LEARNING WHAT TO PRESERVE: GRAMMAR CONSTRAINED SEARCH ACROSS SCIENTIFIC DOMAINS

Anonymous authors
Paper under double-blind review

ABSTRACT

Scientific strings encode structure directly in their notation. A chemical formula such as $\mathrm{LiMn_2O_4}$ specifies elemental identities and stoichiometric quantities, while a protein mutation such as `M105K` specifies residues and a positional coordinate. Yet pretrained models typically process such inputs using tokenizers learned from corpus statistics rather than from the structure of the notation itself. Existing solutions often introduce domain-specific parsers, vocabularies, or tokenizers, requiring a new representation strategy for each scientific domain. We ask whether a shared input representation rule can instead be discovered from downstream tasks across domains with different structural requirements. We introduce GCMS, a grammar-constrained multi-domain search framework that uses a policy gradient search over interpretable input rewrites. Joint search on crystal-system classification and protein mutation fitness discovers a common representation without an element inventory, residue classifier, or domain-specific parser. The learned rule transfers across unseen tasks and notation systems, including held-out materials properties, additional protein assays, modified peptides, and genomic variants. GCMS reduces band-gap MAE from 0.6161 to 0.4918 relative to WordPiece and reaches 0.8789 Spearman on the held-out ProteinGym BLAT assay. These results show that transferable structure in scientific notation can be discovered from task supervision rather than engineered separately for each domain.

1 INTRODUCTION

Scientific strings carry structure in their notation. In materials science, a formula such as $\mathrm{LiMn_2O_4}$ encodes elements and stoichiometric quantities. In the life sciences, a mutation such as `M105K` encodes the original residue, its position, and the substituted residue. Standard subword tokenizers do not use this structure when deciding where to cut. Their segmentation is largely driven by corpus frequency. As a result, a tokenizer can merge different elements into one piece or split a multi-digit mutation position into separate digits (Figure 1). These errors change how the scientific object is represented to the downstream model, because the resulting pieces become the units from which the model must recover its structure.

Existing work addresses this problem mainly within individual domains. Materials tokenizers add domain-specific vocabulary or use curated chemical information (Oh et al., 2025; Gupta et al., 2022; Trewartha et al., 2022), while vocabulary refinement methods modify the tokenizer itself (Chizhov et al., 2024). These approaches can work well within the notation for which they were designed, but they do not transfer naturally across scientific domains. An element inventory, for example, provides no information about mutation notation and cannot capture all formula patterns, such as fractional subscripts in `La0.7Sr0.3MnO3`. Other approaches remove the subword vocabulary and operate directly on characters or bytes (Clark et al., 2022; Xue et al., 2022), or learn segmentation within the model (Tay et al., 2021), but these require changes to the encoder architecture or pretraining setup. We instead ask whether a common input representation rule can be learned from downstream task performance and reused across distinct scientific notations, while retaining the vocabulary and pretrained model already in place.

The challenge becomes clearer when materials and life-science notation are considered together. The two domains impose different requirements on how their structure should be represented. Chemical formulas often benefit from finer segmentation, while protein mutation strings require multi-digit

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.

positions to remain intact. A representation optimized for one scientific notation can perform poorly on another. Learning a separate rule for each notation avoids this conflict, but still requires a new representation strategy whenever the notation changes. The question is whether we can learn a representation that preserves useful structural primitives in scientific notation, thereby enabling it to generalize across different scientific domains.

We address this problem with Grammar-Constrained Multi-Domain Search (GCMS), which learns how structured scientific inputs should be transformed before being passed to an existing pretrained model. GCMS searches an explicit input rewrite over a closed grammar of detectors, cut policies, digit-attachment modes, and exception handlers (Figure 1). It requires no trained structure detector, element recognizer, residue classifier, or domain-specific parser. A small autoregressive policy assigns credit to grammar decisions using a group-relative advantage estimator adapted from GRPO (Shao et al., 2024; Guo et al., 2025). We use this estimator without the PPO surrogate, consistent with recent analyses of reward normalization in GRPO (Liu et al., 2025). GCMS follows the idea of invalid-action masking (Huang & Ontañón, 2022), but also handles choices that are valid yet have no effect under the current rule, which may still matter on unseen domains. The selected rewrite is applied before the existing WordPiece tokenizer. GCMS keeps the pretrained encoder initialization and vocabulary fixed, changing only the input transformation. Importantly, the search policy is domain-unconditioned. It receives neither domain identity nor domain-specific features, and proposes a single rewrite rule that is evaluated across all search domains.

Learning one representation across scientific domains requires a reward that can compare tasks with different metrics and different levels of evaluation noise. GCMS measures each rule relative to the identity rewrite and normalizes the improvement by the seed-to-seed standard deviation of that domain. The normalized improvements are then combined with a soft minimum, so a large gain on one domain cannot compensate for failure on another while improvements on non-binding domains still affect the reward. This is important for crystal-system classification and ProteinGym mutation fitness, which impose conflicting requirements on the representation. The objective is related to worst-case optimization (Namkoong & Duchi, 2016; Sagawa et al., 2019) and multi-task learning (Yu et al., 2020; Chen et al., 2018; Kendall et al., 2018; Sener & Koltun, 2018); related multi-task objectives have also been studied in GRPO (Ramesh et al., 2026).

Our contributions are threefold.

- **RL-guided grammar-constrained search.** We use a lightweight autoregressive policy with group-relative policy gradients to search explicit, interpretable rewrite rules from downstream scientific feedback.
- **Noise-normalized multi-domain optimization.** We combine task improvements on an empirical noise scale with a soft-min objective to search across domains with conflicting segmentation requirements.
- **Cross-domain scientific transfer.** We show that rules learned from materials and protein mutation tasks transfer to held-out materials properties, additional protein assays, modified peptides, and genomic variant notation.

2 Problem Formulation

Given structured scientific inputs from multiple domains, our goal is to learn a common input representation that improves downstream prediction. We search for this representation over a finite grammar of explicit rewrites and use downstream scientific task performance as supervision. Let $\mathcal{G}$ denote the rewrite grammar and let a rule $r \in \mathcal{G}$ be specified by a sequence of discrete decisions $\tau = (a_1, \ldots, a_{|\tau|})$. We cast rule construction as a finite-horizon Markov decision process

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \delta, R, H), \qquad |\tau| \leq H. \tag{1}$$

At step t, the state

$$s_t = (a_1, \ldots, a_{t-1}) \tag{2}$$

is the partially specified rule, and the action

$$a_t \in \mathcal{A}(s_t) \tag{3}$$

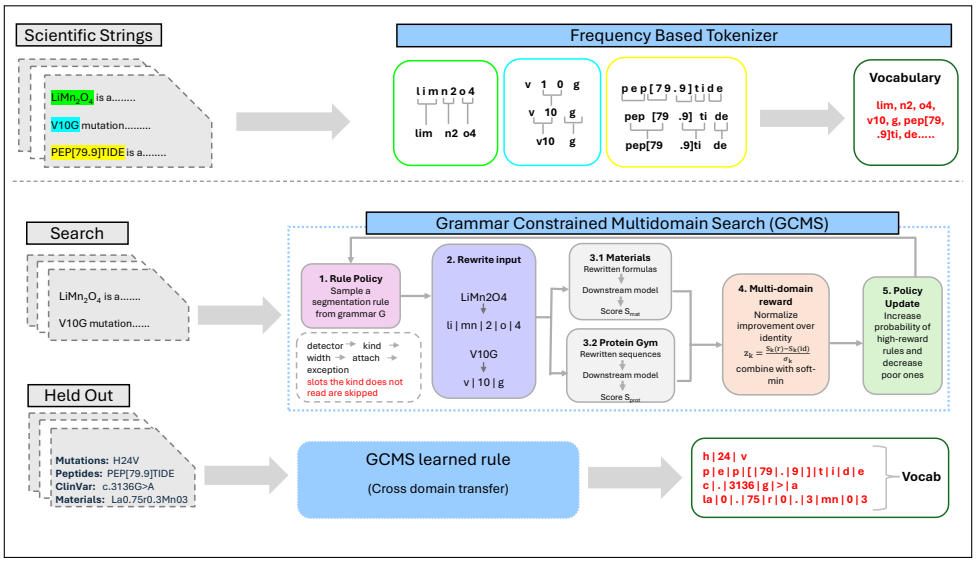


Figure 1: Overview of GCMS. A policy samples an explicit rewrite rule from a constrained grammar, deciding only the slots the chosen segmentation kind reads (inset). The rule is scored on every domain and the improvements are combined by a noise-normalized soft minimum.

selects the next valid grammar option, such as the detector, segmentation policy, digit-handling mode, or exception rule. The transition is deterministic,

$$s_{t+1} = s_t \oplus a_t, \tag{4}$$

written $\delta(s_t, a_t)$, and the grammar restricts $\mathcal{A}(s_t)$ to decisions that are relevant given the preceding choices. A completed trajectory deterministically defines a rewrite rule,

$$r = g(\tau). \tag{5}$$

There is no informative reward for an incomplete rule. Thus

$$R_t = 0, \qquad t < |\tau|, \tag{6}$$

and the reward arrives only at $t = T$, after r is applied to the inputs and the resulting downstream models are evaluated on K domains. Section 3.1 defines how the per-domain scores are put on a common scale and Section 3.2 how they are combined into the scalar $R(r)$. The search objective is

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[R(g(\tau))\right], \tag{7}$$

where $\pi_\theta(a_t \mid s_t)$ is an autoregressive policy over grammar decisions.

Two features of $\mathcal{M}$ decide how J can be optimized. Transitions are deterministic and the reward is terminal, so the return of a trajectory is the single scalar $R(g(\tau))$. Differentiating J gives

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\Big[\, R(g(\tau)) \, \textstyle\sum_{t=1}^{|\tau|} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\Big], \tag{8}$$

which is the REINFORCE estimator (Williams, 1992). R is not differentiable in θ, because it is obtained by fine-tuning a separate encoder, so this score-function form is the only route available.

Subtracting a baseline b from R leaves the estimator unbiased when b is independent of the action being scored, since $\mathbb{E}_\tau[\, b\, \nabla_\theta \log \pi_\theta(\tau)] = 0$ holds only under that condition. A terminal reward does not rule out a learned critic, which could predict the expected return of a partially specified rule even though the intermediate rewards are zero. We do not use one for a practical reason. Every reward costs a full fine-tuning run, so a search of 120 evaluations observes 120 returns in total, which is far too few to fit a value function over the slot space. GCMS instead takes b from the rules scored alongside the sample in the same group (Section 3). That baseline is the group mean, which includes the sampled rule itself, so it is not strictly independent of the action and the resulting gradient is biased. The dependence is of order $1/G$, and the uniform reserve reduces it further, since those rules are drawn from the grammar rather than from the policy.

3 THE GCMS ALGORITHM

GCMS searches a finite grammar of deterministic input rewrites. A rule is written as

$$r = (d, \ell_{\min}, \ell_{\max}, c, \phi, \eta, e), \tag{9}$$

where d is the detector, $(\ell_{\min}, \ell_{\max})$ are its length bounds, c is the segmentation family, ϕ contains family-specific parameters, η controls digit handling, and e is the exception policy. Appendix A.1 gives the complete grammar. The blind grammar contains no element symbols, residue inventory, or task-specific lexicon.

A 31.5k-parameter recurrent policy generates the rule sequentially,

$$\pi_\theta(r) = \prod_{t=1}^{|\tau|} \pi_\theta(a_t \mid a_{<t}). \tag{10}$$

The policy is domain-unconditioned, domain identity and domain-specific features are not inputs to π_θ. Later decisions therefore condition on earlier grammar choices. Slots that are not used by the selected segmentation family are skipped. This is important because the effect of a choice such as digit handling depends on the preceding segmentation policy.

Each sampled rule rewrites the input before tokenization,

$$x \xrightarrow{r} \tilde{x} \xrightarrow{\text{WordPiece}} \text{encoder}. \tag{11}$$

For every candidate, we start from the same pretrained checkpoint and vocabulary and use the same downstream fine-tuning protocol. The only difference is the rewritten input passed to the tokenizer.

Different grammar trajectories can produce the same rewrite behavior. GCMS therefore compares candidates on a fixed probe set $\mathcal{P}$ and treats

$$r_i \equiv r_j \iff r_i(x) = r_j(x) \quad \forall x \in \mathcal{P} \tag{12}$$

as behaviorally equivalent. Behavioral equivalence is used to keep a sampled group distinct: a draw whose behavior already appears in the group, or matches a barred baseline, is rejected and resampled. Evaluations are cached separately, keyed on the rule specification itself, so a rule already scored anywhere in the search is never rescored. The budget therefore counts distinct rule specifications, and two specifications that happen to segment the probe set identically are charged twice if they arise in different groups.

At each iteration, GCMS evaluates a group $\mathcal{B}$ of G unique rules on every search domain. Their task scores are converted to the scalar reward defined in Sections 3.1 and 3.2. The group has two parts. The policy draws in $\mathcal{B}_\pi$ carry a log-probability, while a uniform reserve drawn from the grammar does not. For rewards $\{R_i\}_{i \in \mathcal{B}}$, the group-relative advantage is

$$A_i = \frac{R_i - \bar{R}_\mathcal{B}}{\max(s_\mathcal{B}, \epsilon)}, \qquad i \in \mathcal{B}, \tag{13}$$

where $\bar{R}_\mathcal{B}$ and $s_\mathcal{B}$ are the mean and standard deviation over all G rules, reserve included. Only the policy draws enter the update,

$$\mathcal{L}_{\text{PG}} = -\frac{1}{|\mathcal{B}_\pi|} \sum_{i \in \mathcal{B}_\pi} \Big[A_i \sum_{t=1}^{T_i} \log \pi_\theta(a_{i,t} \mid a_{i,<t}) + \lambda_H \mathcal{H}_i \Big], \tag{14}$$

where $T_i = |\tau_i|$ counts only the slots rollout i actually sampled and $\mathcal{H}_i$ is the policy entropy at those decisions. The reserve therefore shapes the baseline without contributing gradient. This is group-relative REINFORCE. It uses the advantage normalization of GRPO, but each rollout is used once in a fresh on-policy update. We therefore do not use PPO importance ratios, clipping, or a reference-policy KL term. Algorithm 1 gives the search loop for one search seed in full.

Algorithm 1 GCMS (Grammar-Constrained Multi-Domain Search)

Require: domains $\{D_k\}_{k=1}^{K}$, grammar $\mathcal{G}$ with slots $1..L$, policy π_θ, evaluation budget B, group size G, probe set P, soft-min temperature β, advantage floor ϵ, uniform-reserve fraction ν

Calibration (once): for each k, evaluate the identity rule and record $S_k(\mathrm{id})$ and its seed noise σ_k
Barred behaviors: $\mathcal{H}_0 \leftarrow \{\mathrm{beh}(\mathrm{identity}, P),\ \mathrm{beh}(\mathrm{charsplit}, P)\}$
Adaptive group size: $G \leftarrow \min\big(G,\ |\{\mathrm{beh}(r, P) : r \sim \mathcal{G}\} \setminus \mathcal{H}_0|\big)$
Evaluation cache: $\mathcal{C} \leftarrow \emptyset$
while budget spent $< B$ **do**
  $\mathcal{B}_{\mathrm{unif}} \leftarrow \lceil \nu G \rceil$ uniform-grammar
  $\mathcal{B}_\pi \leftarrow \emptyset$; $\mathcal{B} \leftarrow \mathcal{B}_{\mathrm{unif}}$
  while $|\mathcal{B}| < G$ **do**
    $\rho \leftarrow \emptyset$; $\tau \leftarrow ()$
    for $i = 1$ to L **do**
      if slot i is read by the segmentation kind already chosen **then**
        $\rho_i \sim \pi_\theta(\cdot \mid \rho_{<i})$; $\tau \leftarrow \tau \,\|\, \rho_i$
      else
        $\rho_i \leftarrow$ canonical value
    if $\mathrm{beh}(\rho, P) \notin \mathcal{H}_0 \cup \mathrm{beh}(\mathcal{B}, P)$ **then**
      $\mathcal{B}_\pi \leftarrow \mathcal{B}_\pi \cup \{(\rho, \tau)\}$; $\mathcal{B} \leftarrow \mathcal{B} \cup \{\rho\}$
  for $r \in \mathcal{B}$ **do**
    if $\mathrm{spec}(r) \notin \mathcal{C}$ **then**
      $\mathcal{C}[\mathrm{spec}(r)] \leftarrow$ SoftMinReward$(r, \{D_k\}, \{S_k(\mathrm{id})\}, \{\sigma_k\}, \beta)$
    $R(r) \leftarrow \mathcal{C}[\mathrm{spec}(r)]$
  if $\mathrm{sd}_{\mathcal{B}}(R) < \epsilon_{\mathrm{var}}$ **or** $|\mathcal{B}_\pi| < 2$ **then**
    inject fresh uniform behaviors; raise entropy coefficient; **no update**
  else
    $A_i \leftarrow \big(R_i - \bar{R}_{\mathcal{B}}\big) / \max\big(\mathrm{sd}_{\mathcal{B}}(R), \epsilon\big),\ i \in \mathcal{B}$
    $\theta \leftarrow$ Optimizer$\big(\theta,\ \nabla_\theta \sum_{i \in \mathcal{B}_\pi} [-A_i \log \pi_\theta(\tau^{(i)}) - \lambda \mathcal{H}(\pi_\theta)]\big)$, $\log \pi_\theta(\tau) = \sum_t \log \pi_\theta(a_t \mid s_t)$
return rule maximizing R

3.1 Multi-domain scoring

GCMS scores rules across tasks with different metrics by expressing improvement over identity in units of each task's evaluation variability and combining these gains with a soft minimum. This prevents improvement on one domain from masking failure on another while retaining information from both domains. Let S_k be the higher-is-better score on domain k, negative cross-entropy for classification and Spearman for regression. Let σ_k be its seed-to-seed standard deviation, measured once on the identity rule. The per-domain signal is

$$z_k(r) \;=\; \frac{S_k(r) - S_k(\mathrm{identity})}{\sigma_k}, \tag{15}$$

the improvement over doing nothing in units of that domain's evaluation noise. Note that σ_k measures evaluation variability only. It says nothing about how far apart the domain places different rules. Minimum-like objectives over noisy estimates are biased downward, with the bias growing in σ, so every domain and every rule is evaluated with the same number of seeds.

3.2 Soft-Minimum Formulation

Both obvious ways to combine the z_k fail, in opposite directions (Table 7, Appendix B). A *mean* tolerates failure on one domain when the other compensates. Character splitting is $+9.53\sigma$ on crystal and -3.58σ on ProteinGym, so its mean is positive and it outranks rules that are merely mediocre everywhere.

A *hard minimum* rejects character splitting but goes blind once the binding domain saturates. ProteinGym admits only four reachable behaviors, and every good rule realizes the same one. Three structurally different rules therefore score exactly $+2.73$ there while differing by nearly $3\times$ on crystal,

Algorithm 2 SOFTMINREWARD, one scalar from K domains on different scales

Require: rule r, domains $\{D_k\}$, identity references $\{S_k(\mathrm{id})\}$, per-domain noise $\{\sigma_k\}$, temperature β

for all $k = 1..K$ **do**
  $S_k(r) \leftarrow$ score of r on D_k, higher-is-better ▷ $-$cross-entropy, or Spearman
  $z_k \leftarrow \big(S_k(r) - S_k(\mathrm{id})\big)/\sigma_k$ ▷ noise-normalized improvement
if $\beta = 0$ **then**
  return $\min_k z_k$ ▷ hard minimum; saturates when one domain does
return $-\frac{1}{\beta}\log\sum_k \exp(-\beta z_k)$

All K domains use the same seed count: min*-like objectives on noisy estimates are biased downward with bias growing in σ, so unequal budgets would penalize a rule for having a noisy domain rather than a bad one.*

$+9.91$ against $+3.47$. The minimum discards that difference. Run under a hard minimum, GCMS returns the worst of the three on the domain being ignored.

SEC-style adaptive reweighting (Ramesh et al., 2026) prioritizes the tasks currently supplying the largest advantages. It fails for the same underlying reason, in the opposite direction. ProteinGym's small spread across a group reads as an absence of learning, so SEC downweights it, with weights drifting to 0.59 on crystal against 0.41 on ProteinGym. The hard minimum ignores crystal because ProteinGym binds. SEC ignores ProteinGym because ProteinGym saturates.

GCMS uses a soft minimum,

$$R(r) \;=\; -\frac{1}{\beta}\log\sum_k \exp\big(-\beta\, z_k(r)\big), \qquad \beta = 1, \tag{16}$$

which keeps the failure penalty, still scoring character splitting at -3.58, while separating the three rules a hard minimum ties. Every domain affects the reward in every update. As β grows the soft minimum tends to the hard one, so β interpolates between the two failure modes.

4 EXPERIMENTAL SETUP

GCMS searches on crystal-system classification and ProteinGym PSEAI mutation fitness. They are chosen because chemical formulas and mutation strings impose conflicting segmentation constraints. Each search task is divided into *fit*, *evolve*, *select*, and *test* splits using an approximately 48/16/16/20 ratio. GCMS is run with five search seeds and three evaluation seeds. Each run uses *fit* for model fitting and *evolve* for search reward. The best rule from each run is then evaluated on the independent *select* split, and the rule with the highest multi-domain select reward is chosen as the final rule. The selected rule is fixed before evaluation on any transfer task or test set. Final benchmark results are averaged over three evaluation seeds. All remaining datasets are excluded from both search and final rule selection. Transfer is evaluated on is-metal, glass, band gap, dielectric, $\log G_{\mathrm{VRH}}$, and $\log K_{\mathrm{VRH}}$ to test generalization across materials targets under formula notation. Additional ProteinGym assays test transfer across protein fitness landscapes, and modified peptides and ClinVar cDNA variants test transfer to unseen scientific notations. These tasks cover recurring scientific primitives including elemental composition, stoichiometric quantities, residue identities, positional coordinates, sequence modifications, and genomic variants. Materials splits are formula-disjoint and ClinVar splits string-disjoint. We compare GCMS with WordPiece, BPE, PickyBPE (Chizhov et al., 2024), and Unigram tokenization, as well as MATTER (Oh et al., 2025). The tokenizer baselines use their corresponding pretrained encoders. Implementation and optimization details are given in the Appendix B.

5 RESULTS

Table 1 evaluates whether the representation learned from crystal-system classification and ProteinGym transfers to scientific tasks that were never used during search. The strongest transfer is to band-gap prediction, for which GCMS never observes a band-gap label during search. Learned tokenizer baselines obtain between 0.5230 and 0.6161 MAE, while GCMS reaches 0.4918 without

Table 1: Materials and life-science tasks. Accuracy is reported for materials classification tasks, MAE for materials regression tasks, and Spearman ρ for life-science tasks. Values are mean $\pm$ standard deviation over three evaluation seeds. Crystal and ProteinGym PSEAI are included in the search reward while all other tasks are held out. For peptide iRT, the GCMS length bound is lifted as described in the text. ↑ indicates higher is better and ↓ indicates lower is better.

task	MATTER	WordPiece	BPE	PickyBPE	Unigram	**GCMS**
Materials						
crystal * ↑	0.5804 ± 0.0047	0.5820 ± 0.0051	0.5934 ± 0.0032	0.5942 ± 0.0008	0.6063 ± 0.0018	$\mathbf{0.6147} \pm 0.0017$
is-metal ↑	0.8537 ± 0.0020	0.8526 ± 0.0027	0.8621 ± 0.0047	0.8621 ± 0.0056	0.8784 ± 0.0026	$\mathbf{0.8913} \pm 0.0081$
glass ↑	0.7958 ± 0.0054	0.8028 ± 0.0087	0.8005 ± 0.0033	0.8016 ± 0.0050	0.8262 ± 0.0140	$\mathbf{0.8269} \pm 0.0075$
band gap ↓	0.6144 ± 0.0020	0.6161 ± 0.0220	0.5405 ± 0.0023	0.5574 ± 0.0089	0.5230 ± 0.0287	$\mathbf{0.4918} \pm 0.0044$
dielectric ↓	0.5546 ± 0.0027	0.5509 ± 0.0045	0.5429 ± 0.0046	0.5367 ± 0.0038	0.5027 ± 0.0066	$\mathbf{0.4998} \pm 0.0028$
$\log G_{\mathrm{VRH}}$ ↓	0.1810 ± 0.0014	0.1801 ± 0.0017	0.1676 ± 0.0010	0.1692 ± 0.0009	0.1552 ± 0.0007	$\mathbf{0.1500} \pm 0.0009$
$\log K_{\mathrm{VRH}}$ ↓	0.1597 ± 0.0003	0.1617 ± 0.0016	0.1434 ± 0.0021	0.1496 ± 0.0024	$\mathbf{0.1349} \pm 0.0009$	0.1358 ± 0.0022
Life sciences						
peptide iRT ↑	0.8919 ± 0.0037	0.8907 ± 0.0100	0.8745 ± 0.0062	0.8936 ± 0.0084	0.9459 ± 0.0017	$\mathbf{0.9649} \pm 0.0007$
ProteinGym PSEAI * ↑	0.8533 ± 0.0031	0.8566 ± 0.0020	0.8283 ± 0.0033	0.8385 ± 0.0058	0.8728 ± 0.0019	$\mathbf{0.8754} \pm 0.0019$
ProteinGym BLAT↑	0.8565 ± 0.0032	0.8657 ± 0.0024	0.8471 ± 0.0010	0.8549 ± 0.0087	0.8791 ± 0.0006	$\mathbf{0.8789} \pm 0.0019$
ProteinGym BRCA1 ↑	0.5723 ± 0.0104	0.6092 ± 0.0114	0.5657 ± 0.0266	0.5084 ± 0.0133	$\mathbf{0.6344} \pm 0.0078$	0.5874 ± 0.0019
ProteinGym INFA ↑	0.2447 ± 0.0024	0.2308 ± 0.0028	0.2726 ± 0.0134	0.2449 ± 0.0246	0.3015 ± 0.0436	$\mathbf{0.3158} \pm 0.0019$
ClinVar cDNA ↑	0.8210 ± 0.0066	0.8183 ± 0.0092	0.7705 ± 0.0053	0.7882 ± 0.0183	0.8045 ± 0.0035	$\mathbf{0.8221} \pm 0.0050$

retraining the tokenizer, reducing error by 20% relative to WordPiece. The learned representation also transfers to dielectric, $\log G_{\mathrm{VRH}}$ and $\log K_{\mathrm{VRH}}$ with error reductions of 9.3%, 16.7%, and 16.0% respectively relative to WordPiece. GCMS also transfers across formula-based classification tasks, reaching 0.8913 on held-out is-metal compared with 0.8526 for WordPiece, while remaining close to the strongest learned-tokenizer baseline on glass. GCMS receives no element list, yet outperforms MATTER across all seven materials tasks. It reaches 0.6147 versus 0.5804 on crystal-system classification and 0.8913 versus 0.8537 on held-out is-metal classification, while reducing band-gap MAE from 0.6144 to 0.4918. This suggests that useful structure in chemical-formula inputs can be learned from downstream scientific supervision rather than supplied through a curated element-aware vocabulary.

ProteinGym provides the biological domain used during search and imposes a different structural requirement from chemical formulas. In mutation strings such as `V10G`, the numeric span represents a residue position and must remain intact. GCMS learns this behavior without being given a residue alphabet or an explicit notion of position, reaching 0.8754 Spearman on PSEAI. The representation also transfers across additional ProteinGym assays. Across the four assays in Table 1, GCMS outperforms BPE and PickyBPE on all four and WordPiece on three, with mean gains of 0.036, 0.053, and 0.024, respectively. This shows that the learned representation is not specific to the particular protein assay used during search.

More importantly, the representation transfers to scientific notations that are never used during optimization. On ClinVar coding-DNA variants, GCMS reaches 0.8221 accuracy and improves over three of the four learned tokenizers, despite never receiving a genomic parser or variant-specific vocabulary. The gains range from 0.018 to 0.052 accuracy over those baselines. This suggests that the learned rule captures local structural regularities that extend beyond protein mutation notation, particularly the separation of symbolic units while preserving numeric coordinates.

Modified peptides provide a further test because their notation differs from both formulas and single-site mutations. With the learned detector bound unchanged, GCMS reaches 0.9304 Spearman because only 63% of peptide inputs are rewritten. Relaxing that bound while keeping the segmentation logic fixed raises performance to 0.9649, close to the hand-built PTM lexicon at 0.9660. Peptides are never included in the search reward, and GCMS is not given a modification table. The result therefore separates two aspects of transfer: the learned segmentation behavior generalizes well, while the detector range is more domain dependent.

6 Ablations

We use ablations to isolate the contribution of the main design choices in GCMS. Firstly, to isolate the effect of cross-domain supervision, we repeat the search within materials (crystal + is-metal) and within life sciences (PSEAI + BLAT). The within-domain searches produce more specialized

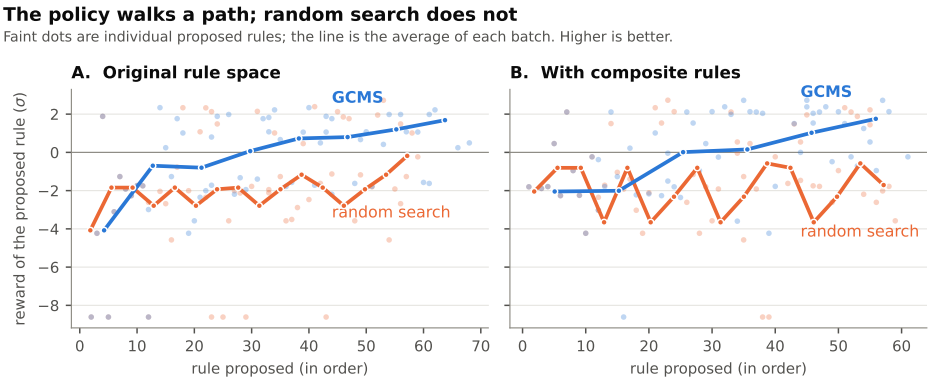


Figure 2: What the policy contributes. Each dot is one proposed rule and each line a group mean, for the flat (left) and composite (right) slot lists. The policy's mean climbs while random search stays flat.

Table 2: Sealed consequence of the objective, with policy, grammar, budget and seed held fixed. The two alternatives fail in opposite directions, both because ProteinGym admits few distinguishable behaviors.

objective	crystal	ProteinGym	failure mode
soft-min (β=1)	**0.6147**	**0.8754**	—
SEC-style reweighting	0.6077	0.8612	downweights the saturated domain
hard minimum	ties three rules at $+2.73\sigma$		blind to the non-binding domain
mean	selects a rule at -3.58σ		tolerates failing a domain

rules, while the cross-domain search gives the most transferable rule (Appendix Table 8). Second, the aggregation function controls how the search balances the two domains. A mean selects character splitting despite a -3.58σ drop on ProteinGym, while a hard minimum assigns the same $+2.73\sigma$ ProteinGym score to three structurally different rules. SEC-style weighting performs worse than soft-min by 0.0070 on crystal and 0.0142 on ProteinGym. Sweeping $\beta \in 0.25, 0.5, 1, 2$ shows the expected shift from mean-like compensation at low β to hard-min saturation at high β. The learned policy also increases the group mean reward during search, while matched random search remains approximately flat (Fig. 2). Also, expanding the grammar with case-aware operations improves formula-specific rules but weakens cross-notation transfer.

Thirdly, we repeated GCMS with crystal+BLAT and is-metal+BLAT as search domains to determine which parts of the learned representation are specific to the headline search pair. Across all search pairs and seeds, GCMS preserves multi-digit spans, identifying numeric quantities and coordinates as a stable structural primitive. Allowing identity and plain character splitting still returns digit-preserving rules at all three seeds. Using the same grammar and evaluation budget, greedy search, beam search, and Monte Carlo tree search (MCTS) transfer less well than GCMS to held-out tasks (Appendix C.4). Further design ablations are in Appendix C.

Finally, a pretrained language model encodes chemical and biological knowledge, which makes it a natural alternative to a learned policy for proposing rules. We substitute Qwen2.5-7B (Bai et al., 2023) for the policy, holding the grammar, budget, group size and objective fixed. It produces the strongest formula rules we measured but fails the multi-domain objective. Across three search seeds none of its 112 proposals segments `V10G` into `V 10 G`, the parse the mutation assay depends on. Appendix C.3 gives the comparison in full.

7 Discussion

GCMS searches over functional input behaviors rather than only over rule syntax. This distinction matters for scientific strings because different grammar choices can produce the same rewrite on the data being searched. For example, formulas such as `LiMn2O4` admit many useful boundary patterns, whereas mutation strings such as `V10G` admit far fewer distinct behaviors. GCMS therefore forms groups from behaviorally distinct rewrites and adapts the group size to the number of behaviors that are actually reachable. This prevents repeated versions of the same transformation from dominating the group-relative credit signal.

The results suggest that cross-domain supervision favors structural choices that transfer across scientific notations. Fine segmentation is beneficial for `LiMn2O4`, but applying the same operation blindly to `V10G` produces `V 1 0 G` and destroys the residue position. The within-domain searches recover complementary structure. The materials-only rule favors fine letter separation but fragments numeric spans (`A123G` → `A 1 2 3 G`), while the life-sciences-only rule preserves the numeric span but leaves adjacent symbols merged (`A123G` → `A 123G`). GCMS combines both behaviors, producing `A 123 G`. This behavior is not specific to one search pair. Across crystal+PSEAI, crystal+BLAT, and is-metal+BLAT, multi-digit spans remain intact even when other boundaries change. Numeric-span preservation emerges as a stable local primitive during the search, while finer segmentation choices remain task dependent.

The policy and the grammar play different technical roles. Expanding the grammar with case-aware operations produces stronger formula-specific rules, showing that the grammar determines which scientific structures are reachable. The learned policy instead determines where the limited evaluation budget is spent within that space. The evidence suggests that reinforcement learning can concentrate expensive downstream evaluations on rewrite rules with higher observed reward.

Behavioral equivalence is deliberately local to the probe set rather than treated as global equivalence. Two rules that rewrite the probe strings identically are not compared within the same group, but the evaluation cache remains keyed by the full rule specification. This is important for transfer because rules that agree on search examples such as `LiMn2O4` and `V10G` may behave differently on unseen forms such as modified peptides or ClinVar variants. GCMS therefore uses probe behavior to reduce redundant search without assuming that agreement on the search domains implies agreement on new scientific notations.

8 Limitations

GCMS searches within a predefined grammar of input rewrites, so it can only recover structure that is expressible by that grammar. The learned rules are selected for predictive utility and should not be interpreted as discovering the unique or scientifically correct decomposition of a notation. Our search uses two scientific domains and is evaluated on structured string representations, so broader validation is needed across other scientific fields, notation systems, and data modalities. GCMS also requires downstream model training to score candidate rules, which makes search more expensive as models and datasets grow. Finally, the method learns from task supervision, so the structure it recovers may depend on the available labels and objectives. Extending the framework to larger scientific foundation models, richer grammars, and more diverse domains is therefore an important direction for future work.

9 Conclusion

Scientific models often receive structured inputs through tokenizers that were trained from corpus statistics rather than from the structure of the scientific notation itself. GCMS treats this as a learning problem. Instead of supplying an element parser, residue classifier, trained structure detector, or domain-specific tokenizer, GCMS searches an explicit grammar of input rewrites using downstream scientific performance. A lightweight autoregressive policy assigns credit to grammar decisions, while noise-normalized improvements and a soft-min objective allow the same search to learn from domains with conflicting representational requirements.

The resulting rule is simple and interpretable and most importantly, this behavior is learned rather than specified. Searching on crystal-system classification and protein mutation fitness produces a representation that transfers to tasks never used in optimization. Multi-domain supervision retains strong within-domain generalization while improving cross-domain transfer. GCMS reduces held-out band-gap MAE from 0.6161 to 0.4918 relative to WordPiece. The same segmentation logic transfers to additional protein assays and modified peptides, and remains competitive on genomic variant notation. On materials tasks it also outperforms MATTER, which explicitly incorporates chemical element information into its vocabulary. More broadly, GCMS shows that useful scientific input structure can be learned directly from task supervision through interpretable representations that transfer across domains, rather than being specified through hand-designed parsers, curated vocabularies, or separate detectors.

10 Reproducibility Statement

The GCMS algorithm and experimental protocol are described in Sections 3 and 4. Appendix A provides the full rewrite grammar and dataset splits, while Appendix B reports optimization details and search hyperparameters. An anonymized code is provided in the supplementary material.

11 AI use statement

Generative AI tools were used only to review and polish the manuscript, while all scientific content, analyses, experiments, and conclusions were developed and verified by the authors.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of ICML*, 2018.

P. Chizhov, C. Arnett, E. Korotkova, and I. P. Yamshchikov. Pickybpe: Efficient vocabulary refinement during tokenizer training. In *Proceedings of EMNLP*, 2024.

Jonathan H Clark, Dan Garrette, Iulia Turc, and John Wieting. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91, 2022.

D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

T. Gupta, M. Zaki, N. M. A. Krishnan, and Mausam. Matscibert: A materials domain language model for text mining and information extraction. *npj Computational Materials*, 8(102), 2022.

S. Huang and S. Ontañón. A closer look at invalid action masking in policy gradient algorithms. In *FLAIRS*, 2022.

A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of CVPR*, 2018.

Z. Liu, C. Chen, W. Li, P. Qi, T. Pang, C. Du, W. S. Lee, and M. Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

H. Namkoong and J. C. Duchi. Stochastic gradient methods for distributionally robust optimization with f-divergences. In *Advances in Neural Information Processing Systems*, 2016.

Y. Oh, J.-H. Park, J. Kim, S. H. Kim, and S. K. Lee. Incorporating domain knowledge into materials tokenization. In *Proceedings of ACL*, pp. 9623–9644, 2025.

S. S. Ramesh, X. Ji, M. Zimmer, S. Yoon, Z. Wang, H. Bou Ammar, A. Lucchi, and I. Bogunovic. Multi-task grpo: Reliable llm reasoning across tasks. *arXiv preprint arXiv:2602.05574*, 2026.

S. Sagawa, P. W. Koh, T. B. Hashimoto, and P. Liang. Distributionally robust neural networks for group shifts. In *International Conference on Learning Representations*, 2019.

O. Sener and V. Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, 2018.

Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yi Tay, Vinh Q Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. Charformer: Fast character transformers via gradient-based subword tokenization. *arXiv preprint arXiv:2106.12672*, 2021.

A. Trewartha, N. Walker, H. Huo, S. Lee, K. Cruse, J. Dagdelen, A. Dunn, K. A. Persson, G. Ceder, and A. Jain. Quantifying the advantage of domain-specific pre-training on named entity recognition tasks in materials science. *Patterns*, 3(4), 2022.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306, 2022.

T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, 2020.

A Appendix

A.1 The grammar in full

Table 3 lists the slots the policy decides and their cardinalities. Table 4 gives the slots the composite grammar adds on top of them.

Table 3: The grammar as the slots the policy decides, with cardinalities for the reported configuration, in which case-dependent options are withheld. A slot is decided only when the chosen kind reads it, so a rollout makes 7.2 decisions rather than 11.

slot	$\lvert\cdot\rvert$	values	decided when
`detector`	3	`always`, `digit_bearing`, `alphabetic`	always
`min_len`	3	$2, 3, 4$	always
`max_len`	3	$16, 24, 32$	always
`kind`	5	`identity`, `fixed_width`, `transition`, `greedy_pattern`, `longest_literals`	always
`width`	8	$1..8$	`fixed_width`
`transitions`	3	non-empty subsets of {`letter_digit`, `digit_letter`}	`transition`
`patterns`	4	`[A-Za-z]+ | \d+`; `.{1,2}`; `[a-z]+ | \d+`; `\d+ | [A-Za-z]+`	`greedy_pattern`
`n_literals`	4	$16, 32, 48, 64$ mined substrings	`longest_literals`
`longest_match`	2	true, false	`greedy_pattern`, `longest_literals`
`attach_digits`	7	`to_previous`, `to_next`, `separate`, `group_digits`, `numeric_suffix`, `by_prev_class`, `preserve_numeric`	always
`exception`	3	`leave_unsplit`, `absorb_unknown`, `all_or_nothing`	always

Composite rules. The slot list of Appendix A.1 describes a single segmentation policy. Writing $\mathcal{B}$ for the set of base policies it defines, a *composite* grammar closes $\mathcal{B}$ under two constructors,

$$\mathcal{S} \;=\; \mathcal{B} \;\cup\; \{\textsc{pipe}(b_1, \ldots, b_n) : b_i \in \mathcal{B},\; 2 \le n \le 3\} \;\cup\; \{\textsc{cond}(c, b_{\mathrm{t}}, b_{\mathrm{f}}) : b_{\mathrm{t}}, b_{\mathrm{f}} \in \mathcal{B}\}, \qquad (17)$$

with c drawn from seven string predicates (`has_digit`, `no_digit`, `len_gt_4`, `len_gt_8`, `digit_heavy`, `starts_digit`, `ends_digit`). PIPE applies its stages left to right, each stage cutting the pieces the previous one produced. COND applies b_{t} to strings satisfying c, and b_{f} to the rest. The constructors do not nest: a stage is always a base policy, which bounds the space and keeps rollouts short.

The policy expresses this without a recursive sampler. Every scope a rule could use is present in one static slot list, and a scope's slots are decided only when the decisions above select it. Stage s is live when `kind` is `pipeline` and $s \le$ `n_stages`, or when `kind` is `conditional` and $s \in \{1, 2\}$. In the conditional case stage 1 is the `then` branch and stage 2 the `otherwise` branch. Table 4 in Appendix A.1 gives the added slots. In the grammar we report this raises the reachable behavior count on our probe set from 111 to 492 over 4000 rollouts, and the parameter count from 31,533 to 47,195. The sampler, the deduplication and the credit assignment are unchanged.

A.2 Dataset sizes

Table 5 reports the dataset sizes and the fit, evolve, select, and test splits used for each task. The search tasks follow an approximately 48/16/16/20 split, with the fit set used for model training, evolve for search reward, select for choosing the final rule, and test reserved for final evaluation.

B Optimization details

Encoder and fine-tuning. Every rewrite arm uses the same pretrained BERT encoder: 6 layers, hidden 512, 8 heads, and a 31,090-entry WordPiece vocabulary, pretrained on a materials corpus. Scoring a rule fine-tunes that checkpoint end to end, with all 35M parameters receiving gradient. Fine-tuning uses AdamW at 5×10^{-5}, weight decay 0.01, batch 32, a linear schedule with 10% warmup and gradient clipping at 1.0. Epoch counts are 8 for the classification tasks and ProteinGym,

Table 4: Slots added by the composite grammar, on top of Table 3. The stage group is instantiated three times, giving 34 slots. A base kind inside a stage cannot itself be composite.

slot	$\lvert\cdot\rvert$	values	decided when
`kind` (extended)	7	the five base kinds, plus `pipeline`, `conditional`	always
`n_stages`	2	$2, 3$	`pipeline`
`cond`	7	the seven predicates above	`conditional`
`kind_`ss	5	the five base kinds	scope s live
`width_`ss	8	$1..8$	`kind_`ss is `fixed_width`
`transitions_`ss	3	as `transitions`	`kind_`ss is `transition`
`patterns_`ss	4	as `patterns`	`kind_`ss is `greedy_pattern`
`n_literals_`ss	4	as `n_literals`	`kind_`ss is `longest_literals`
`longest_match_`ss	2	true, false	`greedy_pattern`, `longest_literals`
`attach_`ss	7	as `attach_digits`	scope s live

Table 5: Dataset sizes, with materials splits made formula-disjoint and the ClinVar splits made string-disjoint. Of the five ProteinGym assays only PSEAI is in the search reward, and the other four assays, ClinVar and every task below the first row are transfer.

task	target	fit	evolve	select	test	metric
crystal system	7-class	16,152	5,383	5,383	6,730	accuracy
is-metal	binary	2,362	787	788	984	accuracy
glass formation	binary	2,726	908	910	1,136	accuracy
band gap	scalar	2,210	736	738	920	MAE
dielectric	scalar	2,333	773	734	924	MAE
$\log G_{\mathrm{VRH}}$	scalar	5,247	1,758	1,803	2,179	MAE
$\log K_{\mathrm{VRH}}$	scalar	5,247	1,758	1,803	2,179	MAE
peptide iRT	scalar	25,000	5,000	5,000	20,000	Spearman ρ
pgym PSEAI	scalar	2,419	806	807	972	Spearman ρ
pgym BLAT_Stiffler	scalar	2,388	796	796	1,016	Spearman ρ
pgym BRCA1	scalar	890	296	298	353	Spearman ρ
pgym INFA_Wu	scalar	1,129	376	377	468	Spearman ρ
pgym NEIME	scalar	448	149	150	175	Spearman ρ
ClinVar `c.`	binary	16,809	5,603	5,604	7,004	accuracy

15 for band gap and 6 for peptides. Doubling the search budget from 120 to 240 domain evaluations returned the same final rule. The Table 6 gives the search hyperparameters for reproducibility.

Learned-tokenizer baselines. Each baseline is a vocabulary trained on our pretraining corpus, followed by an encoder pretrained from scratch with that vocabulary. Architecture, corpus, optimizer steps, batch size, learning rate and seed are identical for every arm, so the only thing that varies is the vocabulary. The encoder is a 6-layer BERT (hidden size 512, 8 heads, intermediate size 2048, sequence length 128; 35.2M parameters) trained for 21,195 steps at batch 64 and learning rate 10^{-4} with seed 0, which is five passes over the 34.7M-token corpus.

WordPiece, BPE and Unigram use the standard trainers in the HuggingFace `tokenizers` library. All three share the same BERT normalizer and pre-tokenizer, so only the subword algorithm differs. We deliberately avoid a byte-level BPE, which would also change pre-tokenization and case handling and would confound the comparison.

PickyBPE is trained with the authors' reference implementation and then exported to a HuggingFace tokenizer directory. PickyBPE interleaves merges with token removals, and a HuggingFace BPE holds only an ordered merge table, so the export keeps the final vocabulary together with the merges that survive. This is the same approximation used in the MATTER evaluation.

MATTER's trained vocabulary is not publicly released, so we rebuild it on our corpus from the authors' tokenization code and MatDetector checkpoint. We therefore do not compare against the numbers reported in their paper, which are measured on a different corpus and benchmark. Vocabularies contain 31090 tokens for every arm except PickyBPE, whose refinement procedure terminates at 32538.

Table 6: GCMS search hyperparameters. Settings used for the headline multi-domain search.

Hyperparameter	Value
Search	
Search budget B	120 domain evaluations
Maximum group size G	12
Minimum distinct behaviours per group	8
Uniform grammar reserve ν	0.15
Composite-rule sampling probability	0.50
Sampling temperature	1.3
Probe-set size	200 strings (100 per domain)
Mined literal inventory	64 literals
Objective	
Soft-min temperature β	1.0
Advantage denominator floor ϵ	0.02
Degenerate-group threshold ϵ_{var}	1×10^{-4}
Policy	
Policy architecture	GRU cell, 1 layer, hidden size 64
Policy parameters	31,533
Policy optimizer	Adam
Policy learning rate	0.02
Entropy coefficient λ_H	0.02 (adaptive, capped at 0.30)
Reward evaluation	
Encoder fine-tuning epochs	8
Batch size	32
Encoder learning rate	5×10^{-5}
Maximum sequence length	64 tokens

Scale of the soft-minimum reward. R is a ranking signal at a fixed β rather than a calibrated score. It carries an offset of $-\log K/\beta$ relative to the per-domain z_k: when every domain improves equally by z, the reward is $z - \log K/\beta$ rather than z. With $K = 2$ and $\beta = 1$ that offset is -0.693. It is constant within a run, so it cannot reorder rules and does not affect the search or any result reported at $\beta = 1$. It does make R incomparable across values of β, where the offset runs from -2.773 at $\beta = 0.25$ to -0.347 at $\beta = 2$, and across different numbers of domains. Comparisons of that kind therefore use the per-domain z_k. As an optimization control, removing the group standard-deviation normalization failed across the tested learning rates, supporting its role in stabilizing policy updates. Estimating σ_k from identity was conservative and did not change the selected rule, indicating that the final search outcome is not sensitive to this calibration choice.

The optimizer draws one component from each of two sources, which we state explicitly. From GRPO (Shao et al., 2024) we take the *group-relative advantage estimator*. Rewards are baselined against the mean of a group of samples drawn under the same conditions and scaled by that group's standard deviation, which removes the need for a learned value function and makes a constant offset in the reward irrelevant. From REINFORCE (Williams, 1992) we take the *update rule* itself, a single on-policy gradient step on $-A_i \log \pi_\theta(a_i)$ with an entropy bonus, applied once per group to freshly drawn rollouts.

What we do not take is the PPO surrogate that GRPO inherits. There is no importance ratio, no ratio clipping and no KL penalty against a reference policy. This is a consequence of the setting rather than a simplification for its own sake. Every rollout here costs an encoder fine-tune, so rollouts are the expensive resource and are never reused across updates. With one on-policy step per group the importance ratio is identically one and the clipped surrogate reduces exactly to the vanilla policy-gradient term, so the clipping machinery would be inert. It earns its place only when several epochs are taken over the same rollouts, which at this evaluation cost would be the wrong trade.

Advantages are group-relative, $A_i = (R_i - \bar{R})/\max\big(\mathrm{sd}(R), \epsilon\big)$ with $\epsilon = 0.02$, which requires one remark. The division by the group standard deviation is what makes the update scale-invariant, and with two domains on different metrics that invariance stops being a convenience: removing it made the learning rate fragile, one setting under-training while a $20\times$ larger one collapsed to a near-deterministic policy within five iterations. Its only real defect is the degenerate case, where a group whose rewards are identical divides by nearly zero, so GCMS floors the denominator instead of removing it, bounding amplification while keeping scale invariance.

Table 7: Why the soft minimum. σ-normalized improvements on the two training domains under three ways of combining them. A mean rewards a rule that fails a domain, and a hard minimum ties the top three because ProteinGym saturates.

rule	crystal	ProteinGym	mean	hard min	**soft-min**
charsplit + preserve-numeric	$+9.91$	$+2.73$	$+6.32$	$+2.73$	$\mathbf{+2.73}$
transition + separate	$+3.64$	$+2.73$	$+3.19$	$+2.73$	$+2.39$
pipeline (hard-min winner)	$+3.47$	$+2.73$	$+3.10$	$+2.73$	$+2.34$
character split	$+9.53$	-3.58	$+2.97$	-3.58	-3.58

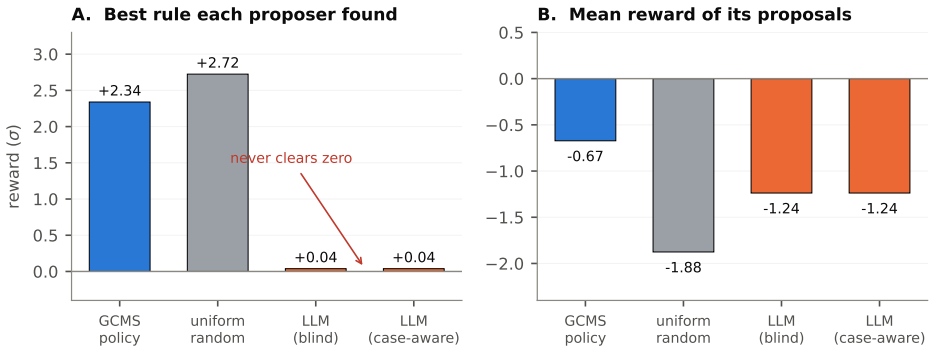


Figure 3: Comparison of proposers. Reward attained by the rules each proposer generated at seed 3, with grammar, budget, group size and objective held fixed. **(A)** best proposal, **(B)** mean proposal.

Groups are built from behaviorally distinct rules, hashed on their rewrites over the probe set rather than on their fields, and sized to what the grammar can actually express on the given inputs, which is 149 reachable behaviors on formulas against 4 on mutation strings. A fixed group size would otherwise leave the sampler exhausting its retries on the latter. A fixed fraction of each group is drawn from the uniform grammar rather than from the policy. These off-policy samples contribute to the baseline and to diversity but receive no gradient, since weighting them by the policy's own log-probability would bias the estimator. Identity and character-split behaviors are seeded into the sampler's exclusion set, so a trivial baseline can never be proposed and returned as a discovery. Finally, a group whose reward spread falls below a threshold carries no learning signal and produces no update at all. Fresh uniform behaviors are injected instead, and the entropy coefficient is raised until behavioral diversity recovers.

C Additional Experiments and Analyses

C.1 Transferable and domain-specific input structure

The controlled rewrite experiments separate the structure that transfers across scientific domains from choices that are specific to one notation. Plain character splitting performs strongly on chemical-formula tasks but reduces ProteinGym Spearman to 0.8297, below the identity representation at 0.8566. GCMS instead reaches 0.8754 by retaining fine-grained symbolic segmentation while preserving multi-digit spans (Table 8).

Some structure is more domain-specific. Giving the grammar access to letter case produces a substantially stronger materials representation: crystal accuracy increases from 0.6147 to 0.6594, is-metal from 0.8913 to 0.9126, and band-gap MAE falls from 0.4918 to 0.4431. The resulting rule behaves like an element-aware segmentation despite receiving no element list, but it does not transfer to modified peptides. The blind grammar therefore sacrifices some materials-specific performance in exchange for a representation that can be used across notations.

Modified peptides separate segmentation transfer from detector transfer. The searched rule initially rewrites only 63% of peptide inputs and reaches 0.9304 Spearman because its learned length bound reflects the shorter search domains. Increasing only this bound raises performance to 0.9649, close to the hand-built PTM lexicon at 0.9660. Thus the learned segmentation behavior transfers more

Table 8: Input-rewrite controls and ablations. Accuracy is reported for materials classification tasks, MAE for materials regression tasks, and Spearman ρ for life-science tasks. Values are mean $\pm$ standard deviation over three evaluation seeds. Material-domain uses crystal-system and is-metal classification as the two search tasks. LifeSciences domain uses ProteinGym assays PSEAI and BLAT regression as the two search tasks. Blank cells were not run or are not informative because of very low rewrite coverage. ‡ The case-aware grammar adds options that let a rule split on capitalization. ↑ indicates higher is better and ↓ indicates lower is better.

task	LifeSciences-domain	matched-random	Material-domain	SEC-style	**GCMS**	case-aware‡
Materials						
crystal ↑	0.5925 ±0.0054	0.5872 ±0.0057	0.6143 ±0.0050	0.6077 ±0.0013	0.6147 ±0.0017	**0.6594** ±0.0027
is-metal ↑	0.8547 ±0.0030	0.8496 ±0.0061	0.8943 ±0.0010	0.8818 ±0.0021	0.8913 ±0.0081	**0.9126** ±0.0051
glass ↑	0.8016 ±0.0033	—	0.8131 ±0.0065	0.8104 ±0.0052	0.8269 ±0.0075	**0.8172** ±0.0085
band gap ↓	0.5553 ±0.0074	—	0.5014 ±0.0208	0.4928 ±0.0221	0.4918 ±0.0044	**0.4431** ±0.0043
dielectric ↓	0.5470 ±0.0098	—	0.4908 ±0.0019	0.5095 ±0.0011	0.4998 ±0.0028	**0.4197** ±0.0020
log G_{VRH} ↓	0.1781 ±0.0009	—	0.1463 ±0.0023	0.1453 ±0.0016	0.1500 ±0.0009	**0.1238** ±0.0005
log K_{VRH} ↓	0.1589 ±0.0017	—	0.1280 ±0.0029	0.1278 ±0.0013	0.1358 ±0.0022	**0.1084** ±0.0007
Life sciences						
peptide iRT ↑	0.8907 ±0.0100	0.9117 ±0.0124	0.9234 ±0.0046	0.9630 ±0.0006	**0.9649** ±0.0007	—
ProteinGym ↑	0.8682 ±0.0019	0.8093 ±0.0054	0.8297 ±0.0040	0.8612 ±0.0013	**0.8754** ±0.0019	—

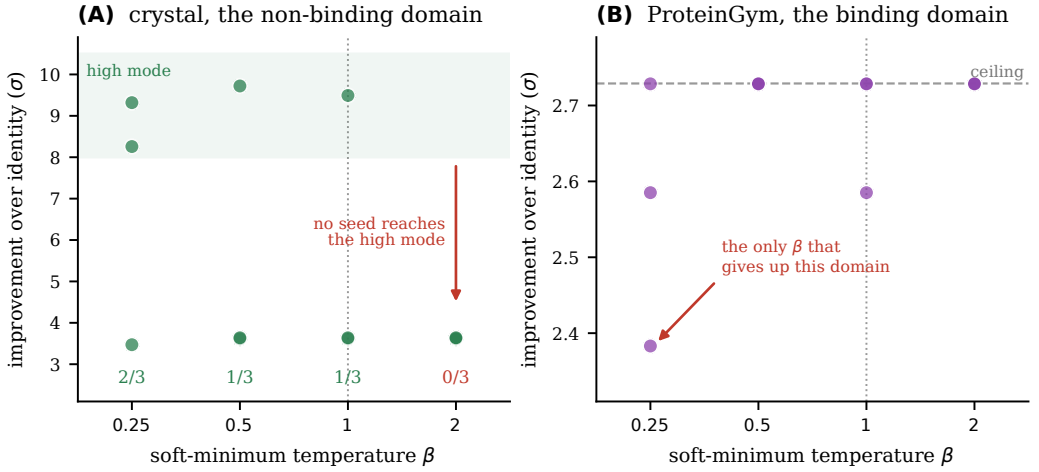


Figure 4: Soft-minimum temperature. Improvement over identity in seed standard deviations, one point per search seed. **(A)** The crystal winner is bimodal and the fraction counts seeds in the high mode, none at β=2. **(B)** ProteinGym leaves its ceiling only at β=0.25.

strongly than the detector range in which it was discovered. Simple heuristics do not provide the same cross-domain solution (Table 9).

Search controls. The remaining experiments test whether these conclusions depend on the search mechanics. Sweeping the soft-min temperature shows the transition from mean-like compensation to hard-min saturation (Figure 4). Across five search seeds, the learned policy recovers one winning segmentation family with an observed best-reward standard deviation of 0.029, compared with 0.335 for matched random search. Finally, allowing identity and plain character splitting does not improve the outcome: the returned rule performs at or below the reported GCMS rule on every evaluated task.

C.2 Transfer to clinical variant notation

We test the rule on a fourth notation it was not searched on: coding-DNA variants from ClinVar, written in HGVS form. The task is to predict clinical significance, pathogenic against benign, from the variant string alone.

Inputs are the coding-DNA portion of the HGVS name with the accession removed, so `NM_014630.3(ZNF592):c.3136G>A` becomes `c.3136G>A`. Removing the accession takes gene identity out of the string, which would otherwise let a model predict from a per-gene base rate rather than from the variant, and it brings every string inside the rule's length bound, raising coverage

Table 9: Identity, character splitting and GCMS across all evaluation tasks. Accuracy for classification, MAE for materials regression, Spearman ρ for life-science tasks. Mean $\pm$ standard deviation over three evaluation seeds.

Task	Identity	Character split	**GCMS**
Materials			
crystal* ↑	0.5848 ±0.0007	0.6176 ±0.0014	0.6147 ±0.0017
is-metal ↑	0.8509 ±0.0033	0.8970 ±0.0021	0.8913 ±0.0081
glass ↑	0.8102 ±0.0033	0.8219 ±0.0013	0.8269 ±0.0075
band gap ↓	0.6129 ±0.0185	0.4907 ±0.0100	0.4918 ±0.0044
Life sciences			
peptide iRT ↑	0.8907 ±0.0100	0.9638 ±0.0009	0.9649 ±0.0007
ProteinGym BLAT ↑	0.8657 ±0.0024	0.8484 ±0.0076	0.8789 ±0.0002
ProteinGym BRCA1 ↑	0.6092 ±0.0114	0.4885 ±0.0091	0.5874 ±0.0042
ProteinGym INFA ↑	0.2308 ±0.0028	0.2400 ±0.0193	0.3158 ±0.0140
ProteinGym NEIME ↑	0.0396 ±0.0293	0.0637 ±0.0184	0.1593 ±0.0291
ProteinGym PSEAI* ↑	0.8566 ±0.0020	0.8297 ±0.0040	0.8754 ±0.0019
ClinVar `c.` ↑	0.8183 ±0.0092	0.7656 ±0.0006	0.8221 ±0.0050

Table 10: ClinVar coding-DNA variants. Accuracy over three evaluation seeds $(42/7/1)$ on a notation the search never saw, with bold marking the proposed method. WordPiece and *identity* are the same arm and agree exactly, which checks the two code paths against each other.

method	accuracy	`c.3136G>A` becomes
learned tokenizers, own encoder		
MATTER (Oh et al., 2025)	0.8210 ±0.0066	`c . 313 ##6 ##g > a`
BPE	0.7705 ±0.0053	`c . 3 136 g > a`
WordPiece	0.8183 ±0.0092	`c . 313 ##6 ##g > a`
PickyBPE	0.7882 ±0.0183	`c. 3 13 6 g > a`
Unigram	0.8045 ±0.0035	`c . 3136 g > a`
input rewrites, one pretrained WordPiece encoder		
identity	0.8183 ±0.0092	`c.3136G>A`
character split	0.7656 ±0.0006	`c . 3 1 3 6 G > A`
GCMS	**0.8221** ±0.0050	`c . 3136 G > A`

from 11.2% to 100%. We keep single-nucleotide variants, map pathogenic and likely pathogenic to one class and benign and likely benign to the other, and discard variants of uncertain or conflicting significance. One coding-DNA string occurs in several genes, so strings are deduplicated and a string whose records disagree on the label is dropped rather than resolved by majority. This leaves $35{,}020$ strings, split $48/16/16/20$ over unique strings, with 44% positive. Values come from the first $300{,}000$ records of the ClinVar release, which span $12{,}447$ genes.

Table 10 gives the result. No arm improves on leaving the string unchanged, and the learned tokenizers are mostly worse: three of the four fall below identity, BPE by 4.8 points. Character splitting is the weakest arm at 0.7656, because it breaks the coordinate `3136` into single digits and destroys the position, which is the same failure it shows on protein mutations. The rule GCMS selects keeps digit runs whole and is the highest value in the table, though it does not separate from identity.

C.3 Comparison with a language-model proposer

A pretrained language model encodes substantial chemical and biological knowledge, which makes it a natural alternative to a learned policy for proposing candidate rules. We evaluate this alternative by substituting Qwen2.5-7B for the policy while holding the grammar, the evaluation budget of 120 rules, the group size, the uniform reserve and the objective fixed, so that the proposer is the only point of difference. Figure 3 in Appendix C.1 reports the search reward attained by the rules each proposer generated. Over three search seeds the language model's best proposal averages -0.14 and exceeds zero at one seed of three, while the uniform reserve drawn inside the same runs averages $+2.16$. Comparing against that reserve holds the run fixed, so the proposer is the only difference. These are search rewards on the training objective and not sealed performance, and they are a separate comparison from Table 1. The result is not explained by sampling volume, since the language model

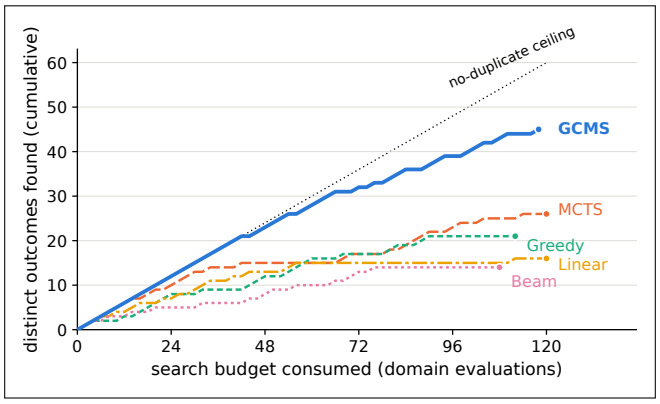


Figure 5: Search efficiency at matched budget. Cumulative distinct rewards against evaluations consumed. The dotted line is the no-duplicate ceiling of one new outcome per rule.

contributed 38 of the 59 distinct behaviors evaluated in its run, distributed across six segmentation kinds.

The mode of failure is not an inability to produce good rules. On formulas the language model produces the strongest rules we have measured. Its highest-scoring formula proposal is an element-symbol parser recovered without an element list, mapping `LiMn2O4` to `Li Mn2 O4` and `La0.7Sr0.3MnO3` to `La 0.7 Sr 0.3 Mn O3`, including the fractional subscripts that a fixed element inventory cannot parse. Evaluated under the protocol of Section 4, it reaches 0.6539 on crystal, 0.8991 on is-metal and 0.4145 MAE on band gap, the last of which is below every arm in Table 1.

That rule does not transfer. It rewrites none of the 5004 mutation strings in the PSEAI assay and none of 800 peptide strings, so on both biological notations it reduces to the identity rule. The reason is that the rule is specific to formula orthography. Its detector requires mixed capitalization, which the all-capital mutation strings such as `H24W` do not have, and its length bound of 20 characters excludes peptides, whose median length is 27.

The pattern holds across the model's proposals rather than only its best one. Of 112 rules proposed over three search seeds, none segments `V10G` into `V 10 G`, the parse that isolates the substitution position. Roughly half leave the string unchanged, and most of the remainder attach the position digits to the preceding residue, giving `V10 G`. That choice is the correct reading of a chemical formula, in which a numeral modifies the symbol before it, as in `Mn2`. In mutation notation the numeral is an independent coordinate, so the same convention removes the structure the assay depends on. Domain knowledge therefore transfers to the notation it was acquired from and not to the others, which is the behavior the multi-domain objective is designed to penalize.

C.4 Alternative Grammar Search Strategies

We compare GCMS with greedy decoding, beam search, and Monte Carlo tree search over the same rewrite grammar. All methods use the same downstream evaluation setup, while differing only in how candidate rules are selected from the grammar. Table 11 reports transfer to representative held-out tasks. Among the search strategies, GCMS shows the strongest transfer across held-out tasks from different scientific domains. We report band gap, peptide iRT, and ClinVar as representative examples, and GCMS also matches or exceeds the other search strategies across the remaining held-out datasets.

Table 11: Alternative search strategies over the same grammar. GCMS shows the strongest transfer across held-out tasks from different scientific domains, including band gap, peptide iRT, and ClinVar. Results are mean ± standard deviation over three evaluation seeds.

Method	Band gap MAE ↓	Peptide iRT ↑	ClinVar ↑
Greedy	0.5337 ± 0.0070	0.8907 ± 0.0100	0.8211 ± 0.0033
MCTS	0.5415 ± 0.0073	0.8907 ± 0.0100	0.8211 ± 0.0033
Beam	0.5296 ± 0.0106	0.9317 ± 0.0030	0.7626 ± 0.0060
GCMS	**0.4918 ± 0.0044**	**0.9649 ± 0.0007**	**0.8221 ± 0.0050**

At the same evaluation budget, the search methods differ mainly in how many distinct outcomes they explore as shown in Figure 5. A rule costs two evaluations, one per domain, so the 120-evaluation budget covers 60 rules. GCMS produces 45 distinct rewards, compared with 26 for MCTS, 21 for greedy search, 16 for the linear chain, and 14 for beam search. Since each rule is evaluated on two domains, at most 60 distinct rule outcomes can be obtained from 120 domain evaluations. GCMS reaches 75% of this maximum, while no baseline exceeds 43%. The difference comes largely from repeated evaluations of behaviorally similar rules in the baselines. GCMS reduces this redundancy through behavioral deduplication and its uniform-grammar reserve.