SELF-STRUCTURING LATENT ACTION SPACES FOR BLACK-BOX INSTRUCTION OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Instruction optimization for large language models (LLMs) is a high-dimensional discrete black-box optimization problem in which reliable feedback comes only from costly model evaluations. Existing methods search directly in raw text or use LLMs as high-frequency mutation operators, which makes the search hard to analyze and ties generation-side cost to population size and optimization depth. We propose Self-Structuring Latent Action-Space Evolution (SLAE), which represents each instruction as an integer-coded genotype that selects a syntactic scaffold and slot-wise semantic actions and is mapped deterministically to text. Routine search runs in this categorical genotype space without LLM calls, while the space itself evolves: Dynamic Structural Evolution (DSE) adds newly discovered scaffolds, and Adaptive Action-Space Refinement ($\mathrm{A}^2\mathrm{R}$) adds slot-level actions when search stagnates. Because genotype coordinates are nominal categories, we show that differential-evolution arithmetic on them depends on how categories are indexed and give a label-equivariant variation rule. Under matched evaluation budgets, SLAE with integer-coded differential evolution achieves the best results on all nine language understanding and generation benchmarks we evaluate. In controlled studies under the main experimental protocol, disabling the expansion mechanisms costs 8.7 accuracy points on AG News and 5.3 on a held-out task with a different model family, while on AG News these mechanisms account for about 0.1% of the measured token budget.

1 INTRODUCTION

Instruction optimization for large language models (LLMs) is a high-dimensional discrete black-box optimization problem: candidate instructions are textual objects, and reliable feedback comes only from costly model evaluations (Brown et al., 2020; Grattafiori et al., 2024; Sahoo et al., 2025; Ramnath et al., 2025). A common strategy searches directly over raw text, generating candidate instructions, revising previous ones, or using an LLM as a proposal, mutation, or crossover operator (Zhou et al., 2022; Guo et al., 2024; Yang et al., 2024b; Agrawal et al., 2026). Such methods produce fluent instructions but make the search opaque: structural choices, semantic edits, and surface realization are entangled in one free-form generation step, so it is difficult to tell whether an improvement comes from a better structure, a better semantic choice, or an incidental wording change. Moreover, when the LLM serves as a high-frequency mutation operator, generation-side cost grows with the number of candidates and iterations, limiting the search depth affordable under a fixed budget.

A structured latent representation can make this search more controllable by separating the optimized representation from text-level proposal generation, a principle also used in structured black-box optimization and neural architecture search (Real et al., 2019; Elsken et al., 2019). Yet a fixed representation can be too restrictive: useful instructions may require scaffolds or semantic actions not specified before search begins.

We propose Self-Structuring Latent Action-Space Evolution (SLAE), which casts instruction optimization as evolutionary search over a self-structuring discrete latent action space. Each instruction is an integer-coded genotype: one coordinate selects a syntactic scaffold, the others select slot-wise semantic actions, and a deterministic mapping $\mathrm{Materialize}(\cdot)$ converts the genotype into an executable instruction. Routine offspring generation therefore proceeds in genotype space without LLM calls. The space itself is not static: Dynamic Structural Evolution (DSE) adds newly discovered scaffolds

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.

to the library through a sparse, generation-level Structural Mutation Operator (SMO), and Adaptive Action-Space Refinement ($\mathrm{A}^2\mathrm{R}$) adds slot-level actions when search stagnates. LLM calls are thus reserved for representation-space expansion. We report them separately from the matched evaluation budget, since two methods may evaluate the same number of candidates while making very different numbers of generation-side calls.

Because genotype coordinates are nominal categories, a numerical operator applied to their indices can make the search depend on how the categories happen to be numbered. We analyze this dependence for variation in the style of differential evolution and give a label-equivariant variation rule. We make the following contributions:

- We formulate black-box instruction optimization as structured search over a discrete latent action space, where a deterministic genotype-to-instruction mapping makes candidate instructions explicit, analyzable, and searchable without per-candidate LLM calls.
- We introduce SLAE, a self-structuring evolutionary framework that expands the latent action space during optimization through DSE and $\mathrm{A}^2\mathrm{R}$.
- We show that differential-evolution arithmetic on nominal genotype coordinates depends on how categories are indexed, and give a label-equivariant variation rule with a formal equivariance guarantee (Proposition 1).
- We provide a matched-budget empirical study in which SLAE, with integer-coded differential evolution, achieves the best results on all nine benchmarks against human-authored, LLM-generated, and evolutionary baselines, while reducing generation-side LLM calls from $\mathcal{O}(N_p T)$ to $\mathcal{O}(T)$. Controlled studies under the main protocol show that disabling DSE and $\mathrm{A}^2\mathrm{R}$ costs 8.7 and 5.3 accuracy points on two tasks, that these mechanisms take about 0.1% of the token budget on AG News, that the gain is not explained by prompt initialization, and that on AG News it persists under the label-equivariant rule (Sections 4.2 and 4.5).

2 Related Work

Automated prompt optimization. APE (Zhou et al., 2022) proposes and filters candidates with an LLM, and APO (Pryzant et al., 2023) refines a prompt through LLM-written critiques, which is effective locally but struggles to move far from the starting point. OPRO (Yang et al., 2024b) and GEPA (Agrawal et al., 2026) likewise ask an LLM to propose revised instructions from optimization feedback. EvoPrompt (Guo et al., 2024) uses an LLM as the mutation and crossover operator of an evolutionary algorithm over prompt text, which yields fluent offspring but makes search dynamics hard to analyze and ties generation-side cost to population size and iterations, and Promptbreeder (Fernando et al., 2024) additionally evolves the mutation prompts that direct this LLM-based mutation. PromptQuine (Wang et al., 2025) evolves in-context prompts through aggressive token-level pruning, often producing strong but less readable contexts. In all of these methods an LLM, or free-form text editing, is the routine search operator. SLAE instead performs routine search in an explicit genotype space and calls the LLM only to expand that space. Other methods search over discrete edits or tokens: GrIPS (Prasad et al., 2023) applies phrase-level edits to an instruction, RLPrompt (Deng et al., 2022) trains a policy that generates discrete prompt tokens, and TEMPERA (Zhang et al., 2023a) learns to edit instructions, demonstrations, and verbalizers at test time. These methods fix their edit operations or action space before optimization, whereas SLAE expands its scaffold library and action sets during search.

Structured representations for black-box optimization. Optimization in non-numerical domains hinges on the representation (Eiben & Smith, 2015), as in genetic programming over syntax trees (Koza, 1993) and neural architecture search over structured genotypes (Real et al., 2019). LLM systems likewise impose manually defined structures on black-box models, such as the reasoning-action traces of ReAct (Yao et al., 2023), the modular pipelines of DSPy (Khattab et al., 2024), and the tree search of LATS (Zhou et al., 2024). These structures are fixed before optimization and therefore bound the search space. SLAE instead lets the representation co-evolve with the solution, expanding both the scaffold grammar and the slot vocabulary during search.

Categorical variables in black-box optimization. Optimizers designed for continuous spaces are often applied to discrete variables by mapping continuous proposals to the nearest valid value, as in

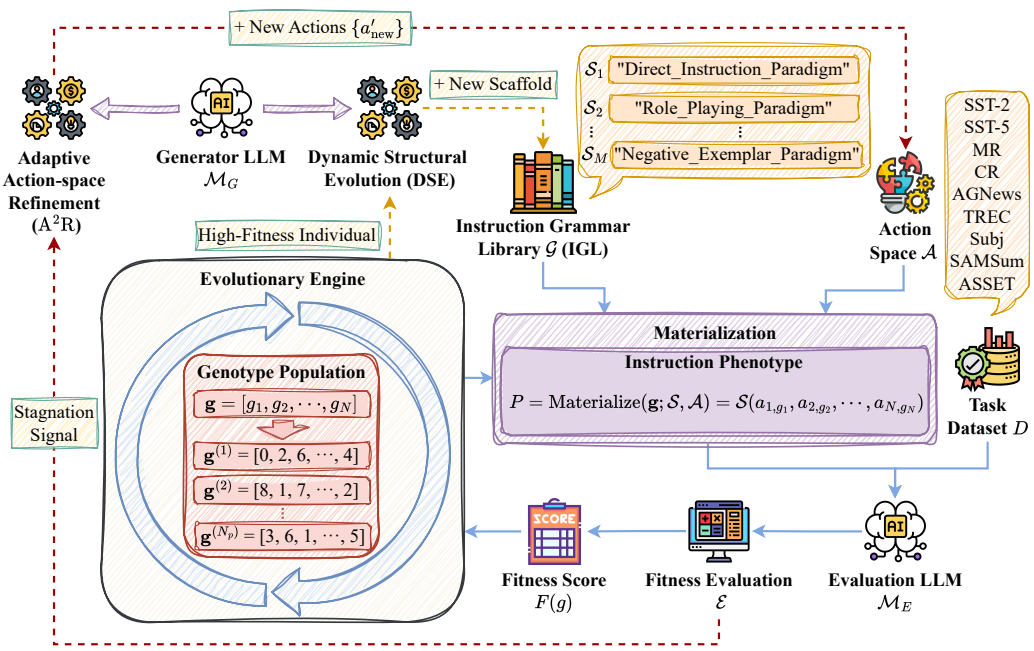


Figure 1: Overview of SLAE. A fast inner loop evolves integer-coded categorical instruction genotypes without LLM calls, where the structural gene selects from the current scaffold library and slot-wise genes select semantic actions. Two slower representation-space expansion mechanisms make the latent action space self-structuring. DSE expands the scaffold library through a generation-level SMO event, which may add at most one new scaffold per generation, while A^{2}R expands selected semantic action sets when stagnation is detected. Routine offspring generation remains in genotype space, so LLM calls are reserved for sparse representation-space expansion rather than per-individual text mutation.

integer-coded DE (Price et al., 2005); for Bayesian optimization, Garrido-Merchán & Hernández-Lobato (2020) show that such naive treatments of integer and categorical inputs degrade performance and propose a transformed kernel. For nominal categories, rounding also imposes an order that the categories do not have. Methods designed for categorical spaces instead sample from distributions over categories, as in estimation-of-distribution algorithms (Larrañaga & Lozano, 2002) and CatCMA (Hamano et al., 2024). The label-equivariant landing rule of Section 3.2 follows this principle within DE-style variation: it relies only on equality tests between donors and uniform draws over the current categories.

3 The SLAE Framework

Figure 1 gives an overview. We first define the latent action space (Section 3.1), then the categorical search that operates in it (Section 3.2), and finally the mechanisms that expand it (Section 3.3).

3.1 Latent Action-Space Parameterization

Let $\mathcal{M}$ be an LLM that maps an instruction P and an input x to an output $\mathcal{M}(P, x)$. For a task with data distribution $\mathcal{D}_\mathcal{T}$ and evaluation metric $\mathcal{E}$, instruction optimization seeks

$$P^* = \arg\max_P \ \mathbb{E}_{(x,y)\sim\mathcal{D}_\mathcal{T}}\big[\mathcal{E}(\mathcal{M}(P, x), y)\big], \tag{1}$$

where every evaluation of the objective requires costly model calls. SLAE structures this search with syntactic scaffolds. A scaffold $\mathcal{S}$ is a template with N open slots $s_1, \ldots, s_N$, and slot s_i is filled by an action from a discrete action set $\mathcal{A}_i = \{a_{i,0}, \ldots, a_{i,k_i-1}\}$, where each action is a word or phrase.

Genotype-to-instruction mapping. An instruction is represented by an integer-coded genotype $\mathbf{g} = [g_0, g_1, \ldots, g_N]$. The structural gene g_0 indexes a scaffold in the current library $\mathcal{G}_t = \{\mathcal{S}_1, \ldots, \mathcal{S}_{M_t}\}$, and each semantic gene $g_i \in \{0, \ldots, k_i - 1\}$ indexes an action in $\mathcal{A}_i$. Each coordinate is a nominal category index: the integers identify categories but carry no order or distance. A deterministic function materializes the instruction,

$$P = \text{Materialize}(\mathbf{g}; \mathcal{G}_t, \mathcal{A}) = \mathcal{S}_{g_0}\big(a_{1,g_1}, a_{2,g_2}, \ldots, a_{N,g_N}\big), \tag{2}$$

so every genotype within bounds corresponds to a unique, executable instruction. A population is a set $\text{Pop} = \{\mathbf{g}^{(1)}, \ldots, \mathbf{g}^{(N_p)}\}$ of such genotypes.

Semantic Diversity Seeding. The initial action sets are built by Semantic Diversity Seeding (SDS): the generator model $\mathcal{M}_G$ proposes candidate actions for each slot, an LLM-based filtering step groups them by functional similarity, and one compact, function-preserving representative is kept per group, so that diversity comes from covering distinct functional groups (Algorithm 2).

3.2 Categorical Search in Genotype Space

The latent action-space parameterization turns instruction discovery into search over categorical genotypes. We use variation in the style of differential evolution (DE) (Storn & Price, 1997), in which offspring are generated from differences between current population members.

Variation and selection. For each target genotype $\mathbf{g}^{(j)}$ in Pop, three distinct donors $r_1, r_2, r_3 \neq j$ are sampled. Let $\mathcal{D}_i^t$ denote the current action set of slot i. At every slot where the two difference donors agree, $g_i^{(r_2)} = g_i^{(r_3)}$, the mutant inherits the base donor, $v_i^{(j)} = g_i^{(r_1)}$. Where they disagree, a landing rule decides where the coordinate moves. The integer landing rule applies the classical DE update to category indices and rounds the result, a common way of applying DE to integer variables (Price et al., 2005; Das & Suganthan, 2011),

$$v_i^{(j)} = \text{clip}_{\mathcal{D}_i^t}\Big(\text{round}\big(g_i^{(r_1)} + F\,(g_i^{(r_2)} - g_i^{(r_3)})\big)\Big), \tag{3}$$

with scaling factor F. The label-equivariant landing rule instead samples uniformly from the current action set,

$$v_i^{(j)} \sim \text{Uniform}\big(\mathcal{D}_i^t\big) \qquad \text{if } g_i^{(r_2)} \neq g_i^{(r_3)}. \tag{4}$$

The two rules differ only in where a disagreeing coordinate lands and share the remaining steps. Each slot is additionally resampled uniformly with probability $0.05 + 0.15\,(1 - \bar{H})$, where $\bar{H}$ is the mean pairwise normalized Hamming distance of the population, and a slot on which the whole population has converged is resampled among the other actions. The structural gene g_0 is resampled uniformly from the current scaffold library $\mathcal{G}_t$ with probability 0.1 and is otherwise inherited. Binomial crossover with rate CR combines target and mutant into a trial genotype, taking g_0 from either with equal probability. After all trials of a generation are evaluated on the development set, parents and trials are pooled: the fittest 30% survive, and binary tournaments among the rest fill the remaining slots.

Why the landing rule matters. Scaffold and action indices are nominal: renumbering the categories of a coordinate changes nothing about the instructions they materialize. The agreement test, the diversity and convergence checks, and every uniform draw above are unaffected by such renumbering, but Eq. (3) is not: its output depends on where categories sit in the numbering. Consider a slot with four actions $\{a, b, c, d\}$ indexed $0, 1, 2, 3$, $F = 0.5$, base donor b, and difference donors d and b. Eq. (3) gives $\text{round}(1 + 0.5\,(3 - 1)) = 2$, that is, action c. After swapping the indices of a and c, the donors keep the indices 1, 3, and 1, so the update again returns index 2, which now denotes action a. Two numberings of an identical population therefore produce different instructions. Eq. (4) removes this dependence.

Proposition 1 (Label equivariance)**.** *Let π relabel the categories of every coordinate by a bijection, applied consistently to the scaffold library, the action sets, and all genotypes. With the landing rule of Eq.* (4)*, the trial genotype produced from the relabeled population is distributed as the relabeling of the trial genotype produced from the original population. Consequently, the distribution of materialized instructions after one generation of variation and selection does not depend on how categories are indexed.*

Proof. Fix the population P and a relabeling $\pi = (\pi_0, \pi_1, \ldots, \pi_N)$, where π_0 acts on the scaffold library and π_i on the action set of slot i, and write $\pi(\mathbf{g})$ for coordinate-wise application. Donors are drawn by position in the population and are unaffected by π. For every slot i: (i) $\pi_i(a) = \pi_i(b)$ if and only if $a = b$, so the agreement test has the same outcome, and when donors agree the relabeled mutant is $\pi_i(g_i^{(r_1)})$; (ii) if U is uniform on a set S, then $\pi_i(U)$ is uniform on $\pi_i(S)$, so the landing draw of Eq. (4), the diversity-dependent resampling, and the resampling among the other actions at a converged slot all have the relabeled distribution; (iii) the Hamming diversity and the convergence test depend only on equalities and are unchanged. The same arguments apply to g_0, which is either inherited or resampled uniformly from the library. Binomial crossover selects positions independently of their values and takes g_0 from either parent with equal probability. Hence the trial genotype $\mathbf{u}'$ produced from $\pi(P)$ satisfies $\mathbf{u}' \stackrel{d}{=} \pi(\mathbf{u})$, where $\mathbf{u}$ is the trial produced from P. Fitness evaluation depends on a genotype only through its materialized instruction, and materialization under relabeled libraries satisfies $\mathrm{Materialize}(\pi(\mathbf{g}); \pi(\mathcal{G}), \pi(\mathcal{A})) = \mathrm{Materialize}(\mathbf{g}; \mathcal{G}, \mathcal{A})$. Selection ranks the pooled parents and trials by fitness and draws tournament pairs uniformly, so it uses only fitness values and draws that do not depend on labels. With the evaluation randomness (development batch and decoding) and the tournament draws coupled between the two runs, the relabeled run therefore selects the relabelings of the individuals selected in the original run; without coupling, this holds in distribution. □

The result covers variation and selection; initialization and representation expansion are outside its scope. The SLAE results in Tables 2 and 3 use the integer landing rule; Section 4.5 compares the two rules under matched conditions on AG News.

3.3 Self-Structuring Expansion

Inner-loop search can only recombine the scaffolds and actions that currently exist. SLAE therefore expands the representation space during optimization through two mechanisms.

Dynamic Structural Evolution (DSE). Structural search has two components. The structural gene g_0 lets routine genotype-space search select among all scaffolds in the current library without any LLM call. To create scaffolds beyond the library, SMO is activated at most once per generation with probability p_{smo}: it asks $\mathcal{M}_G$ to rewrite the syntactic structure of the current best instruction $P_{\mathrm{best},t}$ while preserving its semantic slots, and the parsed scaffold is added to the library,

$$\mathcal{G}_{t+1} \leftarrow \mathcal{G}_t \cup \{\mathrm{SMO}(P_{\mathrm{best},t})\}. \tag{5}$$

The new scaffold then becomes available to all subsequent search through g_0.

Adaptive Action-Space Refinement ($\mathrm{A}^2\mathrm{R}$). $\mathrm{A}^2\mathrm{R}$ expands the action sets when the best fitness stops improving over a window of generations. It first selects, among the current high-performing individuals, the slot whose choices are most associated with fitness variation. $\mathrm{A}^2\mathrm{R}$ then asks $\mathcal{M}_G$, given the task description and the actions already in $\mathcal{A}_c$, for a few new actions that are semantically distinct from the existing ones, and appends them,

$$\mathcal{A}_{c,t+1} \leftarrow \mathcal{A}_{c,t} \cup \{a'_{\mathrm{new}}\}. \tag{6}$$

The requirement of semantic distinctness steers expansion toward new search directions rather than synonyms of existing actions.

3.4 The Main Algorithm

Algorithm 1 gives the complete procedure. After SDS builds the initial action sets, the initial population is sampled from the same prompt pool used by EvoPrompt (Guo et al., 2024), which applies available manual prompts as its initial population: both methods use the same prompt files, population size, and sampling procedure, and SLAE encodes each sampled instruction into its genotype. The pool mixes 16 human-authored and 20 APE-generated instructions, and human-authored instructions make up 20–50% of each initial population; Section 4.5 tests the dependence on this pool.

Algorithm 1 Self-Structuring Latent Action-Space Evolution (SLAE)

Require: Task $\mathcal{T}$; population size N_p; generations T; stagnation window k_{stag}; SMO probability p_{smo}.
$\mathcal{G} \leftarrow$ initial scaffold library; $\mathcal{A} \leftarrow \text{SDS}(\mathcal{T})$
Pop $\leftarrow$ genotypes encoded from the shared prompt pool; evaluate Pop
for $t = 1$ to T **do**
  if best fitness has not improved for k_{stag} generations **then**
    $s_c \leftarrow$ slot most associated with elite fitness; $\mathcal{A}_c \leftarrow \mathcal{A}_c \cup \text{A}^2\text{R}(s_c)$ {Eq. (6)}
  end if
  if $\text{random}() < p_{\text{smo}}$ **then**
    $\mathcal{G} \leftarrow \mathcal{G} \cup \{\text{SMO}(P_{\text{best},t})\}$ {Eq. (5)}
  end if
  $\mathbf{u}^{(j)} \leftarrow$ categorical variation of $\mathbf{g}^{(j)}$ for every $\mathbf{g}^{(j)} \in$ Pop (Section 3.2)
  evaluate $\text{Materialize}(\mathbf{u}^{(j)})$ for all j
  Pop $\leftarrow$ fittest 30% of Pop $\cup \{\mathbf{u}^{(j)}\}$, plus binary-tournament winners among the rest
end for
return the best genotype in Pop

Table 1: Evaluation-side and generation-side cost accounting. B is the evaluation batch size. The initialization evaluation cost N_pB is shared and omitted.

Method	Evaluation calls	Generation-side LLM calls	Role of LLM generation
Text-space evolution	N_pTB	N_pT	routine per-individual mutation
SLAE-static	N_pTB	0	none in inner-loop search
SLAE	N_pTB	$\mathbb{E}[C_{\text{gen}}] \leq (1 + p_{\text{smo}})T$	sparse representation expansion

4 Experiments

4.1 Experimental Setup

We compare instruction-optimization methods under a fixed evaluator and a matched candidate-evaluation budget. Qwen2.5-7B-Instruct (Yang et al., 2024a) serves as the evaluation model $\mathcal{M}_E$, and all methods are re-run under the same evaluator, task splits, and budget. SLAE also uses Qwen2.5-7B-Instruct as the generator $\mathcal{M}_G$, restricted to SMO and A^2R. We evaluate seven language understanding tasks with accuracy, namely SST-2 (Socher et al., 2013), MR (Pang & Lee, 2005), CR (Hu & Liu, 2004), SST-5 (Socher et al., 2013), AG News (Zhang et al., 2015), TREC (Voorhees & Tice, 2000), and Subj (Pang & Lee, 2004), and two generation tasks, SAMSum summarization (Gliwa et al., 2019) with ROUGE and ASSET simplification (Alva-Manchego et al., 2020) with SARI. These are the nine datasets evaluated by EvoPrompt, which enables a like-for-like comparison with the evolutionary baseline. Evolution uses the development set, and results are reported on the test set. Hyperparameters are listed in Appendix C.2.

4.2 Cost Analysis

We separate evaluation-side prompt evaluations from generation-side LLM calls. Under the matched budget, every population-based method evaluates N_p candidates for T generations on B development examples, so $C_{\text{eval}} = N_pTB$. Text-space evolutionary methods such as EvoPrompt call an LLM once per individual per generation, giving N_pT generation-side calls. SLAE calls $\mathcal{M}_G$ only for expansion: SMO at most once per generation with probability p_{smo}, and A^2R at most once per generation upon stagnation, so that $\mathbb{E}[C_{\text{gen}}] \leq (1 + p_{\text{smo}})T$ (Table 1; derivation in Appendix C.5). With $N_p = 10$, $T = 20$, and $p_{\text{smo}} = 0.3$, text-space evolution makes 200 generation-side calls, whereas SLAE has an expected upper bound of 26 and a worst case of $2T = 40$.

Call counts do not map one-to-one to cost, so we also measured token usage on AG News (Table 4). DSE and A^2R add about 9.6K generation-side tokens per run relative to SLAE-static, 0.106% of the full-run token budget, and in the same runs they add 8.73 accuracy points (Section 4.5). Across all AG News runs of the controlled study, fitness evaluation consumes about 315 times as many tokens as generation. On CoLA (Warstadt et al., 2019), with both methods using Llama-3.1-8B-Instruct (Grattafiori et al., 2024), EvoPrompt used 8.5× as many generation-side tokens as SLAE and 2.9×

as many calls; the token ratio exceeds the call ratio because each text-mutation call embeds several complete prompts in its metaprompt. These ratios concern generation only, since the evaluation workflows of the two implementations differ.

4.3 Main Performance Comparison

Table 2 compares SLAE with baselines that represent different ways of searching the instruction space: human-authored and curated prompts (MI, NI, PromptSource), LLM-driven generation or refinement in raw text (APE, APO), evolutionary search with LLM-based text mutation and crossover (EvoPrompt), and token-level evolution of in-context prompts (PROMPTQUINE). SLAE achieves the best accuracy on every dataset and the best macro average, 85.51 against 83.45 for the strongest baseline, PROMPTQUINE. Against EvoPrompt, the most closely related method, which shares SLAE's evolutionary setting, initial prompt pool, and evaluation budget, the macro-average gain is 3.54 points (85.51 against 81.97). Its gains over both text-evolution methods and PROMPTQUINE, obtained without LLM calls in routine search, indicate that exposing explicit structural and semantic degrees of freedom benefits black-box prompt search. The advantage extends to generation (Table 3), where SLAE leads on all ROUGE metrics for SAMSum and on SARI for ASSET.

4.4 Ablation Studies

The ablation results in Table 2 show that both self-structuring mechanisms contribute to SLAE's final performance. Removing DSE reduces the macro-average accuracy from 85.51% to 83.60%, indicating that scaffold-library expansion is important for exploring new structural regions of the latent action space. Removing $\mathrm{A}^2\mathrm{R}$ yields a smaller but still clear drop to 84.59%, suggesting that slot-level action-space refinement mainly serves as a semantic exploitation mechanism once the current representation begins to stagnate. Table 2 removes the two mechanisms separately; Section 4.5 completes this matrix by disabling both jointly.

Expansion trajectory. Figure 2 shows how the two mechanisms change what genotype-space search can reach. On MR, search first improves within the initial latent action space. When progress stalls, $\mathrm{A}^2\mathrm{R}$ adds slot-level actions, and the accuracy of the best instruction rises from 89.22 at generation 1 to 91.84 at generation 4. After further expansion by DSE and $\mathrm{A}^2\mathrm{R}$, the best instruction uses a newly added scaffold with two labeled demonstrations, which the structural gene g_0 made available to all subsequent search, and reaches 93.41 at generation 8.

4.5 Controlled Study: Attribution, Variation Rule, and Initialization

The main comparison leaves open whether SLAE's gains come from its expansion mechanisms or from two natural alternatives: the human-authored prompts used for initialization, and artifacts of applying integer arithmetic to category indices. Generation-side LLM usage, a third alternative, is addressed in Section 4.2. We test these explanations with self-contained matched studies.

Table 2: Main results on language understanding tasks. We report accuracy (%) averaged over three seeds (5, 10, 15), with standard deviations in parentheses. The best result in each column is bolded. The "Avg." column shows the macro-average accuracy across all seven datasets. SLAE rows use the integer landing rule of Eq. (3).

Method	SST-2	CR	MR	SST-5	AG News	TREC	Subj	Avg.
MI	94.28	90.04	89.05	54.52	75.40	75.14	79.70	79.73
NI	95.44	91.10	91.00	54.24	78.60	76.24	83.60	81.46
PromptSource	93.90	-	-	-	81.20	76.40	-	-
APE	94.53(0.54)	90.58(0.42)	90.51(0.42)	53.55(1.15)	78.54(0.54)	75.28(0.67)	79.88(1.02)	80.41
APO	95.04(0.32)	90.49(0.36)	90.17(0.42)	53.86(1.18)	79.26(0.85)	76.36(0.58)	80.76(1.12)	80.84
EVOPROMPT (GA)	95.64(0.43)	91.26(0.36)	91.35(0.33)	55.62(1.25)	80.63(0.52)	77.05(0.75)	81.63(1.88)	81.88
EVOPROMPT (DE)	95.77(0.54)	91.67(0.32)	91.14(0.45)	55.27(1.68)	80.55(0.42)	77.58(0.87)	81.78(1.64)	81.97
PROMPTQUINE	96.38(0.43)	92.36(0.47)	91.85(0.56)	57.38(0.86)	84.56(0.60)	77.85(0.75)	83.75(0.98)	83.45
SLAE w/o DSE	96.09(0.32)	92.33(0.44)	91.84(0.56)	58.28(0.97)	86.34(0.88)	77.11(1.02)	83.20(0.94)	83.60
SLAE w/o $\mathrm{A}^2\mathrm{R}$	96.54(0.46)	93.14(0.42)	92.74(0.33)	58.94(0.88)	87.48(0.47)	78.48(0.82)	84.82(0.87)	84.59
SLAE (ours)	**96.77**(0.34)	**93.64**(0.25)	**93.41**(0.36)	**59.77**(1.08)	**89.79**(0.46)	**79.54**(0.94)	**85.65**(0.82)	**85.51**

Table 3: Main results on SAMSum dataset (summarization task) and ASSET dataset (simplification task) for Qwen2.5-7B-Instruct. The best result is bolded. SLAE uses the integer landing rule of Eq. (3).

	SAMSum			ASSET
Method	ROUGE-1	ROUGE-2	ROUGE-L	SARI
MI	40.54	15.47	36.42	44.70
APE	40.98(1.25)	15.75(1.10)	35.84(1.36)	45.96(0.84)
EvoPrompt (GA)	41.14(1.31)	16.98(0.85)	39.75(1.25)	47.42(0.76)
EvoPrompt (DE)	42.85(0.48)	17.84(0.98)	39.22(0.71)	47.50(0.63)
SLAE (ours)	**43.45**(1.14)	**18.25**(0.95)	**40.86**(0.86)	**48.54**(0.74)

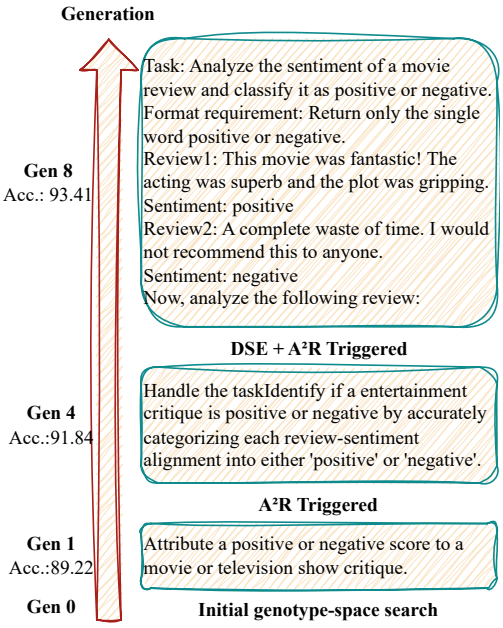


Figure 2: Best instruction during one SLAE run on MR, with the expansion events between the shown generations.

Table 4: Average per-run token usage on AG News (Qwen2.5-7B-Instruct), split into fitness evaluation by $\mathcal{M}_E$ and SDS/DSE/A^{2}R generation by $\mathcal{M}_G$.

	Evaluation		Generation	
Variant	Input	Output	Input	Output
SLAE	7.57M	1.46M	17.1K	8.4K
SLAE-static	7.80M	1.59M	9.6K	6.3K

Table 5: Accuracy (%) of SLAE under different initial scaffold libraries: a single-scaffold Cold Start, an automatically generated IGL, and the human-authored IGL.

Initial library	TREC	SST-5	AG News	Subj
Cold Start	79.15	59.72	89.75	85.66
Automatic IGL	79.23	59.79	89.77	85.62
Human IGL	79.54	59.77	89.79	85.65

Protocol. The AG News studies use Qwen2.5-7B-Instruct, and the held-out CoLA (Warstadt et al., 2019) studies use Llama-3.1-8B-Instruct (Grattafiori et al., 2024). CoLA was not used during method development and receives no task-specific tuning. All studies use the default hyperparameters (Table 8) and three seeds. Search selects on development fitness only, and held-out accuracy is computed afterwards with greedy decoding. Each comparison varies one factor within a matched condition, except the comparison of randomly initialized SLAE with pool-seeded SLAE-static, which changes both initialization and expansion. Comparisons between arms with and without DSE use the main protocol, which matches the main experiments. The other two conditions can affect an arm with DSE for reasons other than the benefit of expansion: the post-processing removes markdown fences and unfilled placeholders, which occur only in DSE-generated scaffolds, and a cap of 32 tokens can cut off the label when a scaffold, such as some generated by DSE, asks for an analysis first. We use these two conditions only to compare arms that both use DSE. Repeated scoring passes are averaged within a search seed before seeds are compared.

Joint ablation. SLAE-static retains the SDS-initialized action sets, the initial scaffold library, prompt-pool initialization, and inner evolutionary search, and disables only DSE and A^{2}R. As shown in Table 6, SLAE exceeds SLAE-static by 8.73 points on AG News. On held-out CoLA, with a

Table 6: Controlled study: mean held-out accuracy differences (points) between variants within matched conditions, averaged over three seeds. DE and LE denote the integer and label-equivariant landing rules (Eqs. 3 and 4); SLAE-static disables DSE and $\mathrm{A}^2\mathrm{R}$. Main protocol: generation cap 128 during search; post-processing: identical scaffold post-processing for all variants; cap 32: generation cap 32 with post-processing.

Comparison	Condition	Mean Δ
SLAE − SLAE-static (DE)	AG News, main protocol	$+8.73$
	CoLA, main protocol	$+5.25$
SLAE − SLAE-static (LE)	AG News, main protocol	$+10.10$
LE − DE (SLAE)	AG News, main protocol	$+2.10$
	AG News, cap 32	$+4.23$
	AG News, post-processing	$+9.13$
LE − DE (SLAE-static)	AG News, main protocol	$+0.73$
Random-init SLAE − SLAE-static	AG News, main protocol	$+5.83$
Random-init SLAE − SLAE	AG News, main protocol	-2.90

different model family and unchanged search hyperparameters, the effect is $+5.25$, a cross-task and cross-family replication of the mechanism effect. Dynamically expanding the scaffold and action spaces therefore accounts for a substantial share of SLAE's gain, although this comparison does not by itself separate adaptive expansion from the larger final search space of full SLAE.

Expansion under either landing rule. On AG News, we crossed the two landing rules with the expansion switch under the main protocol. Expansion improves accuracy by 8.73 points with the integer rule and by 10.10 points with the label-equivariant rule, for every seed with the latter, so in this setting the contribution of self-structuring expansion does not rely on index arithmetic, rounding, or clipping. The label-equivariant rule itself improves mean accuracy over the integer rule in all three AG News conditions with expansion ($+2.10$, $+4.23$, and $+9.13$) and by 0.73 points without it.

Initialization. Removing the entire manual-plus-APE pool and initializing every SLAE individual as a random valid genotype costs 2.90 points relative to default SLAE, yet randomly initialized SLAE still exceeds pool-seeded SLAE-static by 5.83 points. The prompt pool thus improves the starting point but does not explain the advantage of self-structuring expansion. Scaffold-library initialization shows the same pattern (Table 5): a single-scaffold Cold Start, which must build all structural diversity through DSE, an automatically generated IGL, and the human-authored IGL differ by at most 0.39 points on any task and 0.12 points in macro average. $\mathrm{A}^2\mathrm{R}$ further expanded the initial SDS vocabulary 14–17 times per run.

5 Conclusion

We presented SLAE, which casts black-box instruction optimization as search over a structured, integer-valued genotype that decouples a syntactic scaffold from slot-wise semantic actions, enabling inner-loop evolution in genotype space without per-candidate LLM calls. To avoid confining search to a fixed parameterization, SLAE further supports sparse expansion of the discrete latent action space via DSE and $\mathrm{A}^2\mathrm{R}$, allowing the representation to grow as optimization proceeds. Across diverse tasks, our results show consistent improvements over strong baselines under matched evaluation accounting, while providing trajectory transparency through explicit genotype trajectories and inspectable expansion events. Controlled studies show that, under the main protocol, the advantage of self-structuring expansion replicates on a held-out task with a different model family and is not explained by prompt initialization, and, on AG News, that the benefit of expansion persists under a label-equivariant variation rule, which improves on integer-coded DE in every condition compared. Evaluating SLAE with larger backbones and on reasoning, code, and agentic benchmarks is left to future work.

AI USE STATEMENT

In this work, we used generative AI tools for providing feedback on experiments, implementing methods, and assisting with translation. We have not used generative AI tools for generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, cleaning and reformatting datasets, or interpreting results, and supporting qualitative and thematic data analysis is not applicable to this work. We have reviewed all AI-assisted work. AI-generated code was verified and tested for correctness by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Fernando Alva-Manchego, Louis Martin, Antoine Bordes, Carolina Scarton, Benoît Sagot, and Lucia Specia. Asset: A dataset for tuning and evaluation of sentence simplification models with multiple rewriting transformations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4668–4679, 2020.

Stephen Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Févry, et al. Promptsource: An integrated development environment and repository for natural language prompts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 93–104, 2022.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Swagatam Das and Ponnuthurai Nagaratnam Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31, 2011.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3369–3391, 2022.

Agoston E Eiben and James E Smith. *Introduction to evolutionary computing*. Springer, 2015.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 13481–13544, 2024.

Eduardo C. Garrido-Merchán and Daniel Hernández-Lobato. Dealing with categorical and integer-valued variables in Bayesian optimization with Gaussian processes. *Neurocomputing*, 380:20–35, 2020.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *EMNLP-IJCNLP 2019*, pp. 70, 2019.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *The Twelfth International Conference on Learning Representations*, 2024.

Ryoki Hamano, Shota Saito, Masahiro Nomura, Kento Uchida, and Shinichi Shirakawa. CatCMA: Stochastic optimization for mixed-category problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 656–664, 2024.

Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 168–177, 2004.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, Heather Miller, et al. Dspy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*, 2024.

John R. Koza. *Genetic programming - on the programming of computers by means of natural selection*. Complex adaptive systems. MIT Press, 1993. ISBN 978-0-262-11170-6.

Pedro Larrañaga and Jose A. Lozano (eds.). *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, volume 2 of *Genetic Algorithms and Evolutionary Computation*. Springer, 2002.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3470–3487, 2022.

Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain*, pp. 271–278, 2004.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pp. 115–124, 2005.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. GrIPS: Gradient-free, edit-based instruction search for prompting large language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 3845–3864, 2023.

Kenneth V. Price, Rainer M. Storn, and Jouni A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series. Springer, 2005.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7957–7968, 2023.

Kiran Ramnath, Kang Zhou, Sheng Guan, Soumya Smruti Mishra, Xuan Qi, Zhengyuan Shen, Shuai Wang, Sangmin Woo, Sullam Jeoung, Yawei Wang, et al. A systematic survey of automatic prompt optimization techniques. *arXiv preprint arXiv:2502.16923*, 2025.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *The Thirty-Third AAAI Conference on Artificial Intelligence*, pp. 4780–4789, 2019.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. arXiv preprint arXiv:2402.07927, 2025.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations*, 2022.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

Ellen M Voorhees and Dawn M Tice. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 200–207, 2000.

Jianyu Wang, Zhiqiang Hu, and Lidong Bing. Evolving prompts in-context: An open-ended, self-replicating perspective. In *Forty-second International Conference on Machine Learning*, 2025.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024a.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2024b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E. Gonzalez. TEMPERA: Test-time prompt editing via reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023a.

Wenxuan Zhang, Yue Deng, Bing Liu, Sinno Pan, and Lidong Bing. Sentiment analysis in the era of large language models: A reality check. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 3881–3906, 2024.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

Yue Zhang, Leyang Cui, Deng Cai, Xinting Huang, Tao Fang, and Wei Bi. Multi-task instruction tuning of llama for specific scenarios: A preliminary study on writing assistance. *arXiv preprint arXiv:2305.13225*, 2023b.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Forty-first International Conference on Machine Learning*, 2024.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The eleventh international conference on learning representations*, 2022.

A Semantic Diversity Seeding

Algorithm 2 details Semantic Diversity Seeding.

Algorithm 2 Semantic Diversity Seeding (SDS)

Require: Slot type s; Generator LLM $\mathcal{M}_G$; desired action set size k.
Ensure: A diverse action set $\mathcal{A}_s$ of size k.

// Step 1: Candidate action generation
metaprompt $\leftarrow$ design an instruction for $\mathcal{M}_G$ to generate a large pool of candidate actions for slot s, covering diverse semantic categories.
Candidates $\leftarrow \mathcal{M}_G(\text{metaprompt})$
// Step 2: Diversity-aware filtering
Remove exact duplicate candidate actions.
Prompt $\mathcal{M}_G$ to group the remaining candidates into semantic clusters according to their functional role for slot s.
From each cluster, select a compact function-preserving representative action (a within-cluster compactness criterion; diversity comes from covering distinct clusters).
Repeat cluster-level selection until k representative actions are obtained.
$\mathcal{A}_s \leftarrow$ selected representative actions
return $\mathcal{A}_s$

B Instruction Grammar Library

The Instruction Grammar Library (IGL) is a core component of our Dynamic Structural Evolution (DSE) mechanism. It stores the set of syntactic scaffoldings, $\mathcal{G} = \{\mathcal{S}_1, \mathcal{S}_2, \cdots, \mathcal{S}_M\}$. The IGL is initialized with a small, diverse set of human-authored templates and is dynamically expanded during the evolutionary run by the SMO.

Role of the initial IGL. The initial IGL provides a strong, interpretable starting library, analogous to starting prompt optimization from human-written instructions. During search, the structural gene g_0 selects among all scaffolds in the current library and DSE adds new ones; a single-scaffold Cold Start and an automatically generated IGL nearly match the human-authored IGL (Table 5).

To provide a concrete example, Table 7 details the initial Instruction Grammar Library used for the SST-2 task. Each scaffolding contains one or more semantic slots that are subsequently filled by actions from their corresponding action sets. This initial library covers several common prompting strategies, including direct imperatives, role-playing, and question-based formats.

C Experimental Details

C.1 Computing Infrastructure

All experiments were run on servers with two NVIDIA GeForce RTX 5090 or two RTX 4090 GPUs, using Python 3.10, PyTorch, and Hugging Face Transformers. All language model inference for evaluation and generation used standard library functionality without further optimization.

C.2 Hyperparameter Settings

To ensure a fair and reproducible comparison, we detail all hyperparameter settings used in our experiments in Table 8. For all baseline methods, we followed the parameter settings reported in their original papers to the greatest extent possible. For our proposed method, SLAE, key evolutionary parameters such as population size and maximum generations were kept consistent with the primary baseline (EvoPrompt) to ensure a controlled comparison of the underlying search strategies. Other parameters specific to SLAE were determined based on preliminary experiments on a held-out portion of the SST-2 development set and then fixed across all other tasks. The generation parameters for our internal LLM (Qwen2.5-7B-Instruct) were set to encourage creativity while maintaining coherence.

Table 7: The initial IGL for the SST-2 sentiment classification task.

Style	Scaffold Text
Declarative	Please perform {task} task. Given the {input}, assign a {output} from {labels}. Return {output} only {modifier}.
Interrogative	What is the {output} of this {input}? Choose from {labels}.
Role play	As a {task} expert, analyze the {input} and provide the {output} from {labels}.
Step by step	Let's analyze this step by step. First, read the {input} carefully. Next, identify the key indicators. Finally, determine the {output} from {labels} and provide your answer.
Reasoning based	Think carefully about the {input}. Consider the key features and overall characteristics. Based on your analysis, classify it as {labels}. Reasoning: [your analysis]. Answer:
Comparative analysis	Compare this {input} to typical examples of {labels}. Which category does it most closely resemble? Provide your classification with a brief justification.
Professional role	You are a professional {task} analyst. Your expertise is required to evaluate this {input}. Apply your knowledge and experience to determine the appropriate {output} from the available options: {labels}.
Detailed analysis	Perform a detailed {task} analysis. Input text: {input}. Break down the key elements, assess the overall characteristics, and conclude with your {output} selection from: {labels}. Format: Analysis: [details] | Conclusion:
Question-driven	Consider these questions about the {input}: What are the key characteristics? What category does it belong to? How would experts classify it? Based on your answers, select the most appropriate {output} from {label}.
Structured format	=== {task} ANALYSIS === Text: {input} Task: Determine {output} Options: {labels} Analysis: [Provide reasoning] Result:
Emotional intelligence	Using emotional intelligence to understand this {input}: What feelings does it evoke? What emotions does the author express? Emotional assessment leads to {output} classification: {labels}. Answer:
Tone analysis	Analyze the tone and mood of this {input}. Consider word choice, phrasing, and emotional undertones. The {output} classification from {labels} is:
Creative metaphor	Imagine you're a {task} detective. Your mission: decode the true nature of this {input}. What verdict do you deliver from {labels}?
Dialogue format	Human: Please analyze this {input} for {task}. AI: I'll examine it carefully. Human: What's your assessment? AI: Based on my analysis, the {output} is clearly one of {labels}. My answer:
Minimalist	{input} $\rightarrow$ {labels}
Forward prediction	Given the {input}, predict the most likely real-world action the author will take next (e.g., 'will recommend this to friends', 'will demand a refund', 'will probably forget about it tomorrow'). Based on this predicted consequence, what is the underlying sentiment? Classify it using {labels}.
Abstract mapping	If the emotional tone of the {input} could be represented as a color, what color would it be and why? Consider attributes like brightness, saturation, and temperature in your explanation. After explaining the color mapping, provide the formal sentiment classification from {labels}.

Table 8: Hyperparameter settings for our experiments.

Hyperparameter	Description	Default Value
Popsize	Number of individuals (candidate instructions) in the population.	10
Budget	Maximum number of evolutionary generations to run.	20
Perturbation rate	The per-gene probability of uniform resampling, $0.05 + 0.15\,(1 - \bar{H})$, where $\bar{H}$ is the mean pairwise normalized Hamming distance of the population.	0.05–0.2
Crossover rate	The probability of inheriting genes from the mutant vector in DE's crossover operator.	0.7
Elite fraction	The fraction of the pooled parents and offspring that survives by fitness before tournament selection.	0.3
Tournament size	The number of candidates compared in each tournament that fills the remaining slots.	2
Scaling factor	The factor (F) of the integer landing rule (Eq. 3).	0.5
Batch size	The number of samples from the development set used to evaluate the fitness of each instruction in a single generation.	32
Action set size	The initial number of candidate actions for each semantic slot, generated by the SDS method.	12
Evolution temperature	The sampling temperature for the Generator LLM $(\mathcal{M}_G)$ during creative operations (DSE/$\mathrm{A}^2\mathrm{R}$) to control the diversity of generated outputs.	0.7
Evaluation temperature	The sampling temperature for the Evaluation LLM $(\mathcal{M}_E)$ during fitness calculation.	0.1
DSE trigger	The generation-level probability p_{smo} of activating SMO to add at most one new scaffold to the scaffold library before routine offspring generation.	0.3
Max slots	The maximum number of semantic slots (N) allowed in an instruction's syntactic scaffolding.	10

C.3 BASELINES

We compare SLAE against a strong set of baselines representing different paradigms of prompt engineering:

- Human-Authored Prompts: This category includes two types of strong baselines. (1) Manual Instruction (MI): High-quality, task-specific prompts crafted by experts and sourced from established works (Sanh et al., 2022; Zhang et al., 2023b; 2024). (2) Curated Repositories: Prompts drawn from large-scale collections of human-composed instructions, such as PromptSource (Bach et al., 2022) and Natural Instructions (Mishra et al., 2022).
- APE (Zhou et al., 2022): An influential method that employs an iterative Monte Carlo Search strategy. It embodies an exploration-focused "generate-and-test" paradigm.
- APO (Pryzant et al., 2023): In contrast to APE, APO emphasizes exploitation. It leverages incorrectly predicted instances to form a "pseudo-gradient," which is then used to iteratively refine an initial high-quality prompt.
- EvoPrompt (Guo et al., 2024): The primary state-of-the-art evolutionary method. It directly employs a powerful LLM as an opaque, black-box operator for textual mutation and crossover. To ensure a comprehensive comparison, we benchmark against its two most powerful reported variants: one driven by a genetic algorithm (GA) and another by DE.
- PROMPTQUINE (Wang et al., 2025): An ICL-focused evolutionary framework that views prompts as self-replicating programs and performs open-ended search over token-level pruning/mutation of in-context demonstrations, often yielding compact (and sometimes non-human-readable) contexts that improve downstream performance under a fixed evaluation accounting.

Table 9: Overview of the datasets used in our experiments, showing the label space and the size of the test set.

Dataset	Label	Texts
SST-2	{positive, negative}	1821
CR	{positive, negative}	2000
MR	{positive, negative}	2000
SST-5	{terrible, bad, okay, good, great}	2210
AG News	{World, Sports, Business, Tech}	7600
TREC	{Description, Entity, Expression, Human, Location, Number}	500
Subj	{subjective, objective}	2000
SAMSum	-	819
ASSET	-	359

C.4 DATASETS

All datasets used in our experiments are standard, publicly available benchmarks. In Table 9, the statistics of the text classification, simplification, and summarization datasets are displayed. Our evolutionary framework contains stochastic components, primarily in the population initialization and the selection of parents for mutation in DE. To ensure the robustness and reproducibility of our findings, all reported results are the average of experiments conducted across three distinct random seeds: 5, 10, and 15.

C.5 COST ANALYSIS

This appendix provides the derivation underlying the cost accounting summarized in Table 1. We separate optimization cost into evaluation-side prompt evaluations and generation-side LLM calls. The main experiments match methods by evaluation-side budget, while generation-side calls quantify how often an LLM is used to propose candidate instructions or expand the search representation.

Evaluation-side cost. Let N_p be the population size, T the number of optimization generations, and B the number of development examples used in one fitness evaluation. During the optimization loop, each population-based method evaluates N_p candidate instructions per generation, giving

$$C_{\text{eval}} = N_p T B.$$

This matches the evaluation-side budget used in the main experiments. We omit the common initialization evaluation cost $N_p B$, which is shared by population-based methods and does not affect the comparison of optimization-stage search overhead.

Generation-side cost of text-space evolution. Text-space evolutionary methods, such as EvoPrompt-style mutation or crossover, use an LLM as a routine candidate-level proposal operator. Under the population-based accounting used in our experiments, this requires one generation-side LLM call per individual per generation:

$$C_{\text{gen}}^{\text{text}} = N_p T.$$

The corresponding optimization-stage request count is therefore

$$C_{\text{total}}^{\text{text}} = C_{\text{eval}} + C_{\text{gen}}^{\text{text}} = N_p T B + N_p T.$$

Generation-side cost of SLAE. SLAE does not use the generator model $\mathcal{M}_G$ for routine per-individual mutation or crossover. Its inner-loop offspring generation is performed by categorical variation on genotypes. Thus, the static variant without representation-space expansion has

$$C_{\text{gen}}^{\text{static}} = 0.$$

For full SLAE, generation-side LLM calls are used only for representation-space expansion. $\mathrm{A}^2\mathrm{R}$ is triggered when $\mathrm{IsStagnant}(\mathrm{history}, k_{\text{stag}})$ holds. Each $\mathrm{A}^2\mathrm{R}$ event calls $\mathcal{M}_G$ once to propose new

actions for the selected slot. Let $N_{\text{stag}} \leq T$ denote the number of stagnation-triggered refinement events. DSE/SMO is triggered at most once per generation with probability p_{smo}. Each SMO event calls $\mathcal{M}_G$ once to propose a new scaffold. Therefore,

$$\mathbb{E}[C_{\text{DSE}}] = p_{\text{smo}} T, \qquad C_{\text{A}^2\text{R}} = N_{\text{stag}} \leq T.$$

The expected generation-side cost of full SLAE is

$$\mathbb{E}[C_{\text{gen}}^{\text{SLAE}}] = p_{\text{smo}} T + \mathbb{E}[N_{\text{stag}}] \leq (1 + p_{\text{smo}}) T.$$

The corresponding expected optimization-stage request count is

$$\mathbb{E}[C_{\text{total}}^{\text{SLAE}}] = N_p T B + p_{\text{smo}} T + \mathbb{E}[N_{\text{stag}}].$$

The deterministic worst-case generation-side bound is

$$C_{\text{gen}}^{\text{SLAE}} \leq 2T,$$

which corresponds to both DSE and A^2R being triggered in every generation.

Default-budget example. With the default setting $N_p = 10$, $T = 20$, and $p_{\text{smo}} = 0.3$, a text-space evolutionary method using one LLM mutation call per individual per generation requires

$$N_p T = 200$$

generation-side LLM calls. In contrast, SLAE has an expected generation-side upper bound of

$$(1 + p_{\text{smo}}) T = 26$$

expansion calls, and a deterministic worst-case bound of

$$2T = 40.$$

Thus, under the same evaluation-side budget $N_p T B$, SLAE reduces generation-side scaling from $\mathcal{O}(N_p T)$ to $\mathcal{O}(T)$ in expectation.

Initialization cost. The analysis above focuses on optimization-stage calls. Initialization calls, such as creating the initial scaffold library or generating initial action candidates, occur before the evolutionary loop. They are implementation-dependent and are not part of the matched budget.

Measured token usage. The AG News measurements are reported in Table 4. On CoLA, SLAE used about 26K generation-side tokens in 69 calls and EvoPrompt about 219K tokens in 201 calls.

C.6 Controlled Study Details

Condition selection. Comparisons between arms with and without DSE are reported under the main protocol, for the reasons given in Section 4.5. Where post-processing was applied, it never changed a SLAE-static instruction.

Population initialization. Every default initial individual is encoded from the shared prompt pool. In the random-initialization variant, the entire pool, including both manual and APE-generated prompts, is removed and every SLAE individual is initialized as a random valid genotype.