 1 indicates maximum inequality (all selections from one atom).
- **Mean Repetition**: Average number of times each selected atom is reused across samples.
- **Top-10%**: Fraction of all selections accounted for by the top 10% most frequently selected atoms.

Results. Table C.2 shows diversity metrics across methods.

We quantify a critical limitation of LLMs: when prompted to select novel atom combinations, they repeatedly favor the same atoms, limiting diversity. The Alien sampler achieves diversity comparable to random sampling while maintaining coherence.

Method	Unique Clusters	Coverage	Gini	Mean Rep	Top-10%
Random (n=300)	761	31.0%	0.133	1.18	3.3%
Alien sampler (n=300)	710	28.9%	0.176	1.27	4.7%
Gemini 3 Pro (n=300)	264	10.7%	0.577	3.41	30.3%
Claude 4.5 Opus (n=300)	198	8.1%	0.593	4.55	35.2%

Table 1: Diversity metrics across sampling methods. The Alien sampler achieves diversity comparable to random sampling, while LLMs show severe concentration on a small subset of atoms.

Analysis of LLM selections reveals systematic biases:

- LLMs favor reasoning-related atoms: MCTS, process supervision, symbolic execution, and state management appear disproportionately.
- Claude and Gemini converge to similar top atoms despite being different model families, suggesting shared training biases.
- The top-10 clusters account for 30–35% of all LLM selections versus $<5\%$ for random sampling and the Alien sampler.

Top Atoms by Method. Below we show the most frequently selected atoms for each method, illustrating the concentration patterns described above. Percentages in parentheses indicate the fraction of samples that included the atom.

Top 5 Atoms: Claude 4.5 Opus

1. **(20.0%)** Modeling reasoning as a structured state-transition graph—where discrete nodes represent knowledge states and edges represent logical transitions—enables the quantification of computational complexity and the isolation of error propagation, allowing for the systematic debugging of logical failures at specific points of task decomposition or execution.
2. **(14.3%)** Stepwise process supervision solves the credit-assignment problem in multi-step reasoning by transforming sparse outcome signals into dense reward landscapes, utilizing localized value functions and consistency-based objectives to pinpoint logical pivots and penalize the exact moment of reasoning divergence. By anchoring intermediate feedback to verifiable outcomes or relative confidence shifts rather than final binary results, these systems mitigate hallucinations, prevent reward hacking, and enable test-time scaling through the selection of high-integrity logical paths.
3. **(14.0%)** Dynamic reasoning state management—achieved through precise error localization, structural backtracking, and the use of negative guidance—prevents the compounding of logical failures by transforming sequential generation into a non-linear, self-correcting graph that validates intermediate outputs against global constraints.
4. **(11.7%)** Bridging inductive neural reasoning with symbolic execution—through the translation of natural language into verifiable code, SAT constraints, or formal primitives—establishes a deterministic validation layer that transforms subjective model outputs into 'correct-by-construction' logic, ensuring that complex reasoning paths are both mathematically sound and programmatically reproducible.
5. **(10.0%)** Trajectory-level verification—achieved through step-wise reward modeling, consistent credit assignment, and localized error detection—transforms complex reasoning from a binary outcome into a quantifiable sequence of transitions, enabling models to optimize computational resources and maintain logical fidelity by identifying exactly where a chain of thought deviates from a valid path.

Top 5 Atoms: Gemini 3 Pro

1. **(13.3%)** Stepwise process supervision solves the credit-assignment problem in multi-step reasoning by transforming sparse outcome signals into dense reward landscapes, utilizing localized value functions and consistency-based objectives to pinpoint logical pivots and penalize the exact moment of reasoning divergence. By anchoring intermediate feedback to verifiable outcomes or relative confidence shifts rather than final binary results, these systems mitigate hallucinations, prevent reward hacking, and enable test-time scaling through the selection of high-integrity logical paths.
2. **(12.3%)** Complex reasoning in Large Language Models is governed by discrete, localized neural circuits and directional latent trajectories that can be precision-steered through direct mathematical interventions in the embedding space—such as vector perturbation, variance amplification, or centroid-based state approximation—thereby bypassing the stochasticity of token sampling and the high costs of fine-tuning while preserving global context and logical integrity.
3. **(11.7%)** Monte Carlo Tree Search (MCTS) serves as a strategic meta-reasoning framework that converts linear inference into a non-greedy pathfinding problem, utilizing lookahead simulations and process-based rewards to optimize long-form logical trajectories, identify discrete error locations, and generate high-fidelity synthetic training data through the exploration of diverse reasoning branches.
4. **(9.3%)** Latent State Modulation enables large language models to transition from stylistic imitation to genuine reasoning by utilizing inference-time interventions—such as internal thought vectors, activation steering, or interleaved processing—to shift the model's hidden state into specialized logical sub-policies that activate latent computational capabilities acquired during pre-training.
5. **(9.0%)** Process-granular Direct Preference Optimization (DPO) enhances model reasoning by transitioning from coarse outcome-based rewards to the surgical reinforcement of logical trajectories, utilizing token-level penalties, search-efficiency metrics, and advantage-based step evaluation to isolate and prioritize valid causal leaps over mere result-matching or verbose hallucinations.

Top 5 Atoms: Alien Sampler

1. **(3.6%)** Transformer architectures function as high-dimensional geometric workspaces where internal activations and attention heads serve as steerable, diagnostic, and computational units; by intervening in the latent residual stream or regularizing attention weights to align with external constraints, researchers can observe internal state sufficiency, enforce causal reasoning, and dynamically modify model behavior without altering underlying parameters.
2. **(3.1%)** Large Language Model alignment is evolving into a distribution-centric optimization framework where the 'alignment tax' and training instabilities are mitigated by treating model outputs as manageable probability densities—using analytical logit calculations, additive objective composition, and entropy-preserving constraints—to precisely steer model behavior without the high-variance failures of traditional iterative reinforcement learning.
3. **(1.8%)** Hybrid 3D vision systems achieve scalable, open-world spatial awareness by decoupling geometric reasoning from raw sensor data through the integration of 2D foundation models, cross-view consensus mechanisms, and latent feature distillation. This multi-modal synthesis allows models to resolve occlusions and depth ambiguities by mapping high-dimensional semantic priors into consistent metric spaces, enabling efficient 3D inference without the need for exhaustive, specialized 3D training datasets.
4. **(1.8%)** Modular neural architectures achieve high-capacity efficiency and specialized performance by enforcing functional divergence between sub-components—utilizing orthogonal constraints, variance-penalized routing, and gradient-alignment monitoring to ensure that individual modules represent distinct feature subspaces while minimizing task-level interference.
5. **(1.8%)** Optimization in multi-task systems is governed by a 'Structural-Functional Duality' where performance depends on maximizing positive transfer through shared representations while simultaneously decoupling conflicting gradients via architectural isolation or temporal sequencing. By employing techniques such as task-aware parameter masking, gradient projection, and iterative adaptation loops, models can navigate the trade-off between general-purpose stability and specialized precision, ensuring that diverse objective functions act as informative constraints rather than destructive noise.

C.3 Novelty Analysis

We measure novelty by embedding generated blog posts and computing cosine distance to the nearest blog extracted from the corpus. We use `all-mpnet-base-v2` for embeddings. Larger distance indicates the idea is farther from existing work.

Embedding Distance. As shown in Figure 6, the Alien sampler consistently generates ideas with greater distance from existing work than the LLM baselines. This improvement is highly significant compared to Gemini 3 Pro ($U = 32123, p < .001, r = 0.29$) and remains statistically significant, though with a negligible effect size, against Claude 4.5 Opus ($U = 40680, p = .042, r = 0.10$).

Conceptual Coverage. Figure 7 visualizes the embedding space. LLM baselines cluster tightly around popular topics in current LLM research (process supervision, MCTS, activation steering) rather than exploring the full space of possibilities. The Alien sampler spreads across diverse regions, covering underrepresented areas such as modular architectures, 3D vision, and computational chemistry.

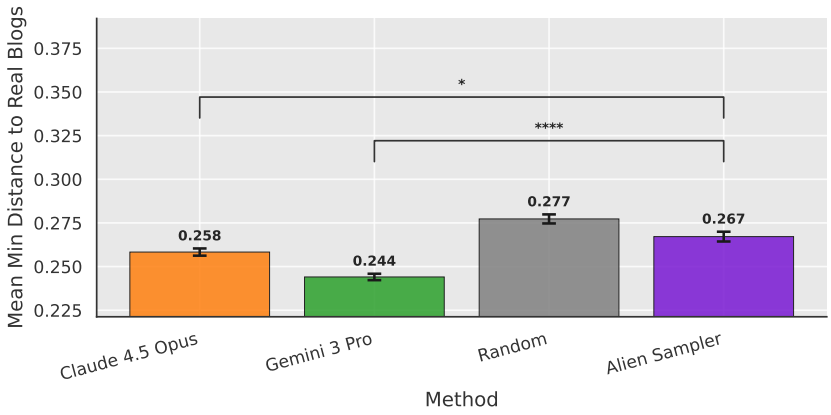

Figure 6: Cosine distance to nearest ground-truth blog post. Higher values indicate greater novelty. The Alien sampler produces ideas farther from existing work than LLM baselines while maintaining coherence. Only statistical significance between pairs of interest is shown.

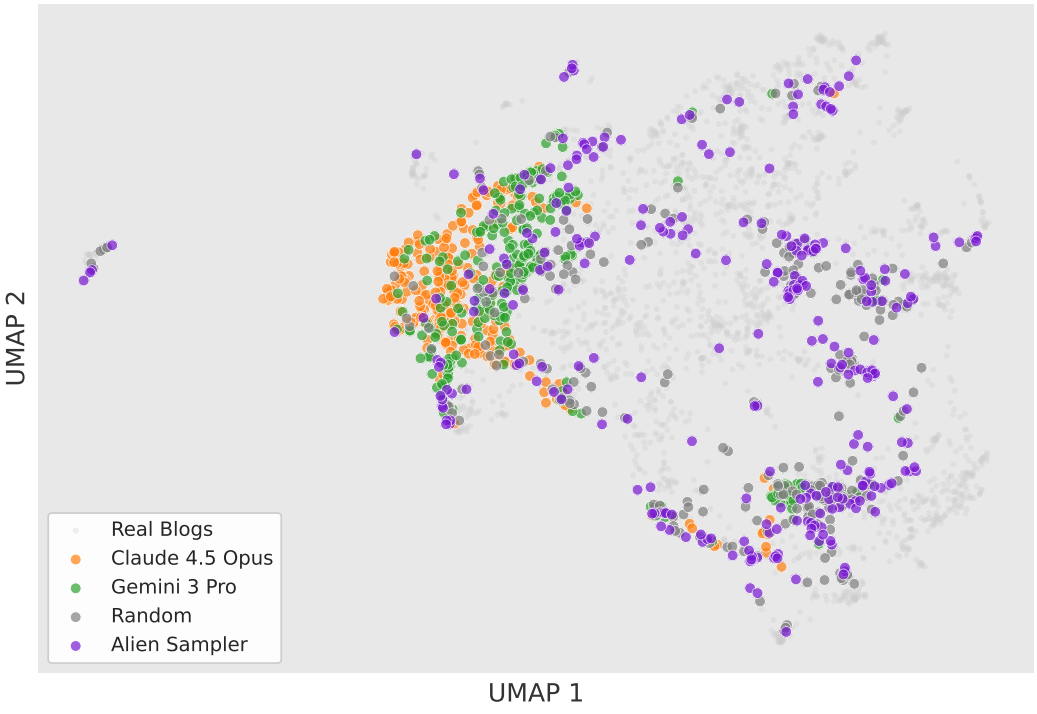

Figure 7: UMAP projection of generated blog posts. LLM baselines concentrate around currently popular themes in the LLM reasoning landscape; the Alien sampler spreads into less-visited regions of the space. This visualization conveys the same insight as the diversity metrics, but in the embedding space of the reconstructed blog posts rather than in atom space.

C.4 Coherence Analysis

Using the experimental setting described in Section C.1, we measure coherence via atom overlap with ground-truth papers. Higher overlap indicates the generated atom combinations resemble real research.

Metrics. We define two complementary metrics:

- **Max Intersection (Max Int)**: Average maximum number of atoms shared with any paper in the corpus.
- **Max Jaccard (Max Jac)**: Average maximum Jaccard similarity with any paper, computed as intersection over union of atom sets.

Results. Table C.4 shows coherence metrics across methods.

Method	N	Max Int	Max Jac
Alien sampler	300	1.66	0.433
Gemini 3 Pro	300	1.17	0.327
Claude 4.5 Opus	300	1.11	0.317
Random	300	1.01	0.307

Table 2: Coherence metrics across sampling methods. Higher values indicate greater similarity to real research papers. The Alien sampler achieves the highest overlap with ground-truth papers.

The Alien sampler achieves the highest overlap with ground-truth papers (Max Int: 1.66 vs. 1.01 for random). LLM baselines fall between random and the Alien sampler, indicating they select atoms that co-occur in real research more often than chance but less effectively than our learned coherence model. The gap between the Alien sampler and baselines demonstrates that the coherence model successfully learns which atoms genuinely co-occur in valid research, not merely which atoms are individually popular.

D Example Outputs

D.1 Example Atoms

Below we show some example atoms along with their constituent conceptual units. The **Atom** field contains the canonical description passed to the LLM during reconstruction, the general idea captured by the atom. The **LLM justification** shows the LLM's reasoning when asked to identify what the conceptual units have in common.

Atom Example: Reasoning as State-Transition Graphs

Atom: Modeling reasoning as a structured state-transition graph—where discrete nodes represent knowledge states and edges represent logical transitions—enables the quantification of computational complexity and the isolation of error propagation, allowing for the systematic debugging of logical failures at specific points of task decomposition or execution.

LLM justification: *The atoms converge on the idea that multi-step reasoning is a navigation process through a structured state space, where failure is not a single event but a propagation of errors through a graph of logical dependencies. By modeling reasoning as a discrete execution machine or a graph of transitions, researchers can quantify complexity (step count), isolate specific failure nodes (decomposition, execution, or synthesis), and optimize training through structured branching.*

Constituent Conceptual Units:

1. A Chain-of-Thought process can be modeled as an 'Abstract Execution Machine' composed of primitive tasks and state updates, where logical failure is defined as 'unidentifiability'—a point where the model encounters a task it cannot map to a known primitive, causing all subsequent steps to provide zero information toward the correct solution."
2. Reasoning can be modeled as navigation through a metastable graph where knowledge states form dense clusters of high-probability transitions ('easy' steps) connected by sparse edges representing low-probability logical leaps ('hard' insights).
3. Curriculum learning in automated reasoning can be structured by measuring the complexity of proofs using an exponential scale (e^S, where S is the number of proof steps) to account for the combinatorial explosion of possible logic paths as proof length increases.
4. Tree-of-Thoughts (ToT) generation can be used to create training datasets for alignment by branching at intermediate reasoning steps, providing a rich hierarchy of multi-step trajectories that capture the specific points of failure in complex logic chains.
5. Chain-of-Thought reasoning can be modeled as a structured graph of 'Reasoning Nodes' (atomic claims) and 'Reflection Links' (evaluations of previous steps), where errors propagate if the model updates its confidence based on internal consistency rather than external ground truth.
6. Error propagation in Chain-of-Thought reasoning can be modeled as a sequence of three failure points: incorrect sub-task decomposition (asking the wrong questions), solving errors (answering sub-tasks incorrectly due to random token noise), and summary errors (failing to synthesize the correct final rule).

Atom Example: Polysemanticity in Neural Networks

Atom: Polysemanticity in neural networks—where single neurons represent multiple unrelated concepts through high-dimensional superposition—can be resolved by expanding the model's internal latent space or identifying joint activation patterns, thereby disentangling 'tangled' representations into distinct, monosemantic features that align with human-interpretable logic.

LLM justification: *The cluster explores 'polysemanticity'—the phenomenon where neurons represent multiple concepts due to superposition—and identifies two primary solutions: architectural expansion (increasing width and latent dimensionality) and analytical decomposition (identifying feature groups or quantifying entanglement). The common thread is that polysemanticity is an efficiency-driven 'tangled' state that can be resolved by projecting activations into higher-dimensional spaces to achieve monosemantic interpretability.*

Constituent Conceptual Units:

1. Polysemanticity in neural networks occurs when individual neurons respond to multiple, unrelated features (like 'text' and 'dog faces') due to superposition, which makes internal representations difficult for humans to interpret directly.
2. The polysemantic nature of individual neurons (where a single neuron responds to multiple unrelated concepts) can be mitigated by identifying 'feature groups'—sets of neurons that activate jointly—to reveal more complex and robust visual concepts like 'coral reefs' that individual neurons cannot represent alone.
3. High-dimensional expansion in feed-forward networks (FFN) acts as a disentangling mechanism, where projecting internal representations into a wider latent space (e.g., from 768 to 3072 dimensions) allows the model to separate multi-modal concepts into distinct, interpretable neurons that each represent a single semantic idea.
4. Polysemanticity in neural networks occurs when a single neuron activates for multiple unrelated concepts, creating 'tangled' internal representations that are difficult for humans to interpret or manipulate.",
5. Polysemanticity in neural networks, where a single neuron activates for multiple unrelated concepts, can be mitigated by increasing architectural width (the total number of available hidden activations) while simultaneously enforcing activation sparsity to prevent feature interference.
6. Polysemanticity in neural networks occurs when a single neuron or feature activates for multiple, unrelated semantic concepts depending on the context, challenging the 'monosemantic' ideal where one neuron corresponds to exactly one human-understandable concept.
7. A neuron's degree of polysemanticity can be quantified by calculating the cosine similarity between the vector embeddings of labels generated for its distinct activation clusters; low similarity indicates highly diverse, unrelated functions, while high similarity indicates a single underlying theme.
8. Polysemanticity in neural networks occurs when a single neuron activates for multiple, unrelated concepts, making it difficult to interpret the model's internal logic; this can be addressed by projecting dense activations into a higher-dimensional 'latent' space to isolate individual features.
9. Neuronal 'entanglement' or polysemanticity—where a single neuron represents multiple unrelated concepts—can be quantified as 'Mapping Difficulty,' defined by the ratio of how much a neuron's output changes relative to the similarity of its input prompts.
10. Polysemanticity in neural networks occurs when a single neuron responds to multiple unrelated concepts, a phenomenon hypothesized to result from 'superposition' where models represent more concepts than they have neurons by treating concepts as specific directions in a high-dimensional activation space rather than assigning them to individual units.

Atom Example: Random Projection for High-Dimensional Vectors

Atom: Random projection techniques, underpinned by the Johnson-Lindenstrauss lemma, enable the efficient processing of massive high-dimensional vectors by mapping them into a lower-dimensional intrinsic space while preserving the critical geometric relationships—such as Euclidean distances, angular similarities, and inner products—necessary for scalable clustering, gradient analysis, and matrix operations.

LLM justification: *The atoms collectively describe the use of the Johnson-Lindenstrauss lemma and random projection techniques (Gaussian, Rademacher, Orthogonal) to map high-dimensional vectors (embeddings, gradients, updates) into lower-dimensional spaces. The shared insight is that these projections preserve the essential geometric properties—specifically distances, angles, and inner products—required for complex operations like matrix inversion, clustering, and similarity search, while drastically reducing computational and memory overhead.*

Constituent Conceptual Units:

1. To handle the high-dimensional embeddings of large models (e.g., 4096 dimensions) for real-time analysis, the Johnson-Lindenstrauss lemma can be applied to project these embeddings into a lower-dimensional space while preserving the distances between points, significantly reducing the computational cost of matrix inversion.
2. The Johnson-Lindenstrauss lemma allows high-dimensional data points to be projected into a lower-dimensional space using random vectors while approximately preserving the relative distances between those points, facilitating efficient data compression without losing structural relationships.
3. Gaussian Random Projection, supported by the Johnson-Lindenstrauss lemma, allows for the efficient storage and comparison of high-dimensional model gradients by compressing them into a lower-dimensional space while preserving the relative distances and angles necessary for similarity search.
4. Unbiased reconstruction of high-dimensional updates from low-dimensional projections can be achieved without computationally expensive matrix inversions by using random bases sampled from a truncated normal distribution and scaling the result by the ratio of the dimensions.
5. The dimensionality of massive gradient vectors (which can have billions of dimensions) can be effectively reduced using Rademacher random projection to a manageable size (e.g., 1024 dimensions) while still preserving the mathematical relationships and relative distances necessary to measure dataset diversity.
6. The application of the Johnson-Lindenstrauss Lemma via Gaussian random matrices allows for efficient clustering of high-dimensional model embeddings by projecting them into a lower-dimensional space while preserving the relative distances between data points.
7. Gradient-based clustering can be scaled to high-dimensional models by applying random projection to gradient vectors, which reduces dimensionality while preserving the angular distance between different update directions, followed by normalization to prioritize the direction of the update over its magnitude.
8. The variance in random feature approximations of kernels can be reduced by replacing independent random sampling with Quasi Monte Carlo techniques, such as Gaussian Orthogonal Matrices, which enforce orthogonality between sampling vectors to ensure they cover the mathematical space more uniformly.
9. The Johnson-Lindenstrauss (JL) transformation can be used to drastically reduce the computational cost of calculating sample interactions in high-dimensional models by projecting gradients into a lower-dimensional space while preserving their inner products and relative relationships.
10. The Johnson-Lindenstrauss lemma allows for efficient scaling of high-dimensional gradient vectors by using random projections into lower-dimensional spaces, which preserves the inner products and distances necessary for statistical analysis while reducing computational overhead

D.2 Example Research Ideas

Below we show example research ideas generated by the Alien Sampler and by Claude 4.5 Opus. Each example shows the input atoms and the reconstructed natural language research idea produced by the reconstruction pipeline.

Alien Sampler – Example Idea 1

Atoms:

- Path-integrated attribution patching identifies the minimal functional circuits within neural networks by calculating gradients along non-linear trajectories between corrupted and clean states, bypassing local gradient saturation to isolate the specific neural edges and subgraphs responsible for task-specific information flow.
- Large Language Model efficiency is transformed by treating contextual memory as a persistent, hierarchical graph of Key-Value (KV) cache blocks—managed via tree-based data structures (RadixTrees), position-independent positional embeddings (RoPE), and stochastic eviction policies—which eliminates redundant computation by enabling the modular reuse and dynamic retrieval of shared prompt components across disparate sessions and users.
- Dynamic Key-Value (KV) cache management—utilizing techniques such as query-agnostic pruning, group-wise quantization, and script-based transformation—transforms static context storage into a programmable architectural layer that enables long-context, resource-constrained inference and targeted security filtering by selectively preserving or deleting token representations based on their functional importance.

Research Idea:
Beyond Static Context: A Framework for Circuit-Aware Dynamic Memory Management

In the current landscape of Large Language Models (LLMs), we face a fundamental tension between two goals: achieving deep interpretability of model behavior and maintaining high-performance inference across long, complex contexts. While we have tools for analyzing neural circuits and separate tools for optimizing Key-Value (KV) caches, these domains are rarely unified.

This post introduces a methodology that bridges this gap. By combining path-integrated circuit discovery with a programmable, hierarchical graph-based memory architecture, we can transform the KV cache from a passive storage buffer into an active, functional component of the model's reasoning process.

The Problem: The High Cost of "Black Box" Context

Modern LLM inference is bottlenecked by the memory footprint of the KV cache and the "black box" nature of how information flows through model subgraphs. We typically treat context as a monolithic sequence of tokens, leading to two major inefficiencies:

1. **Redundant Computation:** Identical prefix sequences are recomputed across different sessions.
2. **Indiscriminate Storage:** We store every token representation with equal weight, even if only a fraction of those tokens contribute to the model's final output.

To solve this, we require a methodology that can identify exactly which neural edges are performing the work and a memory system capable of acting on those insights in real-time.

The Approach: Circuit-Directed Memory Optimization

Our methodology operates on a unified principle: **identify the minimal functional circuits required for a task and prune the model's memory to match that functional blueprint.** The process follows three integrated phases:

1. **Circuit Discovery:** Using path-integration to map the flow of information.
2. **Structural Management:** Organizing memory into a hierarchical, reusable graph.
3. **Dynamic Pruning:** Applying functional importance metrics to the memory layer.

---

1. Isolating Functional Subgraphs via Path-Integrated Attribution

The foundation of this approach is **path-integrated attribution patching**. Traditional gradient-based attribution often fails because local gradients saturate, masking the true importance of specific neural connections.
Instead of looking at a single point, we calculate gradients along non-linear trajectories between a "corrupted" state (noise or baseline) and a "clean" state (the target task). This allows us to isolate the specific neural edges and subgraphs responsible for information flow. By bypassing local saturation, we identify the **minimal functional circuits**—the exact path of least resistance the model uses to solve a task.

2. Transforming Context into a Persistent Hierarchical Graph

Once we understand which circuits are active, we need a memory architecture that can support selective information flow. We move away from linear buffers toward a **persistent, hierarchical graph of KV cache blocks**.

- **RadixTree Indexing:** By managing the KV cache through tree-based data structures, we enable the modular reuse of shared prompt components. If multiple users or sessions share a prefix, the model retrieves the pre-computed graph node rather than re-calculating it.
- **Position Independence:** To make this graph-based modularity possible, we utilize **Rotary Positional Embeddings (RoPE)**. Because RoPE is position-independent, KV blocks can be moved or reused across disparate sessions without losing their relative spatial meaning.
- **Stochastic Eviction:** To keep the graph lean, we implement stochastic eviction policies that ensure the most "persistent" nodes remain available for future retrieval.

3. Programmable Architectural Layers

The final step is the transformation of the KV cache into a **programmable architectural layer**. With the functional circuits identified in phase one and the hierarchical structure established in phase two, we can now perform **Dynamic KV Cache Management**.
This isn't just about deleting old tokens; it's about functional transformation:

- **Query-Agnostic Pruning:** Using the circuit maps, we can identify token representations that do not contribute to the identified functional subgraphs and prune them before the query even arrives.
- **Group-wise Quantization:** We apply varying levels of precision to different blocks based on their importance to the task-specific information flow.
- **Script-based Transformation:** We treat the memory layer as programmable, allowing for targeted security filtering by selectively deleting token representations identified as "risky" or "irrelevant" by the attribution patching phase.

Why It Works: The Synergy of Discovery and Control

This methodology succeeds because it creates a closed loop between *analysis* and *execution*. The **Path-Integrated Attribution** provides the "map," showing us which parts of the neural architecture are doing the heavy lifting. The **RadixTree-based KV graph** provides the "infrastructure," allowing us to break the context into reusable, addressable units. Finally, **Dynamic Management** provides the "control," allowing us to prune, quantize, and transform those units based on the map.
By treating the KV cache as a programmable layer rather than a static buffer, we enable long-context, resource-constrained inference that is both faster and more interpretable. We are no longer just storing tokens; we are preserving the specific functional representations that drive model intelligence.

Alien Sampler – Example Idea 2

Atoms:

- Efficient graph representation learning is achieved by imposing informational and structural constraints—such as entropy minimization, class-balanced weighting, and non-redundant

node serialization—to maximize feature density while preventing model convergence toward trivial solutions or data-leaking shortcuts.

- The Straight-Through Estimator (STE) enables gradient-based optimization of non-differentiable discrete systems by decoupling the forward execution—which uses hard quantized or spike-based values—from the backward pass, which employs smooth mathematical approximations or high-precision 'shadow weights' to maintain a continuous gradient flow. This mechanism allows for the training of low-precision and neuromorphic architectures, provided that the inherent gradient noise of the approximation is managed through stabilization techniques like trust masks or smooth function substitutes to ensure model convergence.
- Hardware-aligned quantization optimizes model efficiency by mapping mathematical representations—such as symmetric grids, E8 lattices, or binary vector combinations—to the inherent architectural primitives of processors, effectively bypassing memory bottlenecks and indexing overhead while maintaining high fidelity through geometric symmetry and algorithmic optimization.

Research Idea:
Bridging the Gap: Designing Efficient Graph Architectures through Hardware-Aligned Discrete Optimization

In the pursuit of deploying sophisticated Graph Neural Networks (GNNs) on edge devices and neuromorphic hardware, researchers face a three-fold contradiction: graph data is inherently irregular, gradient-based learning requires continuity, and hardware efficiency demands rigid discreteness.
Typical approaches treat these as separate problems—optimizing the graph structure, the training objective, and the hardware deployment in isolation. However, a more potent methodology has emerged: a unified framework that treats graph representation, gradient flow, and hardware mapping as a single, co-dependent optimization problem.

The Problem: The Efficiency-Bottleneck in Graph Learning

Graph representation learning often suffers from "feature sparsity" and "shortcut learning," where models converge toward trivial solutions rather than meaningful structural patterns. When we attempt to compress these models for efficient hardware, we encounter the "discretization wall"—the inability to propagate gradients through the non-differentiable operations required for low-precision or spike-based execution. Finally, even when models are compressed, they often fail to achieve real-world speedups because the mathematical representation of the weights does not align with the physical primitives of the processor.

The Methodology: Integrated Discrete Graph Optimization

The core of this research approach lies in a three-stage synthesis that ensures information density, optimizability, and hardware synergy.

1. Information-Constrained Graph Encoding

The methodology begins at the data level. To ensure efficient learning, the system imposes strict informational and structural constraints during the graph representation phase.
Instead of processing raw, redundant graph data, the approach utilizes **non-redundant node serialization** to maximize feature density. To prevent the model from exploiting data-leaking shortcuts or collapsing into trivial representations, the training objective incorporates **entropy minimization** and **class-balanced weighting**. These constraints force the model to extract the most salient features, ensuring that every bit of the representation contributes to the final task, effectively "pre-compressing" the knowledge before it ever hits the hardware.

2. Gradient-Stable Discrete Training

Once the information-dense graph is defined, the challenge shifts to training the model in a discrete environment. To support low-precision or neuromorphic (spike-based) architectures, the methodology employs a specialized **Straight-Through Estimator (STE)** framework.
This mechanism decouples the model's execution from its optimization:

- **Forward Pass:** The model uses "hard" quantized values or discrete spikes, mimicking the actual behavior of target hardware.

- **Backward Pass:** To maintain a continuous gradient flow, the system maintains high-precision **"shadow weights."** Smooth mathematical approximations substitute for the non-differentiable discrete functions.

To manage the inherent gradient noise caused by this approximation, the methodology introduces stabilization techniques such as **trust masks** and **smooth function substitutes**. This ensures that the "shadow" updates actually lead to stable convergence in the "hard" model, allowing us to train architectures that would otherwise be impossible to optimize via standard backpropagation.

3. Hardware-Aligned Geometric Mapping

The final pillar of the methodology is the bridge between the mathematical model and the physical silicon. Efficiency is not just about the number of bits; it is about how those bits are laid out in memory.

The approach utilizes **Hardware-Aligned Quantization**. Rather than using arbitrary floating-point numbers, the model maps its representations to specific geometric structures like **symmetric grids, E8 lattices, or binary vector combinations**. These are not chosen at random; they are selected because they represent the **inherent architectural primitives** of the target processors.

By aligning the model's weight geometry with the processor's indexing logic, the methodology bypasses traditional memory bottlenecks and reduces overhead. The geometric symmetry of these lattices ensures that high fidelity is maintained even at extreme levels of quantization, as the mathematical "grid" of the model aligns perfectly with the physical "grid" of the hardware.

Why It Works: The Synergy of Constraints

This unified methodology succeeds because each component solves a problem created by the others:

- **From Data to Gradient:** The informational constraints (Atom 1) ensure that the discrete spikes used by the STE (Atom 2) carry the maximum possible signal, preventing the low-precision model from losing critical information.
- **From Gradient to Hardware:** The STE allows us to optimize the specific, hardware-friendly mathematical structures (Atom 3) that are normally too rigid for standard gradient descent.
- **From Hardware to Performance:** By mapping these optimized weights to architectural primitives, we ensure that the theoretical efficiency of the graph representation (Atom 1) translates into actual reduction in power consumption and latency.

By synthesizing information theory, discrete optimization, and hardware geometry, this approach provides a robust blueprint for the next generation of efficient, graph-aware intelligent systems.

Claude 4.5 Opus – Example Idea 1

Atoms:

- Modeling reasoning as a structured state-transition graph—where discrete nodes represent knowledge states and edges represent logical transitions—enables the quantification of computational complexity and the isolation of error propagation, allowing for the systematic debugging of logical failures at specific points of task decomposition or execution.
- Iterative trajectory refinement in large language models—driven by on-policy sampling, multi-dimensional reward filtering, and structured exploration of the reasoning state-space—transforms reasoning from a static prediction task into a self-evolving search process, enabling models to autonomously identify, recover from, and prioritize high-success logical paths while minimizing cumulative regret.
- Trajectory-level verification—achieved through step-wise reward modeling, consistent credit assignment, and localized error detection—transforms complex reasoning from a binary outcome into a quantifiable sequence of transitions, enabling models to optimize computational resources and maintain logical fidelity by identifying exactly where a chain of thought deviates from a valid path.

Research Idea:
Beyond Static Prediction: Engineering Self-Evolving Reasoning via State-Transition Search

Traditional Large Language Models (LLMs) often treat reasoning as a "one-shot" sequence generation task. When a model encounters a complex problem, it predicts the next token until it reaches a conclusion. However, this linear approach suffers from a critical flaw: once the model makes a single logical error, the mistake propagates, leading to a total collapse of the reasoning chain.

To overcome this, we propose a methodology that shifts reasoning from static prediction to a **self-evolving search process**. By modeling reasoning as a structured state-transition graph and applying iterative trajectory refinement, we can transform how models navigate complex logical landscapes.

The Core Framework: Reasoning as a State-Transition Graph

The foundation of this methodology is the conceptualization of reasoning not as a string of text, but as a **structured state-transition graph**.

In this framework, discrete nodes represent specific "knowledge states" or intermediate checkpoints in a thought process. Edges represent the logical transitions or steps taken to move from one state to the next. By imposing this structure, we can:

- **Quantify Computational Complexity:** We can measure the density and depth of the graph required to reach a solution.
- **Isolate Error Propagation:** Instead of viewing a failed output as a monolithic error, we can pinpoint the exact edge (transition) where the logic failed.
- **Systematic Debugging:** Task decomposition becomes a matter of identifying which specific nodes in the execution path are prone to failure, allowing for targeted optimization.

The Mechanism: Iterative Trajectory Refinement

With the state-transition graph as our map, the model does not just traverse a path once. Instead, it engages in **iterative trajectory refinement**. This process treats the discovery of a solution as an evolutionary search.

1. On-Policy Sampling and Exploration

The model begins by generating multiple potential reasoning paths through on-policy sampling. This involves a structured exploration of the reasoning state-space, where the model intentionally probes different logical branches rather than settling for the most likely token sequence.

2. Trajectory-Level Verification

As these paths (trajectories) are generated, they are subjected to **step-wise reward modeling**. This is a granular verification process that provides consistent credit assignment to individual transitions.

- **Localized Error Detection:** By evaluating the validity of each transition between nodes, the model identifies exactly where a chain of thought deviates from a valid path.
- **Quantifiable Sequences:** This transforms reasoning from a binary "correct/incorrect" outcome into a quantifiable sequence of transitions, where each step has a specific value.

3. Multi-Dimensional Reward Filtering

Once multiple trajectories are sampled and verified, the system applies multi-dimensional reward filtering. This mechanism prunes the search space by discarding paths with low logical fidelity and prioritizing those with high-success potential. This "filtering" ensures the model focuses its computational resources on the most promising logical avenues.

How the Components Synchronize

These mechanisms work in a continuous loop to drive model self-evolution. The **state-transition graph** provides the environment; the **trajectory-level verification** acts as the sensor identifying the quality of movement within that environment; and the **iterative refinement** acts as the motor that optimizes the path.

As the model samples more trajectories, it autonomously learns to identify and recover from logical pitfalls. By minimizing "cumulative regret"—the sum of errors made across the reasoning process—the model evolves from a simple generator into a search agent. It learns to prioritize high-success logical paths by recognizing the "nodes" that historically lead to successful "states."

Why This Works: From Static to Dynamic

This methodology solves the problem of "brittle" reasoning through three key shifts:

1. **From Global to Local Accountability:** By using step-wise rewards and consistent credit assignment, we no longer penalize a model for an entire wrong answer if only the final step was flawed. Conversely, we don't reward a lucky guess if the intermediate logic was broken.
2. **Resource Optimization:** By identifying exactly where a path deviates, the model can optimize computational resources, focusing its "thinking" on repairing specific transitions rather than regenerating the entire chain from scratch.
3. **Autonomous Recovery:** Because the process is iterative and driven by search, the model develops the ability to identify its own errors mid-stream and pivot to a more successful logical path, effectively "self-debugging" during the inference process.

By combining the structural rigor of graph theory with the flexibility of reinforcement-learning-driven search, we move closer to models that don't just mimic reasoning, but actively navigate and refine it.

Claude 4.5 Opus – Example Idea 2

Atoms:

- Iterative trajectory refinement—achieved by contrastive analysis of successful and failed reasoning branches, step-wise curriculum backtracking, and the pruning of redundant logical operations—optimizes model performance by shifting training from simple outcome-matching to the mastery of structural logic, error diagnosis, and computational efficiency.
- Monte Carlo Rollouts serve as a probabilistic evaluation framework for multi-step reasoning by sampling potential future completions to estimate the conditional probability of success for any intermediate state; this mechanism enables models to pinpoint logical failure points, prioritize informative training samples, and synthesize optimal reasoning paths through look-ahead heuristics.
- Dynamic Inference Interventions—comprising early-path pruning, real-time verification gates, and strict token-budget constraints—optimize the performance of generative models by actively filtering the latent distribution of outputs to isolate high-quality reasoning 'tails' while preventing the propagation of hallucinations and the computational waste of redundant paths.

Research Idea:
Beyond Outcome Matching: A Unified Framework for Structural Reasoning Optimization

In the current landscape of large language model (LLM) development, we are witnessing a shift from simple imitation to the mastery of complex, multi-step reasoning. Traditional training often relies on outcome-based rewards—simply checking if the final answer is correct. However, this approach ignores the "internal logic" of the model, leading to hallucinations and computational inefficiency.
We propose a unified methodology that treats reasoning not as a single generation, but as a dynamic search problem. By combining **Iterative Trajectory Refinement**, **Monte Carlo Rollouts**, and **Dynamic Inference Interventions**, we can move toward a model that fundamentally understands structural logic and error diagnosis.

The Problem: The "Black Box" of Reasoning

Most models suffer from three primary failures in complex tasks:

1. **Logical Drifting:** A single error in an intermediate step propagates, leading to an incorrect conclusion.
2. **Computational Waste:** The model spends equal tokens on trivial steps and critical logical junctions.
3. **Sparse Feedback:** Training on final outcomes provides no signal on *where* a reasoning chain actually went wrong.

The Integrated Approach: Search-Augmented Training and Inference

Our methodology unifies these challenges into a single pipeline. We treat the reasoning process as a tree of potential paths. We use probabilistic sampling to evaluate these paths, iterative refinement to learn from the results, and active interventions to ensure the final output stays within the "high-quality tail" of the distribution.

1. Evaluating the State: Monte Carlo Rollouts

The foundation of this methodology is a probabilistic evaluation framework. Instead of guessing if an intermediate reasoning step is good, we use **Monte Carlo Rollouts**.
For any given intermediate state, the model samples multiple potential future completions. By calculating the conditional probability of success across these samples, we can estimate the "value" of that specific state. This mechanism allows the system to:

- Pinpoint exactly where a logical failure occurs.
- Prioritize the most informative samples for training.
- Synthesize an optimal path by looking ahead to see which branches lead to successful outcomes.

2. Mastering the Logic: Iterative Trajectory Refinement

Once we have identified successful and failed paths via rollouts, we use **Iterative Trajectory Refinement** to optimize the model's underlying logic. This is not merely about reaching the right answer; it is about mastering the structure of the argument.

- **Contrastive Analysis:** The model compares successful reasoning branches against failed ones to learn the difference between sound logic and subtle hallucinations.
- **Curriculum Backtracking:** When a failure is detected, the model backtracks to the last valid step, creating a step-wise curriculum that focuses on recovering from errors.
- **Pruning Redundancy:** By identifying and removing unnecessary logical operations, the model learns to prioritize computational efficiency, shifting the training focus from simple outcome-matching to streamlined structural logic.

3. Execution and Control: Dynamic Inference Interventions

The final layer of the methodology applies these insights in real-time during model inference. We don't just let the model run blindly; we use **Dynamic Inference Interventions** to actively manage the latent distribution of outputs.

- **Verification Gates:** At critical junctions, the model passes through "gates" that verify the logical integrity of the path before proceeding.
- **Early-Path Pruning:** If a trajectory shows a low probability of success (as determined by our rollout heuristics), it is terminated early to save resources.
- **Token-Budget Constraints:** By enforcing strict limits on the computational "spend" for each path, we force the model to isolate high-quality reasoning "tails" and prevent the propagation of redundant or circular logic.

Why It Works: A Feedback Loop of Logic

This methodology works because it creates a closed loop between evaluation, learning, and execution.

Monte Carlo Rollouts provide the ground truth for what a "good" step looks like. **Iterative Trajectory Refinement** bakes that knowledge into the model's parameters by forcing it to analyze its own mistakes. Finally, **Dynamic Inference Interventions** ensure that the model's behavior at runtime matches the structural rigor established during training.

By shifting the focus from "what is the answer" to "how do we get there efficiently," this approach isolates the high-quality reasoning paths that often get lost in the noise of standard generative distributions. The result is a model that doesn't just mimic human-like text, but actively diagnoses its own errors and optimizes its own logical structures.