DEX: DATA-EMERGENT EXPERTS FOR CAPABILITY ABLATION IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern language models are trained on vast corpora that inevitably contain harmful or dual-use content, and a capability that helps a legitimate user can just as easily assist a malicious one. A safe deployment therefore benefits from the ability to strip a specific capability from a trained model on demand, without paying to retrain and without degrading everything else the model does. We propose **DEX** (**D**ata-**E**mergent E**x**perts), a dense model inspired by the Mixture-of-Experts architecture but with an embedding-based router instead of a learned one. We first cluster the training corpus by each document's meaning using a pretrained embedding model, letting the data's own structure decide both the categories and which documents belong to each, rather than relying on a predetermined set of topics and fixed assignments. Each cluster is assigned its own expert, and unlike standard sparse routing, all experts stay enabled: every non-ablated expert contributes to each input, weighted by a softmax over the similarity between the input embedding and the cluster centroids. Removing a capability is then as simple as zeroing the corresponding experts. We study how to balance capability removal against the retention of knowledge we want to keep, and we find each cluster captures a specific interpretable specialization. **DEX** forgets far more of the target capability while marginally affecting the rest. Additionally, since knowledge is split into finer clusters, **DEX** can remove narrower subspaces where prior methods can only drop whole domains.

1 INTRODUCTION

A model trained on a broad slice of the web learns the patterns embedded in the data, indiscriminately, and web-scale corpora are known to carry harmful and dual-use material (Luccioni & Viviano, 2021; Dodge et al., 2021). A single model can therefore absorb dual-use knowledge. The chemistry that designs a therapeutic can be redirected toward a chemical warfare agent (Urbina et al., 2022). The biology that accelerates vaccine platforms shares ground with pathogen engineering (Sandbrink & Koblentz, 2022; Drew & Mueller-Doblies, 2017). Today's language models are capable enough to guide a non-expert through parts of that work (Sandbrink, 2023; Gopal et al., 2023; Li et al., 2024). Furthermore, the expertise that defends a network can equally breach one, increasingly with little human involvement (Truong et al., 2020; Fang et al., 2024). This represents the dual-use problem (Brundage et al., 2018; Roland et al., 2026). A provider would ideally remove the unwanted capability while keeping the rest of the model intact, and without retraining from scratch (Roland et al., 2026). Doing so is hard: a capability is not kept in one place but diffused across the weights, entangled with the abilities one means to keep (Elhage et al., 2022).

Several families of methods target this goal. Machine unlearning edits or fine-tunes a trained model to suppress a target behavior (Geng et al., 2025), for example by controlling internal representations (Li et al., 2024), by optimizing a negative preference objective (Zhang et al., 2024), or, in Mixture-of-Experts (MoE) models, by targeting the single expert most responsible for the behavior (Zhuang et al., 2025). Preference-based alignment instead teaches the model to refuse harmful requests (Bai et al., 2022). Data filtering removes the unwanted content before training, so the capability is never learned in the first place (O'Brien et al., 2026). Each has drawbacks. Unlearning and refusal often suppress a behavior rather than erase the underlying knowledge, so it can resurface under adversarial fine-tuning (Łucki et al., 2024), while data filtering commits to the choice at pretraining time and needs a fresh, expensive run for every different capability profile.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

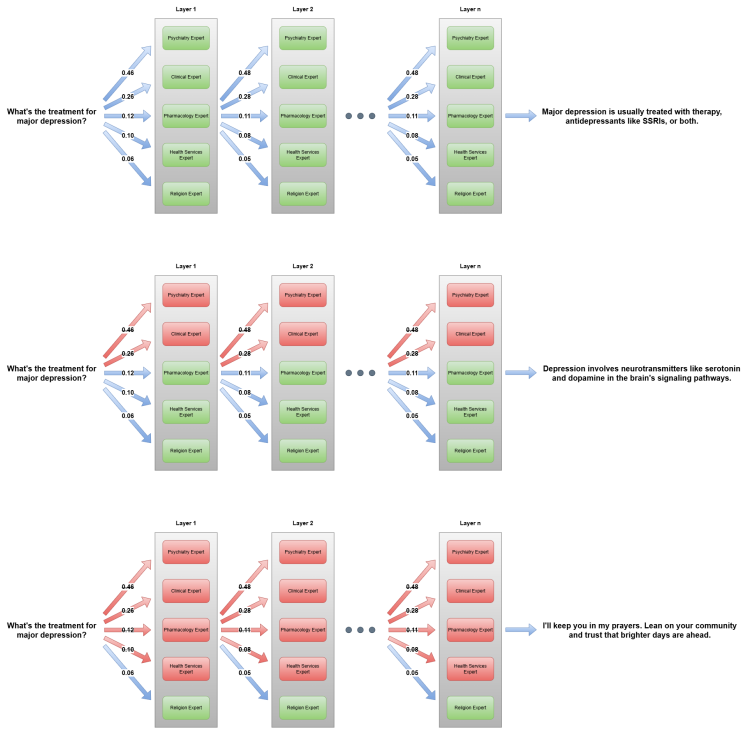


Figure 1: The **DEX** capability-ablation mechanism. The same prompt is run through the model three times, changing only which experts are enabled. Each expert's weight is a softmax over the similarity between the prompt embedding and the cluster centroids associated with each expert. In the first pass no experts are removed and the model answers as usual. Zeroing the experts that carry the target capability shifts the answer rather than breaking it: with the removed experts contributing nothing, the response is recomputed from the remaining ones and speaks from the perspective of the remaining experts. In the second pass two experts are ablated and in the third, four, moving the answer further from the removed capabilities.

Closest to our goal, a line of work localizes a capability into a known part of the network during training so it can later be removed by ablation, either by masking gradients to steer data toward chosen parameters (Cloud et al., 2024) or by routing predefined domains to dedicated modules (Roland et al., 2026). Our method shares this localize-then-ablate strategy but differs in how the specialization is decided: instead of fixing the domains and their routing by hand, we let a statistical clustering of the corpus define both.

We propose **DEX** (**D**ata-**E**mergent E**x**perts), a dense model inspired by the MoE architecture (Jacobs et al., 1991; Shazeer et al., 2017; Dai et al., 2024) but with an embedding-based router instead of a learned one. Building on prior methods that specialize experts to slices of the training data (Gururangan et al., 2022; Li et al., 2022; Gururangan et al., 2023), we first cluster the pretraining corpus by each document's meaning using a pretrained sentence embedding model (Reimers & Gurevych, 2019) and assign one expert to each cluster, so the data's own structure decides both the categories and which documents belong to each, rather than a predetermined set of topics with fixed assignments. Unlike standard sparse routing, all experts stay enabled: every non-ablated expert contributes to each input, weighted by a softmax over the similarity between the input embedding and the cluster centroids. To remove a capability, we zero one or more experts.

Our contributions are as follows:

- We show that a predetermined, hand-defined partition of the data does not match its natural structure. When we embed a domain-labeled corpus, we find that the fixed domains overlap the statistically discovered clusters unevenly, some spanning many clusters and others sharing a single one, so each hand-defined module covers an uneven and uncontrolled slice

of the capability space. We argue that the granularity should be determined by a statistical model over the data, not hardcoded.

- We introduce **DEX** and its capability ablation mechanism, which lets us remove capabilities after training by zeroing the responsible experts while leaving the rest of the model intact.
- We study two rules for selecting which experts to remove, one based on nearest-centroid votes and one based on softmax gate mass, and we analyze how forgetting and retention trade off as the number of removed experts grows.
- We analyze what each cluster represents by inspecting the documents closest to its centroid, showing that each expert captures an interpretable, coherent specialization.

2 Related Work

Machine unlearning. Machine unlearning removes the influence of unwanted data or behavior from an already-trained model, typically by further editing or fine-tuning its weights (Geng et al., 2025). Approaches include perturbing the model's internal representations on the target domain (Li et al., 2024), optimizing a negative preference objective that pushes the model away from the forget set (Zhang et al., 2024), and, for MoE models, concentrating the update on the single expert most responsible for the target knowledge so utility elsewhere is preserved (Zhuang et al., 2025). These methods share a common weakness: they tend to suppress a capability rather than erase the knowledge behind it, and the capability can be revived by light adversarial fine-tuning, often on data unrelated to the forget set (Łucki et al., 2024).

Refusal training and data filtering. Preference-based alignment teaches a model to refuse harmful requests (Bai et al., 2022), but this only gates access to knowledge the weights still contain, and the gate can be bypassed with jailbreaks (Wei et al., 2023; Zou et al., 2023). Data filtering goes further upstream, stripping the unwanted content from the corpus so the capability is never learned, which yields strong tamper resistance (O'Brien et al., 2026). The cost is flexibility: the decision is fixed at pretraining time, and serving a different capability profile requires curating the data and training a fresh model from scratch.

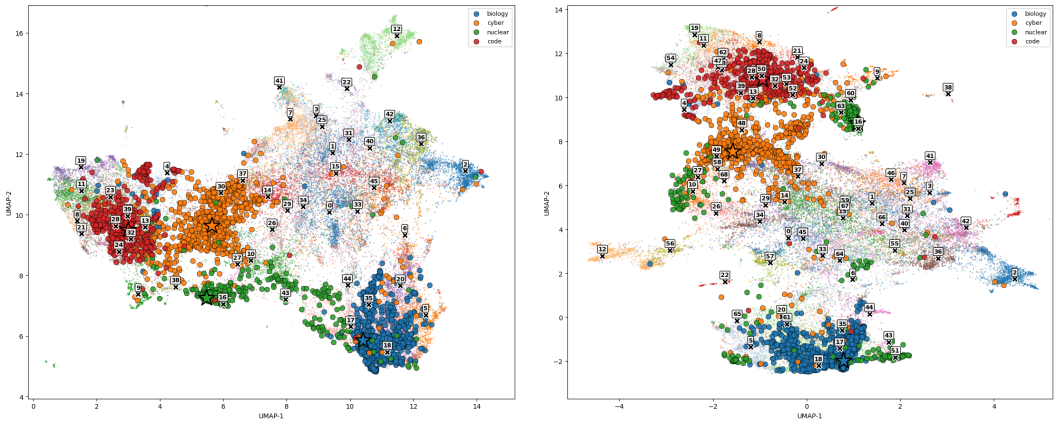
Modular pretraining for capability removal. Closest to our goal is work that localizes a capability into a known part of the network during training, so that ablating that part at inference removes the capability while leaving the rest intact. Gradient routing masks gradients to steer specified data toward chosen parameters, which can then be ablated for removal (Cloud et al., 2024), and GRAM routes predefined dual-use domains to dedicated auxiliary modules that are switched off at deployment (Roland et al., 2026). **DEX** adopts this localize-then-ablate strategy but differs in how the specialization is decided: rather than fixing the domains and their routing by hand, we let a statistical clustering of the corpus determine both the number of experts and what each one covers.

3 Method

DEX is built in three stages. We first embed the pretraining corpus and cluster it to discover a set of domains and their centroids (§3.1). We then attach one expert to each cluster inside a single dense model, trained jointly from scratch. Each expert's contribution is weighted by the similarity between the input's embedding and the centroids (§3.2). Finally, to remove a capability, we identify the experts that carry it and zero them (§3.3).

3.1 Embedding and Clustering the Corpus

While earlier modular methods split the corpus by document metadata (Gururangan et al., 2022; Li et al., 2022), we group documents by their meaning. Given a corpus $\mathcal{D} = \{d_1, \ldots, d_N\}$, each document d_i is encoded with a frozen pretrained sentence embedding model (Reimers & Gurevych, 2019). We then cluster the embeddings to discover domains, following the idea of defining experts by clustering the corpus rather than by fixed metadata (Gururangan et al., 2022; Li et al., 2022; Gururangan et al., 2023). Each resulting cluster is associated with one expert, and its centroid c_k is the L_2-normalized mean of the embeddings assigned to it.



(a) 50M parameters (b) 500M parameters

Figure 2: Clustering of the pretraining corpus at 50M parameters (left) and 500M parameters (right). We embed every document in the pretraining corpus and cluster the embeddings with a cosine-distance threshold of $\tau = 0.95$, which yields 46 clusters at 50M parameters and 69 clusters at 500M parameters, each becoming one expert. For visualization we project a random sample of 40,000 documents to two dimensions with UMAP (McInnes et al., 2018)

Setting the number of clusters with a distance threshold. Rather than fixing the expert count K ahead of time, we let it emerge from the data through a distance threshold. We use a leader (threshold) clustering pass: we scan the embeddings once, and each embedding joins the nearest existing centroid if its cosine distance to that centroid is below a threshold τ, and otherwise seeds a new cluster. The threshold τ is the only knob: a smaller τ yields more, finer clusters, and a larger τ yields fewer, coarser ones, so the granularity is chosen by a single interpretable distance metric.

3.2 The DEX Architecture

DEX is a decoder-only transformer in which the feed-forward block of every layer is replaced by a set of K experts, one per cluster, all of equal size. It is inspired by the Mixture-of-Experts (MoE) architecture (Jacobs et al., 1991; Shazeer et al., 2017; Dai et al., 2024), but departs from it in two ways: the router is not learned, and the model is dense.

Embedding-based routing. For an input sequence with document embedding e (obtained using the same frozen encoder as §3.1), we compute a gate over the K experts as a temperatured softmax of the cosine similarity between e and each centroid c_k,

$$g_k = \frac{\exp\left(e^\top c_k / T\right)}{\sum_{j=1}^{K} \exp\left(e^\top c_j / T\right)}, \tag{1}$$

where T is a temperature that controls how sharply the gate concentrates on the nearest centroids. The router therefore has no trainable parameters: it is the embedding model plus the frozen centroids, in the spirit of the centroid-distance routing used by Gururangan et al. (2023). The gate $g \in \Delta^{K-1}$ is computed once per sequence and shared across all layers and token positions.

Dense expert combination. Let FFN_k denote the k-th expert in a given block, a standard two-layer feed-forward network. The block output combines *all* experts, weighted by the gate,

$$\mathrm{MoE}(h) = \sum_{k=1}^{K} g_k \,\mathrm{FFN}_k(h), \qquad \mathrm{FFN}_k(h) = \left(\sigma(hW_1^k + b_1^k)\right)W_2^k + b_2^k, \tag{2}$$

where h is the block's post-attention hidden state, and σ is a GELU nonlinearity. Unlike the top-k routing of sparse MoE models, no expert is dropped: every expert contributes to every input. We keep all experts active because a single prompt typically draws on more than one domain, so the

model should blend the relevant experts rather than commit to a few; this dense combination is also what makes removal clean, since each expert contributes to Eq. equation 2 as its own additive, gated term. The whole model, experts included, is trained jointly from scratch, so the experts co-adapt within one network rather than being trained separately and combined after the fact.

3.3 Capability Ablation

Because each expert contributes to the block output as a separate additive term (Eq. equation 2), removing a capability is as simple as zeroing the experts that carry it. Given a set $R \subseteq \{1, \dots, K\}$ of experts to remove, we set their gate entries to zero,

$$\tilde{g}_k = \begin{cases} 0 & k \in R \\ g_k & k \notin R, \end{cases} \tag{3}$$

and use $\tilde{g}$ in place of g in every block. This removes the targeted experts' contribution from the model entirely, in the same localize-then-ablate spirit as modular pretraining methods that disable a dedicated part of the network at inference (Cloud et al., 2024; Roland et al., 2026). Optionally, the contributions of the non-ablated experts can be renormalized. We conduct experiments under both configurations, with and without renormalization.

The remaining question is which experts to place in R for a target capability. We assume a small set of probe prompts $\mathcal{P}$ representative of the capability, and study two rules for scoring experts against it.

Nearest-centroid votes (coverage). We route each probe to its single closest centroid and count how often each expert wins,

$$v_k = \left| \{\, p \in \mathcal{P} : \arg\max_j \, e_p^\top c_j = k \,\} \right|. \tag{4}$$

Experts are ranked by their vote count v_k. Intuitively, this ranks highest the experts that the capability's prompts most often land on.

Gate mass (cumulative). Instead of hard nearest-centroid assignments, we average the soft gate (Eq. equation 1) that each expert receives across the probe set,

$$\bar{g}_k = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} g_k(p). \tag{5}$$

Experts are ranked by $\bar{g}_k$. This accounts for the full weighting the model places on each expert, not only which centroid is nearest.

Both rules yield an ordered list of experts, so we can also sweep the number removed, $|R|$, to trace how forgetting the target capability trades off against retaining everything else.

4 Experimental Setup

4.1 Training

We train **DEX** from scratch on The Pile (Gao et al., 2020), in its deduplicated form, a diverse mix of web text, code, and scientific writing. We use no domain labels: the experts are defined purely by clustering the corpus embeddings (§3.1), so the model never sees which documents belong to which category. We report results at 50M and 500M parameters, and size the training set to be Chinchilla-optimal (Hoffmann et al., 2022). All models use a context length of 1024 tokens.

Document embeddings, used both to form the clusters and to compute the routing gate, come from a frozen `all-MiniLM-L6-v2` sentence encoder (Reimers & Gurevych, 2019; Wang et al., 2020). We optimize with a Muon (Liu et al., 2025) and AdamW (Loshchilov & Hutter, 2017) pairing: Muon updates the two-dimensional weight matrices, while AdamW handles the remaining parameters (embeddings, normalization, and biases). The two main **DEX**-specific hyperparameters are the gate temperature, which we set to $T = 0.1$, and the cosine-distance threshold τ used for clustering, which we set to 0.95. On The Pile this threshold yields $K = 46$ experts on 50M and 69 experts on 500M.

4.2 BASELINES

We compare against GRAM (Roland et al., 2026), the closest prior method for capability removal. GRAM is a transformer in which each layer keeps one always-active core feed-forward network alongside one small auxiliary module per domain; during training, gradient routing sends each domain's data to its own module, and at inference a capability is removed by ablating the corresponding module. Unlike **DEX**, GRAM needs the data to be labeled by domain, since the routing that induces specialization is driven by those labels rather than by clustering.

We reproduce GRAM faithfully and train it using the original data recipe: a core mixture of FineWeb-Edu (Penedo et al., 2024) and four labeled auxiliary domains, biology, cybersecurity, nuclear physics and code (extracted from The Stack (Kocetkov et al., 2022)), with one auxiliary module per domain. We give GRAM the same parameter size and the same Chinchilla-optimal token budget as **DEX**, so both models see an equal number of tokens and the comparison is compute-matched.

4.3 EVALUATION

For biology, cybersecurity, and nuclear physics we use GRAM's curated domain probe sets, and for code we sample Python snippets from The Stack (Kocetkov et al., 2022).

Following the original GRAM paper (Roland et al., 2026), we report the mean per-token log-probability the model assigns to the gold (next) token, measured in nats. Higher is better, and it is monotonically related to perplexity, so a lower loss is a higher log-probability and a lower perplexity. For a given removal we report the following:

- Δ_{Forget}: the target domain's own probes. A large positive value means the capability was thoroughly removed.
- Δ_{Retain}: every other domain's probes. A value near zero means unrelated capabilities were preserved.
- Δ_{Core}: a general held-out slice drawn from the training distribution but disjoint from the training documents. A value near zero means general ability was preserved.

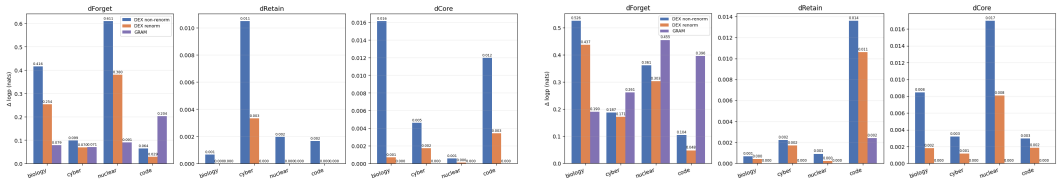
A clean removal is therefore a large Δ_{Forget} with Δ_{Retain} and Δ_{Core} near zero. For **DEX** we ablate experts using both selection rules, the nearest-centroid (coverage) rule and the gate-mass (cumulative) rule, and we additionally sweep the number of experts removed to trace how Δ_{Forget} grows against Δ_{Retain} and Δ_{Core}.

5 RESULTS

5.1 CLUSTERING

The larger dots in Figure 2 are the probes used for evaluation, one group per tested capability. These probes are drawn from the same domain data that GRAM trains its auxiliary modules on (Roland et al., 2026), so each group corresponds one-to-one with a GRAM capability: biology, nuclear, cybersecurity, and code. Overlaying them on our clusters allows us to analyze, for each of GRAM's hand-defined domains, where that domain actually lands in the corpus once the data is grouped by meaning rather than by a hardcoded topic.

The picture shows a clear mismatch between the two ways of carving up the data. A domain that GRAM treats as a single indivisible unit spreads across many of our clusters, and crucially the number it spreads over is uneven from one domain to the next: at 50M, the biology probes fall across roughly 5 of our centroids, cybersecurity across about 5, code across about 7, while nuclear collapses onto only about 2. A fixed, per-domain module therefore commits to a granularity that does not match the data. Some domains are far broader than one module can cleanly hold, so a single module ends up absorbing what is really several distinct sub-topics, while others are narrow enough that a full module is more than they need. Letting the granularity fall out of a distance threshold instead gives each region of the corpus the number of experts its spread warrants, which is what lets us remove a capability at the resolution the data actually has.



(a) 50M parameters

(b) 500M parameters

Figure 3: Single-unit capability removal, **DEX** versus GRAM, at 50M (left) and 500M (right). Three panels report Δ_{Forget}, Δ_{Retain}, and Δ_{Core} (mean per-token gold log-probability change, in nats), with grouped bars over the four capabilities. Each capability has three bars: **DEX** non-renormalized, **DEX** renormalized, and GRAM, all at a matched budget of one removed unit, the single top-ranked expert for **DEX** and the domain's auxiliary module for GRAM. At single-expert removal the coverage and cumulative rules pick the same expert, so **DEX** is one bar per variant. A larger Δ_{Forget} is better, while Δ_{Retain} and Δ_{Core} near zero indicate the rest of the model is left intact.

While a human-determined spread is suitable when it comes to evaluating the performance of a capability ablation algorithm, we argue that it is not appropriate in the training regime.

5.2 Baseline Comparison

Figure 3 compares the two methods when each removes a single unit: for **DEX**, the one expert ranked highest for the capability, which is the same expert under both the coverage and cumulative rules, so the two coincide at this point; for GRAM, that domain's dedicated module.

At 50M parameters, **DEX** removes far more on three of the four target capabilities. On nuclear, non-renormalized **DEX** lowers the forget-set log-probability by 0.611 nats against GRAM's 0.091, roughly sevenfold, and on biology by 0.416 against 0.079, close to fivefold. Cybersecurity is comparable, with 0.099 for **DEX** and 0.071 for GRAM. Code is the only exception, where GRAM removes 0.204 against **DEX**'s 0.064.

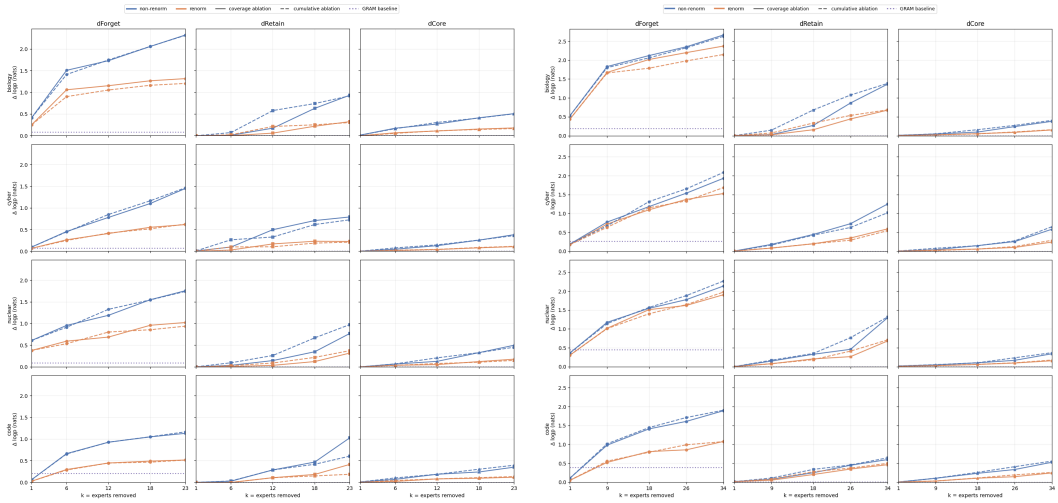
This exception follows from the clustering in Section 5.1. The fewer clusters a capability occupies, the more of it a single expert carries. Nuclear collapses onto about 2 clusters, so removing one expert erases most of it. Code spreads over roughly 7 clusters, so a single removal leaves most of it in place. The sweep in Section 5.3 confirms this, as code's forget gap climbs to match and then exceed the other domains when more of its experts are removed.

At 500M parameters, single-expert removal favors GRAM. **DEX** keeps a clear lead only on biology, with 0.526 nats against 0.190, nearly threefold. GRAM removes more on nuclear (0.455 against 0.361), cybersecurity (0.261 against 0.187), and code (0.396 against 0.104). The same concentration argument explains this. At the larger scale capabilities spread across more experts, so a single expert holds a smaller share of each one. This gap is a property of the one-unit budget rather than of **DEX** itself, and removing a few more experts recovers the advantage (Section 5.3).

Without renormalization, **DEX** changes retain by at most 0.011 nats at 50M and 0.014 at 500M, and core by at most 0.016 at 50M and 0.017 at 500M. With renormalization, retain and core both stay within 0.003 at 50M, while at 500M retain stays within 0.011 and core within 0.008. The forget effect shrinks but remains sizable. On nuclear it drops to 0.380 at 50M and 0.303 at 500M, and on biology to 0.254 at 50M and 0.437 at 500M. In short, **DEX** trades a tiny, controllable amount of collateral for much stronger removal when a capability is concentrated in few experts.

5.3 Scaling the Number of Removed Experts

Figure 4 sweeps the number of removed experts, from 1 to 23 at 50M parameters and from 1 to 34 at 500M parameters. Two trends hold across all four capabilities and both scales. Forgetting increases monotonically with k and quickly goes far beyond anything a single GRAM module reaches. At the top of the sweep, **DEX** lifts Δ_{Forget} to between 1.1 and 2.3 nats at 50M and between 1.9 and 2.7 at 500M, against GRAM ceilings of 0.204 and 0.455. At the same time, the retain and core changes climb with k, so removing more experts buys more forgetting at a rising cost to the rest of the model.



(a) 50M parameters (b) 500M parameters

Figure 4: Effect of removal strength at 50M parameters (left) and 500M parameters (right). We remove $k \in \{1, 6, 12, 18, 23\}$ experts at 50M parameters and $k \in \{1, 9, 18, 26, 34\}$ experts at 500M parameters, and overlay both variants (non-renormalized in blue, renormalized in orange) and both selection rules (coverage solid, cumulative dashed). GRAM's single-module removal is a purple dotted reference. Forgetting rises steadily with k and quickly passes GRAM for every capability at both scales, while retain and core collateral grow more slowly, resulting in a tunable trade-off between how much is removed and how much is preserved.

The sweep therefore exposes an operating curve from which one picks the smallest k that achieves the needed removal.

The forget-to-collateral ratio is most favorable early and degrades as k grows. For biology under coverage at 50M, removing 6 experts raises Δ_{Forget} from 0.416 to 1.511 while retain stays at 0.012 and core at 0.176, whereas 23 experts reach 2.327 but drive retain to 0.936 and core to 0.511. The 500M run follows the same shape. Removing 9 experts lifts biology's forget from 0.526 to 1.830 with retain at only 0.028 and core at 0.043, while 34 experts reach 2.667 but drive retain to 1.367 and core to 0.380. The sweep also resolves the single-expert results of Section 5.2. At 50M, code loses to GRAM at $k = 1$ (0.064 versus 0.204) but reaches 0.666 by $k = 6$, more than triple GRAM. At 500M, where GRAM led on three capabilities, removing 9 experts already puts **DEX** ahead on all four, with code at 0.980 against GRAM's 0.396, about two and a half times more. In both cases the capability is simply spread over more experts and returns once enough of them are removed.

The two rules track each other closely on forgetting and differ only modestly on collateral, with no uniform winner. Coverage tends to incur less retain damage at small budgets, for instance biology and nuclear at $k = 6$ at 50M (0.012 versus 0.074 and 0.036 versus 0.095), and biology at $k = 9$ and $k = 18$ at 500M (0.028 versus 0.145 and 0.270 versus 0.680). The ordering is domain-dependent and can reverse as more experts are removed. On cyber at larger k, cumulative is the cleaner of the two at both scales, for example 1.018 versus 1.251 retain at $k = 34$ at 500M.

By far the largest lever, though, is the renormalization choice. Non-renormalization is the aggressive setting. It forgets more but lets collateral grow, whereas renormalization holds the block's output magnitude fixed and stays much closer to the original model. At 50M the gap is wide. At $k = 23$ on biology, the non-renormalized model forgets 2.327 with retain 0.936 and core 0.511, while the renormalized model forgets 1.320 with retain only 0.326 and core only 0.182, roughly half the forgetting for a third of the collateral. Nuclear follows the same pattern (1.759 forget at retain 0.771, against 1.025 at retain 0.319). At 500M renormalization becomes even more attractive, since it gives up far less forgetting. On biology at $k = 34$ it keeps 2.373 of the 2.667 nats of forgetting while roughly halving retain (0.677 versus 1.367) and cutting core by more than half (0.151 versus 0.380), and nuclear behaves the same way (1.905 forget at retain 0.682, against 2.136 at retain 1.300). Non-renormalization is therefore the choice when the priority is to erase as much of a

Table 1: Final language modeling loss (nats, lower is better) of **DEX** against standard MoE variants and GRAM, at matched parameter count and token budget. Dense MoE uses a learned router with all experts active, and Sparse MoE uses a learned top-8 router.

Model	Final loss (nats)
Dense MoE	2.487
Sparse MoE	2.539
DEX	2.565
GRAM	2.794

capability as possible, and renormalization when preserving the rest of the model matters more. Beyond the first few experts, even the gentler variant forgets far more than any GRAM removal. Together, the number of experts, the selection rule, and the renormalization choice give three knobs for placing **DEX** anywhere along the forget-versus-preserve trade, a flexibility that GRAM's fixed one-module-per-domain design does not offer.

5.4 Language Modeling Quality

Making experts removable should not come at a large cost to the model's general ability. Table 1 compares the final loss of **DEX** against two standard MoE baselines with learned routers, one dense and one sparse, and against GRAM. **DEX** reaches 2.565 nats, only 0.078 above the dense MoE and 0.026 above the sparse MoE. The gap to the dense MoE is the price of replacing a learned router with a fixed, embedding-based one, and it is small relative to what the fixed router buys, namely experts whose specialization is known in advance and can be removed after training. GRAM trails all three at 2.794 nats, 0.229 above **DEX**. **DEX** therefore keeps language modeling quality close to standard MoE designs while offering capability removal.

6 Conclusion

We presented **DEX**, a dense model whose experts emerge from the pretraining corpus and can be ablated. Rather than hardcoding a handful of dual-use domains, we cluster documents by meaning, associate each cluster with an expert, and weight every expert by the similarity between an input's embedding and the cluster centroids. Because every expert enters the block output as its own additive, gated term, taking a capability out of the model reduces to zeroing the experts that carry it, with no additional retraining and no fine-tuning.

Two findings stand out. First, hand-defined domains do not match the structure the data actually has: a single labeled domain scatters across a variable number of discovered clusters, so a fixed one-module-per-domain design commits to a granularity that is too coarse for broad domains and needlessly wide for narrow ones. Letting a distance threshold set the number of experts gives each region of the corpus the resolution its spread warrants. Second, this finer resolution translates into stronger and more controllable removal. At a matched single-unit budget, **DEX** removes far more than GRAM on three of four capabilities at 50M parameters, while leaving retained and core performance almost untouched. At 500M parameters capabilities spread over more experts, so a single expert falls short on three of four, but removing a few more experts puts **DEX** ahead on all of them. The number of experts removed, the selection rule, and the renormalization choice together allow the model to be placed anywhere along the forget-versus-preserve trade. Wherever a single expert underperforms, as for code at 50M parameters, the clustering shows the capability to be spread out, and its removal recovers as soon as more of its experts are dropped. Inspecting the pretraining corpus documents that are nearest to each centroid confirms that the experts settle into coherent, interpretable specializations rather than arbitrary partitions.

These results suggest that the unit of capability removal is better discovered from data than hard-coded in advance. By making each expert both data-emergent and independently removable, **DEX** offers a practical route to serving one trained model at many capability profiles, removing what a given deployment should not expose while preserving the other knowledge that the model possesses.

AI USE STATEMENT

In this work, we used generative AI tools to help implement methods, to assist with writing and editing code, and to improve the readability of the paper. We did not use generative AI tools for any other task with required disclosure. In particular, no research idea, hypothesis, method design, or interpretation of results originated from a generative AI tool.

We have reviewed all AI-assisted work. All AI-assisted code was tested and verified for correctness by the authors, and all AI-assisted text edits were checked to confirm they preserved the intended meaning. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects, personally identifiable information, or the collection of new data. All experiments use publicly available datasets and models, used in accordance with their licenses and intended use. We report our experimental setup and hyperparameters in full so that our results can be independently verified. We declare no conflicts of interest or external sponsorship that could have influenced this work. We do not anticipate direct harmful applications of our method beyond the risks already present in the pretrained models we build on.

REPRODUCIBILITY STATEMENT

We have taken several steps to make our results reproducible. The main text describes our method in full and states the models, datasets, and evaluation protocol used, including the metrics we report and how they are computed. All datasets and pretrained models used in this work are publicly available. We specify the training and inference configurations, including the relevant hyperparameters, so that our setup can be matched. Taken together, these details are intended to be sufficient for an independent reimplementation of our method and for a faithful reproduction of the experiments we report.

REFERENCES

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022.

Miles Brundage, Shahar Avin, Jack Clark, Helen Toner, Peter Eckersley, Ben Garfinkel, Allan Dafoe, Paul Scharre, Thomas Zeitzoff, Bobby Filar, et al. The malicious use of artificial intelligence: Forecasting, prevention, and mitigation. *arXiv preprint arXiv:1802.07228*, 2018.

Alex Cloud, Jacob Goldman-Wetzler, Evžen Wybitul, Joseph Miller, and Alexander Matt Turner. Gradient routing: Masking gradients to localize computation in neural networks. *arXiv preprint arXiv:2410.04332*, 2024.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 1280–1297, 2024.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 1286–1305, 2021.

Trevor W Drew and Uwe Ulex Mueller-Doblies. Dual use issues in research–a subject of increasing concern? *Vaccine*, 35(44):5990–5994, 2017.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

Richard Fang, Rohan Bindu, Akul Gupta, Qiusi Zhan, and Daniel Kang. Llm agents can autonomously hack websites. *arXiv preprint arXiv:2402.06664*, 2024.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Jiahui Geng, Qing Li, Herbert Woisetschlaeger, Zongxiong Chen, Fengyu Cai, Yuxia Wang, Preslav Nakov, Hans-Arno Jacobsen, and Fakhri Karray. A comprehensive survey of machine unlearning techniques for large language models. *arXiv preprint arXiv:2503.01854*, 2025.

Anjali Gopal, Nathan Helm-Burger, Lennart Justen, Emily H Soice, Tiffany Tzeng, Geetha Jeyapragasan, Simon Grimm, Benjamin Mueller, and Kevin M Esvelt. Will releasing the weights of future large language models grant widespread access to pandemic agents? *arXiv preprint arXiv:2310.18233*, 2023.

Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A Smith, and Luke Zettlemoyer. Demix layers: Disentangling domains for modular language modeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5557–5576, 2022.

Suchin Gururangan, Margaret Li, Mike Lewis, Weijia Shi, Tim Althoff, Noah A. Smith, and Luke Zettlemoyer. Scaling expert language models with unsupervised domain discovery, 2023.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, et al. The stack: 3 tb of permissively licensed source code. *arXiv preprint arXiv:2211.15533*, 2022.

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. Branch-train-merge: Embarrassingly parallel training of expert language models. *arXiv preprint arXiv:2208.03306*, 2022.

Nathaniel Li, Alexander Pan, Anjali Gopal, Summer Yue, Daniel Berrios, Alice Gatti, Justin D Li, Ann-Kathrin Dombrowski, Shashwat Goel, Long Phan, et al. The wmdp benchmark: Measuring and reducing malicious use with unlearning. *arXiv preprint arXiv:2403.03218*, 2024.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Alexandra Luccioni and Joseph Viviano. What's in the box? an analysis of undesirable content in the common crawl corpus. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 182–189, 2021.

Jakub Łucki, Boyi Wei, Yangsibo Huang, Peter Henderson, Florian Tramèr, and Javier Rando. An adversarial perspective on machine unlearning for ai safety. *arXiv preprint arXiv:2409.18025*, 2024.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Kyle O'Brien, Stephen Casper, Quentin Anthony, Tomek Korbak, Robert Kirk, Xander Davies, Ishan Mishra, Geoffrey Irving, Yarin Gal, and Stella R Biderman. Deep ignorance: Filtering pretraining data builds tamper-resistant safeguards into open-weight llms. In *International Conference on Learning Representations*, volume 2026, pp. 35579–35633, 2026.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 30811–30849. Curran Associates, Inc., 2024. doi: 10.52202/079017-0970.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Ethan Roland, Murat Cubuktepe, Erick Martinez, Stijn Servaes, Keenan Pepper, Mike Vaiana, Diogo Schwerz de Lucena, Judd Rosenblatt, Addie Foote, Cem Anil, et al. Modular pretraining enables access control. *arXiv preprint arXiv:2607.08077*, 2026.

Jonas B Sandbrink. Artificial intelligence and biological misuse: Differentiating risks of language models and biological design tools. *arXiv preprint arXiv:2306.13952*, 2023.

Jonas B Sandbrink and Gregory D Koblentz. Biosecurity risks associated with vaccine platform technologies. *Vaccine*, 40(17):2514–2523, 2022.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017.

Thanh Cong Truong, Quoc Bao Diep, and Ivan Zelinka. Artificial intelligence in the cyber domain: Offense and defense. *Symmetry*, 12(3):410, 2020.

Fabio Urbina, Filippa Lentzos, Cédric Invernizzi, and Sean Ekins. Dual use of artificial-intelligence-powered drug discovery. *Nature Machine Intelligence*, 4(3):189–191, Mar 2022. ISSN 2522-5839. doi: 10.1038/s42256-022-00465-9.

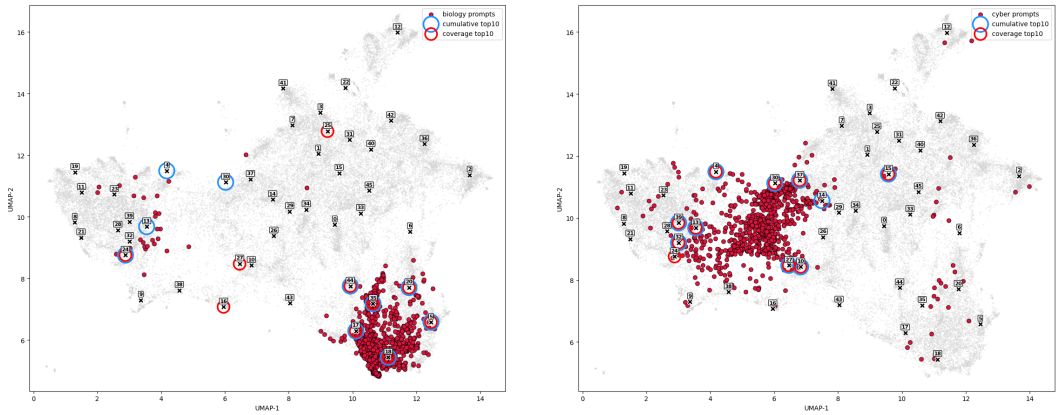
Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, 33:5776–5788, 2020.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 80079–80110. Curran Associates, Inc., 2023. doi: 10.52202/075280-3508.

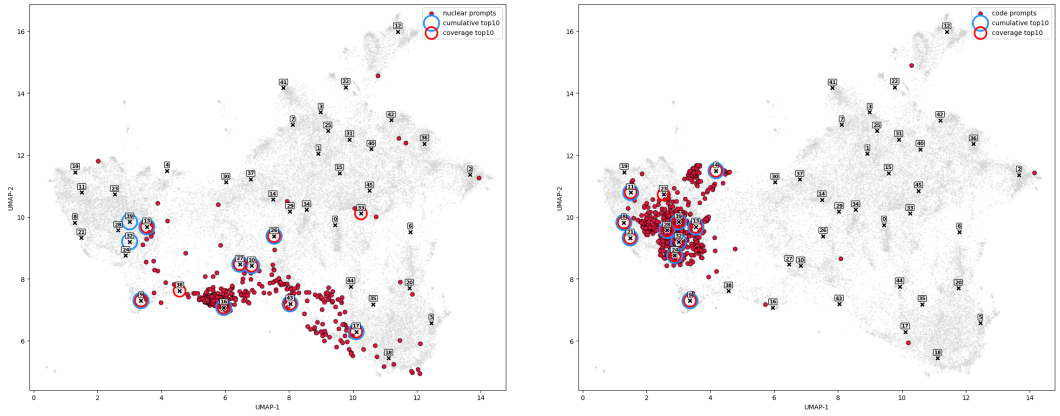
Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. Negative preference optimization: From catastrophic collapse to effective unlearning. *arXiv preprint arXiv:2404.05868*, 2024.

Haomin Zhuang, Yihua Zhang, Kehan Guo, Jinghan Jia, Gaowen Liu, Sijia Liu, and Xiangliang Zhang. Seuf: Is unlearning one expert enough for mixture-of-experts llms? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8664–8678, 2025.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.



(a) Biology (b) Cybersecurity



(c) Nuclear (d) Code

Figure 5: Top-10 experts selected for removal by each rule, per capability. Each figure is the same UMAP projection of the pretraining corpus with the 46 cluster centroids marked by numbered crosses, overlaid with the category's evaluation prompts (red). Circles mark the ten experts each rule ranks highest: coverage in red and cumulative in blue. The two rules agree on the dominant clusters and differ only at the margins, where coverage reaches isolated outlying prompts while cumulative reaches centroids bordering the dense cloud.

A Ablation Methods Visualization

Figure 5 shows, for each capability, the ten experts that each selection rule of §3.3 ranks highest, laid over where that capability's prompts fall. For the most part the two rules pick the same experts: in every case both circle the dense centroids at the heart of the prompt cloud, and for code, which is tightly concentrated in one corner, their top-10 sets are almost identical. The rankings agree here because both rules capture the same thing, how much a capability depends on each expert, so its most relevant experts come out on top either way.

The disagreements between the two methods are confined to the edges of the prompt cloud, and follow a consistent pattern. Coverage counts hard nearest-centroid assignments, so it rewards any expert that is the closest centroid to a group of prompts, even a small one far from the main mass; its extra picks are therefore isolated, outlying centroids, such as 16, 25, and 27 for biology and 33 for nuclear, each sitting alone in a sparse region under a few stray prompts. Cumulative instead sums the soft gate mass an expert receives, which rewards proximity to many prompts rather than being the top choice for any, so its extra picks are centroids next to the dense clusters that collect gate weight from surrounding prompts even though those prompts are nearest to other centroids, such as 4, 13, and 30 around biology's smaller group and 39 and 32 along the edge of nuclear's band.

This margin-level difference is exactly what the aggregate results reflect. Because the two rules share the experts that carry most of a capability, they track each other closely on forgetting throughout the sweep of Section 5.3. Since they diverge only on a few peripheral experts, their collateral also stays close throughout, differing slightly and without a consistent winner as more experts are removed.

B Interpreting the Training Corpus Clusters

Table 2: Cluster taxonomy for the 46 discovered experts, grouped into a three-level hierarchy of domain, subdomain, and cluster. For each cluster, average distance and average confidence are computed over the 10 training prompts closest to its centroid: Avg. Dist. is the mean cosine distance of those prompts to the centroid, and Avg. Conf. is the mean gate probability assigned to the corresponding expert over the same 10 prompts. The gate uses temperature 0.1. Lower distance and higher confidence indicate a more coherent, well-separated cluster.

Cluster	Avg. Dist.	Avg. Conf.
Software and Programming		
Systems and Low-Level Programming		
C Libraries	0.1979	0.9442
C/C++ Headers	0.2664	0.8267
Scripting and Build Tooling	0.3497	0.7658
Web Programming		
PHP	0.3287	0.8472
C# and Java	0.3120	0.8539
JavaScript	0.3168	0.8714
HTML	0.3057	0.9177
Programming Languages and Algorithms		
Typed-Language Programming	0.3782	0.7309
Algorithms and Code Golf	0.3844	0.6999
Date/Time Programming	0.3661	0.8496
Distributed Systems		
Network Programming and Distributed Systems	0.3790	0.7947
Life Sciences and Medicine		
Clinical		
Clinical and Surgical Research	0.3895	0.7897
Infectious Disease and Epidemiology	0.3351	0.8594
Mental Health		
Psychiatry and Cognitive Neuroscience	0.3565	0.8555
Research Biology		
Cancer and Cell Biology	0.3628	0.7971
Molecular Biology and Drug Discovery	0.4087	0.7505
Taxonomy and Phylogenetics	0.3060	0.8841
Health Systems		
Community Health and Public Services	0.3528	0.6990
Physical Sciences and Engineering		
Physics		
Astrophysics and Particle Physics	0.3347	0.8496
Chemistry and Materials		
Natural Products and Analytical Chemistry	0.3866	0.7917
Batteries and Energy Storage	0.3403	0.8655
Devices		
Mechanical Apparatus	0.3222	0.8433
Optical and Display Devices	0.3024	0.8179
Mathematics		

Cluster	Avg. Dist.	Avg. Conf.
Algebra	0.2143	0.9531
Politics, Law, and Current Affairs		
Governance		
US Electoral Politics	0.2649	0.8316
European Politics	0.3161	0.8529
Law		
US Federal Court	0.2005	0.9305
Conflict		
Terrorism and Militant Attacks	0.3058	0.8603
Public Opinion		
Political and Social Commentary Blogs	0.3131	0.7068
Technology		
Industry		
Fintech and Tech Industry Analysis	0.3106	0.6959
Consumer Tech News and Cybersecurity	0.3948	0.5564
Social Media		
Social Media and Online Privacy	0.3068	0.7521
Media and Entertainment		
Criticism		
Film, TV, and Music Reviews	0.3500	0.6452
Music		
Independent Music Releases	0.3015	0.7993
Gossip		
Adult and Celebrity Gossip Content	0.3214	0.7430
Lifestyle, Hobbies, and Recreation		
Crafting		
Handmade Craft Markets and Food Blogging	0.3073	0.7005
Crafting and Handmade Decor	0.2869	0.7299
Vehicles		
Automotive Reviews and Culture	0.3264	0.7793
Outdoors		
Nature, Parks, and Outdoor Recreation	0.3342	0.8028
Sports		
Team Sports Coverage	0.3051	0.8617
Beliefs, Identity, and Society		
Religion		
Christian Faith and Church Life	0.3404	0.7818
Identity		
Gender, Sexuality, and Relationship Essays	0.3198	0.6922
Knowledge and Records		
Genealogy		
Genealogy and British Nobility	0.3894	0.8416
Education		
Education and E-Learning	0.3255	0.7765
Landmarks		
Landmarks	0.6010	0.5018
Commerce		
Retail		
Retail Deals and Promotions	0.3172	0.7176

C Qualitative Analysis of Generation Under Capability Ablation

To complement the aggregate metrics, we show how **DEX**'s generations change as its experts are removed. For each of the four capabilities we take one representative prompt and decode the model's answer at increasing removal budgets k, under both selection rules (coverage and cumulative) and both variants (with and without renormalization). The answer does not simply break: with the responsible experts zeroed, the response is recomputed from the ones that remain and speaks from their perspective, drifting off the target capability as k grows. Two patterns are visible throughout. The renormalization choice is the dominant lever, with non-renorm losing on-topic fluency faster than renorm, while the two selection rules produce broadly similar text at matched k, so the trend is governed by k and the renorm choice rather than the ranking rule.

2pt

Table 3: Effect of ablating category-specialized experts on generation. For each category the experts most responsible for it are removed from the **DEX** mixture, under both selection rules (coverage and cumulative, §3.3) and both variants (with and without renormalization). k is the number of experts removed; k=0 is the intact model, shown once with each prompt. As k grows, the category's own experts are zeroed and on-topic fluency decays, with non-renorm degrading faster than renorm. Coverage and cumulative yield largely interchangeable generations at matched k; the qualitative trend is driven by k and by the renormalization choice, not by the ranking rule.

Sweep	k	**Continuation**
Biology. *Prompt:* `1. Introduction === *Mycoplasma pneumoniae* (MP) as one of the most frequent causes of community acquired pneumonia (CAP),^[[@R1]]^ accounts for up to 40` k=0 *(intact):* `% of all cases of pneumonia,^[[@R2]]^ and the most frequent cause of hospitalization (CAP) is the most frequent cause of hospitalization.` **[...]**		
Coverage, Non-Renorm	1	`years of hospital` **[...]**
	6	`years of hospital` **[...]**
	12	`years of hospital` **[...]**
	18	`years` **[...]**
	23	`years` **[...]**
Cumulative, Non-Renorm	1	`years of hospital` **[...]**
	6	`years of hospital` **[...]**
	12	`years of hospital` **[...]**
	18	`years` **[...]**
	23	`years` **[...]**

Sweep	k	Continuation
Coverage, Renorm	1	% of the total number of patients in the hospital.^[[@R2]]^ The most common cause of hospital admission is the presence of a hospital admission in the hospital.^[[@R3]]^**[...]**
	6	% of the total number of pneumonia cases in the United States. *Mycoplasma pneumoniae* (MP) is a common cause of community acquired pneumonia (CAP),^[[@R2]]^**[...]**
	12	% of the total number of pneumonia cases in the country. *Mycoplasma pneumoniae* (MP) as one of the most frequent causes of community acquired pneumonia (CAP),^[[@R2]]^ accounts for up to 40% of the total number of pneumonia cases in the**[...]**
	18	% of the population. *Mycoplasma pneumoniae* (MP) is a common form of pneumoniae,^[[@R2]]^[[@R3]]^[[@R4]]^[[@R5]]^
	23	% of the population. *Mycoplasma pneumoniae* (MP) as one of the most frequent causes of community acquired pneumonia (CAP),^[[@R2]]^ accounts for up to 40% of the population.**[...]**
Cumulative, Renorm	1	% of the total number of patients in the hospital.^[[@R2]]^ The most common cause of hospital admission is the presence of a hospital admission in the hospital.^[[@R3]]^**[...]**
	6	% of the total mortality of the hospital,^[[@R2]]^ and the hospital's mortality rate is approximately 0.5%**[...]**
	12	% of the total number of patients in the hospital. *Mycoplasma pneumoniae* (MP) is a hospital in the hospital, and is a hospital in the hospital.**[...]**
	18	% of the population. *Mycoplasma pneumoniae* (MP) is a family of the most frequent causes of community acquired pneumonia (CAP),^[[@R2]]^ accounts for up to 40% of the population.**[...]**
	23	% of the population. *Mycoplasma pneumoniae* (MP) is a family of the most frequent causes of community acquired pneumonia (CAP),^[[@R2]]^ accounts for up to 40% of the population.**[...]**
Cyber. *Prompt:* This invention pertains to bussing blocks; and more particularly to bussing blocks which can be readily integrated with a terminal block for a timing mechanism. Timing mechanisms are used to control the operation of appliances such as dryers		
k=0 *(intact):* , and the like. The present invention pertains to a bussing block which can be used in a bussing block**[...]**		
Coverage, Non-Renorm	1	, and the like. The present invention pertains to a bussing block which can be used in a bussing block**[...]**
	6	or dryers or dryers**[...]**
	12	or appliances such as dryers or appliances such as dryers**[...]**
	18	or appliances such as dryers or appliances such as such**[...]**
	23	or appliances such as dryers or appliances such**[...]**

Sweep	k	Continuation
Cumulative, Non-Renorm	1	, and the like. The present invention pertains to a bussing block which can be used in a bussing block **[...]**
	6	or dryers or dryers **[...]**
	12	or dryers or dryers **[...]**
	18	or dryers or dryers **[...]**
	23	or appliances such as dryers or appliances such **[...]**
Coverage, Renorm	1	, and the like. The present invention pertains to a bussing block which can be used in a bussing block **[...]**
	6	, and the use of a bussing block for a timing mechanism is a useful tool for determining the timing of the operation of a bussing block. The bussing block is a bussing block for a timing mechanism. **[...]**
	12	, and to reduce the use of the appliances to reduce the use of the appliances. The use of the appliances in the past is a very important part of the process. **[...]**
	18	and other appliances. The use of a bussing block is a very important step in the process of making a bussing block. **[...]**
	23	and other appliances. The use of the bussing block is a very important step in the development of a bussing block. **[...]**
Cumulative, Renorm	1	, and the like. The present invention pertains to a bussing block which can be used in a bussing block **[...]**
	6	, and the use of a bussing block for a timing mechanism is a useful tool for determining the timing of the operation of a bussing block. The purpose of this invention is to use a bussing block for a timing mechanism. **[...]**
	12	, and to reduce the number of blocks in the block. The purpose of this invention is to use a simple and simple method to control the operation of appliances such as dryers, and to reduce the number of blocks in the block. **[...]**
	18	, and other devices, such as the use of a timer or a timer. The invention pertains to bussing blocks; and more particularly to bussing blocks. **[...]**
	23	and other appliances. The use of a bussing block is a very important step in the development of a bussing block. **[...]**

Nuclear. *Prompt:* --- abstract: 'Mirror matter models have been suggested recently as an explanation of neutrino puzzles and microlensing anomalies. We show that mirror supernovae can be a copious source of energetic gamma rays if one assumes
k=0 *(intact):* a single-particle spin-orbit coupling. The mechanism of the spin-orbit coupling is a powerful tool for understanding the physics of the universe. **[...]**

Coverage, Non-Renorm	1	a one assumes a one assumes **[...]**
	6	a one assumes a one assumes **[...]**
	12	a one assumes a one assumes **[...]**
	18	a one assumes a one assumes **[...]**
	23	assumes **[...]**

Sweep	k	Continuation
Cumulative, Non-Renorm	1	`a one assumes a one assumes` **[...]**
	6	`a one assumes a one assumes` **[...]**
	12	`a one assumes a one assumes` **[...]**
	18	`assumes` **[...]**
	23	`assumes` **[...]**
Coverage, Renorm	1	`a single photon is emitted from the other. We show that mirror supernovae can be a copious source of energetic gamma rays if one assumes a single photon is emitted from the other.` **[...]**
	6	`a single particle is not a single particle. We show that mirror supernovae can be a copious source of energetic gamma rays if one assumes a single particle is not a single particle.` **[...]**
	12	`a single photon. We show that mirror supernovae can be a copious source of energetic gamma rays if one assumes a single photon.` **[...]**
	18	`a single gamma ray is a gamma ray. We show that mirror supernovae can be a copious source of energetic gamma rays if one assumes a single gamma ray is a gamma ray.` **[...]**
	23	`a single photon is a photon.' author: - |`
Cumulative, Renorm	1	`a single photon is emitted from the other. We show that mirror supernovae can be a copious source of energetic gamma rays if one assumes a single photon is emitted from the other.` **[...]**
	6	`a single particle is not a single particle. We show that mirror supernovae can be a copious source of energetic gamma rays if one assumes a single particle is not a single particle.` **[...]**
	12	`a single gamma ray. We show that mirror supernovae can be a copious source of energetic gamma rays if one assumes a single gamma ray.` **[...]**
	18	`a single gamma ray. We show that mirror supernovae can be a copious source of energetic gamma rays if one assumes a single gamma ray.` **[...]**
	23	`a single gamma ray. We show that the gamma rays are not the same as the gamma rays. We show that the gamma rays are the same as the gamma rays.` **[...]**

Code. *Prompt:* `/* * Initialize built-in objects. Current thread must have a valstack * and initialization errors may longjmp, so a setjmp() catch point * must exist. */`
k=0 *(intact):* `#include <stdio.h> #include <stdio.h>` **[...]**

Coverage, Non-Renorm	1	`public static void Main() {`
	6	`public static void Main() { public static void Main() {` **[...]**
	12	*(empty)*
	18	*(empty)*
	23	*(empty)*
Cumulative, Non-Renorm	1	`public static void Main() {`
	6	`public static void Main() throws Exception { }` **[...]**
	12	*(empty)*
	18	*(empty)*
	23	*(empty)*

Sweep	k	Continuation
Coverage, Renorm	1	`public void setjmp(String[] args) {` `System.out.println("\n");` **[...]**
	6	`public void Initialize() { // TODO` `Auto-generated method stub` **[...]**
	12	`#` **[...]**
	18	`#` **[...]**
	23	`#define` **[...]**
Cumulative, Renorm	1	`public void setjmp(String[] args) {` `System.out.println("\n");` **[...]**
	6	`public void Initialize() { // TODO` `Auto-generated method stub` **[...]**
	12	`#` **[...]**
	18	`/* The first thing I have tried is to use the same thing as the first thing. I have tried the same thing as the first thing.` **[...]**
	23	`The first object is a frame of the frame. The frame is a frame of the frame.` **[...]**