

DISCOVERING NOVEL LLM EXPERTS VIA TASK-CAPABILITY COEVOLUTION

005 **Anonymous authors**

006 Paper under double-blind review

ABSTRACT

011 Frontier model developers aim to train models continually to possess emergent, di-
 012 verse capabilities. To extend capabilities, the current pre-training and post-training
 013 paradigm requires manually starting training runs with static datasets or reward
 014 functions every time. Addressing this limitation, our work pursues the insight that
 015 open-endedness (via the coevolution of models and tasks) can discover models
 016 with increasingly novel skills in a single run. We introduce a new model devel-
 017 opment framework that extends coevolution to large language model (LLM) dis-
 018 covery, open-ended *Assessment Coevolving with Diverse Capabilities* (AC/DC).
 019 AC/DC evolves both LLMs via model merging and natural language tasks via syn-
 020 synthetic data generation. AC/DC discovers growing archives of LLMs that surpass
 021 the capabilities of larger LLMs while taking up less GPU memory. In particular,
 022 our LLM populations achieve a broader Coverage of expertise than other curated
 023 models or baselines on downstream benchmarks, without *any* explicit benchmark
 024 optimization. Furthermore, AC/DC improves Coverage over time, continually in-
 025 novates on tasks and models, and improves performance in multi-agent best-of-N
 026 selection. Our findings highlight the potential of coevolution as a means of dis-
 027 covering broader sets of capabilities from base LLMs. Overall, AC/DC brings us
 028 one step closer to a profoundly new paradigm of LLM development, where con-
 029 tinual improvements to the diversity of model capabilities can be accelerated by
 030 leveraging existing models as stepping stones to increasingly powerful models.

1 INTRODUCTION

035 LLMs and foundation models (Brown et al., 2020; Bommasani et al., 2021; Kaddour et al., 2023)
 036 underpin key advances in AI for open-ended discovery and innovation (Nguyen et al., 2016; Lehman
 037 et al., 2023; Zhang et al., 2023; Lu et al., 2024). Such innovation capacity in future AI systems,
 038 innate to human civilization, would not only have profound implications for automated scientific
 039 discovery, but would also accelerate AI research itself. How do we get closer to LLMs as engines
 040 of knowledge accumulation and serendipitous discovery, with the ability to stumble upon greatness
 041 (Stanley & Lehman, 2015) and drive paradigm shifts (e.g., the Transformer (Vaswani et al., 2017))?
 042 Additionally, how do we imbue LLMs with innovation capacity and broader capabilities, given the
 043 prohibitively expensive costs and inaccessibility of running bigger and bigger models (Pan & Wang,
 044 2025) or obtaining more data (Muennighoff et al., 2023), especially for typical ML researchers?

045 While excitement grows around LLMs for scientific discovery (Romera-Paredes et al., 2023;
 046 Novikov et al., 2025), the current paradigm of LLM development struggles to keep up with the
 047 accumulation of knowledge on learnable or discovered data. Developers must continually adapt to
 048 incremental improvements in static datasets (Albalak et al., 2024; Kandpal et al., 2025), environ-
 049 ments (Intellect, 2025; Lambert et al., 2024; Novikov et al., 2025), learning algorithms (Shao et al.,
 050 2024; Liu et al., 2025b), and architectures (Yang et al., 2025; Muennighoff et al., 2024), to push the
 051 boundaries of frontier models. Continually training on synthetic data (Wang et al., 2023d; Xu et al.,
 052 2023; Maini et al., 2024; Havrilla et al., 2024) and broad-domain reward signals (Zhao et al., 2025)
 053 brings us closer to self-improving LLMs. Still, only one model is produced at a time. Trusting
 a single big static model to solve all real-world problems would therefore be challenging, due to
 concerns about fractured entangled representations (Kumar et al., 2025) and costs (Li et al., 2025).

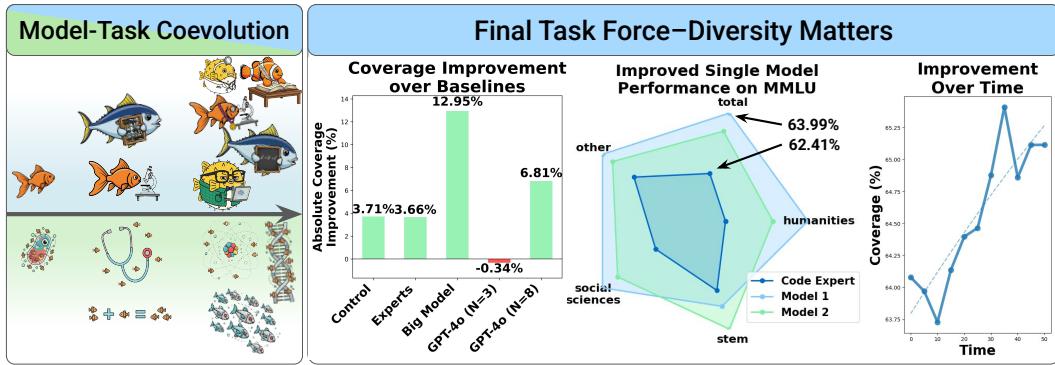


Figure 1: **Method Overview.** AC/DC coevolves an increasing set of diverse LLMs alongside an increasingly diverse and complex set of tasks, measuring the discovered models’ capabilities. Our discovered collective of models covers more skills than baselines across a wide range of benchmarks. Moreover, AC/DC discovers improved single model performance (as seen by MMLU (Hendrycks et al., 2021a) performance) and demonstrates improvement over time (shown as an average of MMLU and MMLU Pro (Wang et al., 2024) accuracy).

In contrast to individual models, collective intelligence (CI) (exemplified by human civilization) is capable of endlessly achieving feats far greater than any single human could (Mitchell, 2009). CI has even inspired new paradigms in AI (Ha & Tang, 2022) and multi-agent LLM systems (Liang et al., 2023; Inoue et al., 2025), making them more robust during test-time scaling. By discovering whole collectives of small and accessible LLMs with diverse capabilities, we can overcome the limitations and weaknesses of any single trained model or the need to train different models separately.

To overcome the challenge of CI discovery, *open-endedness* (OE) is an emerging paradigm aiming for never-ending discovery via open-ended algorithms (Stanley et al., 2017). Pursuing such AI-generating algorithms (Clune, 2020), open-ended coevolution takes inspiration from the creativity explosion of natural evolution and human innovation, and pursues ever-changing learning environments for populations of increasingly intelligent agents to gain diverse capabilities (Wang et al., 2019; Dennis et al., 2020). Leveraging recent advances in OE through LLMs (Faldor et al., 2024; Aki et al., 2024; Nisioti et al., 2024), we introduce a new framework to discover a whole population of expert LLMs through open-ended *Assessment Coevolving with (/\w) Diverse Capabilities* (AC/DC). AC/DC combines evolutionary model merging (Akiba et al., 2025) and synthetic data generation (Lu et al., 2025) to enable LLM populations to continually adapt to novel challenges that are generated, while satisfying minimal criteria for model and task quality (Brant & Stanley, 2017), all without explicit benchmark optimization (Lehman & Stanley, 2011a).

Following coevolution via AC/DC, we discovered a broad array of LLMs with diverse expertise and response styles that solve synthetic tasks spanning engineering, the sciences, and creative writing. When we selected a fixed-size subset of LLMs that make up the broadest skill coverage on synthetic tasks, and then evaluated them on various LLM benchmarks that are out-of-distribution (OOD) to synthetic tasks at test time, we found that our population of smaller evolved LLMs (with a combined lower LLM parameter count than compared baseline models) was able to solve and cover more benchmark tasks than bigger LLMs of the same model family, as well as the initial seed LLM population. Our results also suggest that our LLM collectives surpass or reach GPT-4o (Hurst et al., 2024) levels of knowledge covered with a significantly lower proportion of collective model parameters. Furthermore, a single evolved model achieved better MMLU (Hendrycks et al., 2021b) performance than the best starting seed model (cf. Fig. 1), more iterations of coevolution led to continually improved model population performance at test time, and cooperative final answer (Best-of-N) selection was more often successful with our LLM collectives than with baselines.

In short, our main contributions are (1) the AC/DC method applying coevolution to a novel joint LLM-and-synthetic-data discovery framework, (2) a demonstration of autonomous discovery of diverse LLM experts solving OOD tasks more broadly than baseline methods (some directly optimizing for benchmarks) and off-the-shelf models, and (3) evidence of a path to open-ended improvement of LLMs without explicit benchmark optimization, through an analysis of AC/DC.

108
109

2 BACKGROUND AND PRELIMINARIES

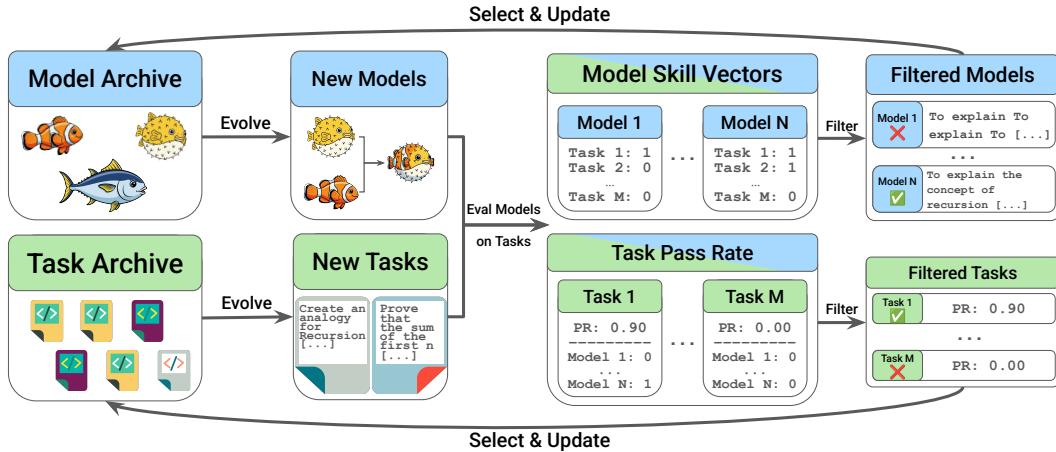
110
111
112
This section introduces key concepts central to AC/DC: model merging operations that enable evo-
lutionary discovery of LLM populations, Coverage metrics that quantify collective capabilities, and
quality-diversity (QD) principles that guide our coevolutionary process.113
114
115
116
117
Evolutionary Model Merging. Model merging combines multiple existing LLMs to produce new
models with lower resource requirements than training from scratch (Wortsman et al., 2022; Il-
harco et al., 2023). Akiba et al. (2025) introduced evolutionary model merge (EvoMerge), which
automates the merging process through evolutionary optimization using CMA-ES (Hansen & Os-
termeier, 2001). Building on this foundation, we employ two key evolutionary operations:118
119
120
121
Crossover: We sample two parent LLMs randomly and merge them using weighted linear interpo-
lation of their task vectors, following Kuroki et al. (2025). The task vector $\tau_{pi} = \theta_{parent_i} - \theta_{base}$
represents the difference between parent LLM i and a base LLM (see Appendix Sec. M for more
details).122
123
124
125
126
127
Mutation: We generalize existing mutation operations by applying noise to the singular values of
weight matrices in merged LLMs. For each weight matrix W , we compute its singular value decom-
position $W = U\Sigma V^T$ and apply perturbations to the first k singular values in Σ before reconstruc-
tion, loosely inspired by Sun et al. (2025). This approach modifies the representational structure
while preserving the overall weight matrix geometry (see Appendix Sec. M for more details).128
129
Coverage Metric. Coverage measures the collective problem-solving capacity of LLM populations.
Given Q total number of questions and N number of LLMs, Coverage is:

130
131
132
$$\text{Coverage} = \frac{1}{Q} \sum_{q=1}^Q \left(\bigvee_{i=1}^N (x_{q,i} = y_q) \right) \quad (1)$$

133
134
135
136
137
where $x_{q,i}$ is the output of LLM i for question q , y_q is the ground truth answer for question q ,
and $\bigvee_{i=1}^N$ denotes the logical OR operation over all N LLMs. Coverage quantifies whether at least
one LLM in the population solves each problem, capturing the collective intelligence potential of
diverse LLM ensembles. Unlike individual LLM accuracy, Coverage emphasizes complementary
capabilities that emerge from LLM diversity.138
139
140
141
142
Skill Vectors. We represent LLM capabilities through binary skill vectors, where each indexed
element indicates task completion status. They serve as behavioral signatures that enable direct
comparison of LLMs without predefining niches (as in MAP-Elites (Mouret & Clune, 2015b)). The
distance between skill vectors informs the diversity of complementary LLM capabilities.143
144
145
146
147
148
Quality-Diversity (QD). QD generates collections of diverse, high-quality solutions (Pugh et al.,
2016; Lehman & Stanley, 2011b), unlike traditional optimization, which seeks a single optimal
solution. For model selection, we apply Dominated Novelty Search (DNS) (Bahlous-Boldi et al.,
2025), a recent QD algorithm particularly suited to skill vector representations (similar to Meyerson
& Miikkulainen (2017)). DNS computes local competition fitness \tilde{f}_i by measuring each solution's
distance from better-performing solutions in the descriptor space. For solution i , \tilde{f}_i is computed as:

149
150
151
$$\tilde{f}_i = \begin{cases} \frac{1}{k} \sum_{j \in \mathcal{K}_i} d_{i,j} & \text{if } |\mathcal{D}_i| > 0 \\ +\infty & \text{otherwise} \end{cases} \quad (2)$$

152
153
154
where \mathcal{D}_i contains solutions fitter than solution i , \mathcal{K}_i contains indices of k solutions in \mathcal{D}_i with small-
est distances $d_{i,j}$ between solutions i and j . Local competition encourages diversity by rewarding
solutions that are distant from higher-performing neighbors in the behavioral space.155
156
157
158
159
160
161
Open-Ended Coevolution. Brant & Stanley (2017; 2020) demonstrate that defining minimal cri-
teria (MC) for both agents and environments enables more open-ended outcomes in coevolution,
filtering out undesired outcomes while enabling exploration to flourish. AC/DC coevolves popula-
tions of LLMs and synthetic tasks, where models must satisfy quality thresholds while maximizing
quality and behavioral diversity through their skill vector representations. This creates a dynamic en-
vironment where increasingly sophisticated capabilities can emerge through the interplay between
model evolution and task complexity. Related work discussion in Sec. G highlights AC/DC as a
novel system combining concepts from various fields.

162 **3 AC/DC: ASSESSMENT COEVOLVING WITH DIVERSE CAPABILITIES**
163

179 **Figure 2: Algorithm Overview.** AC/DC continuously coevolves a model (LLM) and a synthetic
180 task archive. LLMs are evolved using model merging crossover, and weight noising as a mutation
181 operation. Tasks are evolved using a large scientist LLM to adapt the existing tasks, generating
182 increasingly novel and complex tasks. Models are evaluated on this data. We then compute a skill
183 vector (i.e., signature of quality and diversity) for each model and a pass rate for each task.
184 Based on those, we first apply minimal criterion (MC) filters (gibberish LLM filter, impossible task filter)
185 and then select the models and tasks to update the archives, respectively.

187 We describe an open-ended algorithm for automatically discovering diverse LLMs that can collect-
188 tively cover a wide range of skills. AC/DC coevolves two archives: an LLM archive \mathcal{A}_M optimized
189 for quality and diversity, and a synthetic active task archive \mathcal{A}_Q containing an increasingly complex
190 and novel set of challenges that the agent archive must address. We illustrate the algorithm in Fig. 2
191 and provide the core algorithmic steps below (Algorithm 1). Inspired by Brant & Stanley (2017), we
192 highlight minimal criteria (MC) for both LLMs and tasks. [For further details, see Appendix Sec. N.](#)

Algorithm 1 AC/DC: Assessment Coevolving with Diverse Capabilities

```

1: Initialize: Model archive  $\mathcal{A}_M \leftarrow$  seed and init models, Task archive  $\mathcal{A}_Q \leftarrow$  seed and init tasks
2: for  $g = 1$  to  $G$  do ▷ Model Evolution Phase
3:    $P \leftarrow \text{SELECTPARENTS}(\mathcal{A}_M)$  ▷ Select  $P$  parents
4:    $O \leftarrow \text{CROSSOVERMUTATE}(P, N)$  ▷ Generate  $N$  offspring
5:    $E \leftarrow \text{EVALUATE}(O, \mathcal{A}_Q)$  ▷ Evaluate skill vectors, get responses to tasks; Sec. F.2
6:    $T \leftarrow \text{GIBBERISHFILTER}(E)$  ▷ Trim away/discard degenerate models; Sec. F.5
7:    $\mathcal{A}_M \leftarrow \text{DNSUPDATE}(\mathcal{A}_M, T)$  ▷ Select top  $M$  models; Sec. A.3.1
8:   if  $g \bmod G_{task} = 0$  then ▷ Task Evolution Phase
9:      $Q \leftarrow \text{GENERATETASKS}(\text{scientist LLM})$  ▷ Generate  $N_{gen}$  tasks; Sec. F.1
10:     $Q_{nov} \leftarrow \text{NOVELTYFILTER}(Q)$  ▷ Remove similar tasks; Sec. F.4
11:     $Q_{valid} \leftarrow \text{VALIDATETASKS}(Q_{nov})$  ▷ Reflection & validation; Sec. F.1
12:     $\mathcal{A}_Q, \mathcal{A}_{Q_g} \leftarrow \text{UPDATETASKARCHIVE}(\mathcal{A}_Q, \mathcal{A}_{Q_g}, Q_{valid})$  ▷  $\mathcal{A}_{Q_g}$  is global task archive
13:     $\text{REEVALUATEARCHIVE}(\mathcal{A}_M, Q_{valid})$  ▷ Update skill vectors with new task pool
14:   end if
15:    $\text{SAVEARCHIVES}(\mathcal{A}_M, \mathcal{A}_Q, g)$ 
16: end for
17: return  $\text{SELECTTASKFORCE}(\mathcal{A}_M, \mathcal{A}_{Q_g})$  ▷ Select  $N_{tf}$  diverse models
  
```

212 **Model Archive Evolution.** Evolution begins with at least three seed LLMs, representing fine-tuned
213 versions of the same base architecture. We maintain M active LLMs, i.e., LLMs considered as
214 parents for the next generation (as in DNS). We also save a historical archive of LLMs every G_{task}
215 generations (before task adaptation), as candidates for a future task force. Given the existing parent
population in M , we apply crossover and mutation operators as described in Sec. 2, producing

216 N offspring and yielding $M + N$ candidates for evaluation (Lines 3-4). Each candidate LLM is
 217 evaluated on our synthetic task archive and assigned skill vectors (Line 5). We apply a novel MC
 218 filter called the “gibberish filter” to eliminate degenerate models by sampling outputs from the LLM
 219 for three random tasks each and employing a judge LLM to assess text coherence (Line 6). For the
 220 remaining models, we employ DNS to compute adjusted local competition scores \tilde{f} (cf. Sec. A.3.1).
 221 We retain the highest-fitness model and select the remaining $P - 1$ models with top \tilde{f} scores such
 222 that we have at most M active models (Line 7).

223 **Task Archive Evolution.** Alongside LLM agents, we coevolve an increasingly challenging and
 224 diverse set of tasks. We employ a large scientist LLM to synthesize tasks in line with METR Task
 225 Standard Team (2024) (but simplified), where each task comprises a question-answer pair with an
 226 accompanying scoring function defined in Python (Line 9; Sec. F.1). We extend Lu et al. (2025) with
 227 a code extraction tool that enables robust evaluation of tasks requiring code generation, allowing the
 228 scientist LLM to programmatically parse and execute subject model responses. We maintain at
 229 most Q_{max} active tasks per generation that are used to evaluate the current generation of models,
 230 alongside a global task archive updated every G_{task} generations. We employ two vector databases
 231 for efficient similarity search: one for active tasks and another for the global archive.

232 Task evolution begins with N_{seed} manually curated seed tasks (cf. Sec. E.3) and generates N_{init}
 233 initial tasks through our evolution pipeline (Lines 9-13): (1) *Task Proposal Phase* samples a parent
 234 task and three random tasks from the active task database. Based on the parent task’s difficulty pro-
 235 file—determined by average pass rates across the current model population—we classify adaptation
 236 requirements as: increase difficulty, decrease difficulty, or generate a novel variant. The scientist
 237 LLM receives the parent task, three random reference tasks, and an adaptation type to generate a
 238 candidate task. (2) *Novelty Filtering* retrieves the three most similar tasks from the global archive
 239 using cosine similarity in embedding space. A judge LLM then determines whether the proposed
 240 task introduces sufficient novelty relative to existing tasks. (3) *Reflection and Validation* applies
 241 iterative refinement through self-evaluation cycles where the scientist LLM first attempts to solve its
 242 own generated task, and we execute the scoring function to identify implementation issues. Compi-
 243 lation errors trigger automatic correction with error feedback, while logic-based errors prompt task
 244 refinement. (4) *Quality Assurance and MC* implements additional filtering to remove impossible
 245 tasks that no LLM was able to solve, replacing them with their parent tasks. Accepted tasks are
 246 added to both the global archive and the active tasks. See Sec. E.4 for generated example tasks.

247 **Model Selection for Downstream Evaluation.** After coevolution over multiple generations, we
 248 select N_{tf} models for our *task force* that maximize the number of correctly solved tasks across
 249 our synthetic task distribution \mathcal{A}_{Q_g} (Line 17) (see Sec. D.2 for experiments with different selection
 250 strategies). This selection process operates independently of downstream benchmarks, avoiding
 251 optimization pressure and maintaining model generalization for OOD domains.

252 4 QUANTITATIVE RESULTS

254 We compare our task force Coverage (Eq. (1)) against several baseline approaches. We evaluate on
 255 a diverse set of benchmarks covering general knowledge, math, STEM, and code (see Sec. A.4.1 for
 256 details). See Sec. A.2 for model specifications and Sec. A.1 for hyperparameters.

257 **Baselines Setup.** We compare against four baselines: (1) *Experts (N=3)*: Hand-selected instruct
 258 models (code, math, general) prompted once each with temperature 0 (for a discussion on N=8
 259 experts, see Sec. D.4). (2) *Control (N=3/8)*: The general instruct model prompted 3 or 8 times with
 260 temperature 0.7. (3) *Big Model*: A single large instruct model prompted once with temperature 0.
 261 (4) *GPT-4o*: Prompted once as the Big Model.

263 **Best-of-N Selection Setup.** Next to Coverage, we also evaluate Best-of-N (BoN) single-answer
 264 selection from multiple candidates using standard benchmark versions, testing whether Coverage
 265 improvements translate to practical deployment scenarios. We implement three techniques for the
 266 three benchmark types (MCQ, math, code). For further details, see Sec. A.4.

267 **Coverage.** Tab. 1 presents Coverage results across five base model families (see Sec. B.1 for details),
 268 revealing important patterns in AC/DC’s performance across different architectures and scales. Our
 269 evolutionary approach demonstrates broad applicability, achieving positive improvements on aver-
 age across all model families and configurations (+1.99% to +10.43% across comparisons). The

270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323

Table 1: **AC/DC’s Coverage improvement** across different models. Results show average performance improvement across all benchmarks for N=3 and N=8 configurations over the respective baseline. [For analysis on statistical significance, see Sec. K.](#)

Base Model	vs Experts N=3 (%)	vs Control (%)		vs Big Model (%)		vs GPT-4o (%)	
		N=3	N=8	N=3	N=8	N=3	N=8
Qwen2 7B	+2.28	+0.73	+0.38	+5.57	+14.59	-6.01	+3.01
Qwen2.5 7B	+3.66	+2.61	+3.71	+5.80	+12.95	-0.34	+6.81
Qwen3 14B	-0.66	+0.30	+1.86	+5.53	+11.19	+1.67	+7.33
DeepSeek V1 7B	+8.71	+6.30	+4.35	+2.32	+13.44	-21.10	-9.98
Average	+2.80	+1.99	+2.06	+3.85	+10.43	-5.15	+7.17

Qwen 2, Qwen 2.5, and DeepSeek models show consistent improvements across expert and control baselines, indicating effective discovery of complementary capabilities. Qwen3 14B exhibits scaling-dependent behavior where N=3 configurations under-perform expert baselines but demonstrate improvement over the control baselines.

Our approach also achieves substantial parameter efficiency—for example, Qwen2 7B achieves 5.57% improvement over a 72B model using only 29% of the parameters at N=3, growing to 14.59% improvement at N=8, suggesting that distributed specialization benefits compound with scale. Our results show that AC/DC successfully discovers complementary capabilities that extend beyond what can be achieved through either manual expert selection or parameter scaling.

Finally, comparing our task force to GPT-4o, we demonstrate that our N=8 collective of models achieve broader Coverage. This is especially interesting considering that our task forces require very little compute to merge and then serve them in consideration of the potential costs of GPT-4o. Moreover, at N=3, our Qwen 2.5 task force of 3 7B models approach the capabilities of GPT-4o. This finding suggests that a collective of smaller, diverse, and capable models possesses the knowledge of a single frontier model, which can be leveraged given advances BoN selection methods.

Table 2: **AC/DC’s Best-of-N improvement** across different models. Results show average performance improvement across all benchmarks for N=3 and N=8 configurations over the respective baseline. [For analysis on statistical significance, see Sec. K.](#)

Base Model	vs Experts N=3 (%)	vs Control (%)		vs Big Model (%)		vs GPT-4o (%)	
		N=3	N=8	N=3	N=8	N=3	N=8
Qwen2 7B	+3.58	+0.89	+1.18	-0.36	+3.76	-11.82	-7.71
Qwen2.5 7B	+1.38	-2.01	+1.89	-1.82	+1.92	-8.22	-4.48
Qwen3 14B	+0.34	+1.59	-0.18	+0.77	+2.59	-6.70	-4.88
DeepSeek V1 7B	+7.43	+0.13	+4.86	+0.45	+5.30	-22.12	-17.26
Average	+2.55	+0.12	+1.55	-0.19	+2.71	-9.77	-6.86

Best-of-N (BoN). Tab. 2 presents BoN selection results across five base model families (see Sec. B.2 for details), revealing how Coverage improvements translate into practical single-answer scenarios. Our evolutionary approach maintains strong performance when restricted to best-of-N, achieving positive improvements on average across representative comparisons (+0.12% to +2.55% vs control and experts). Most base model groups show positive improvements over control baselines, with particularly strong performance from Qwen2 7B, Qwen2.5 7B, and DeepSeek V1 7B. They show weaker absolute performance than Qwen3 14B, thus having more room for performance gains.

Most notably, our approach achieves exceptional parameter efficiency against big models. For example, Qwen2 7B reaches within 0.36% of the 72B model’s performance using only 29% of the parameters at N=3, and surpasses it by 3.76% at N=8 while using 22% fewer parameters. This pattern extends across model families—our N=8 configurations consistently outperform much larger models while using substantially fewer parameters.

Comparing against GPT-4o, we observe that our 8 Qwen2.5 7B and Qwen3 14B models come close to GPT-4o’s performance, indicating that with improved BoN methods, the collective of smaller models is within reach of outperforming the significantly larger proprietary model.

324 These results demonstrate that our evolutionary process successfully translates Coverage gains into
 325 practical single-answer scenarios, achieving near-parity or superiority compared to much larger
 326 models while maintaining advantages over specialized baselines across scales.

327 **Ablations.** We examine the contribution of individual algorithmic components by systematically re-
 328 moving each from our evolutionary process (detailed results in Sec. D.1). The ablation reveals that
 329 quality-diversity selection and the gibberish filter are the most critical components, with their re-
 330 moval causing the largest absolute performance drops (1.86% and 2.12% at N=3, 0.80% and 0.68%
 331 at N=8, respectively). Removing individual components like mutation or novelty filtering causes
 332 modest decreases ranging from 0.46%-0.67% at N=3 and 0.35%-0.61% at N=8. Overall, larger
 333 populations appear to provide higher algorithmic robustness against individual components.

334 Most importantly, removing all evolutionary components simultaneously causes substantial perfor-
 335 mance degradation (1.81% drop at N=3, 6.94% drop at N=8), demonstrating that our algorithmic
 336 innovations work synergistically to enable effective collective intelligence, with benefits becoming
 337 more pronounced at scale. **In Sec. D.6, we demonstrate that including coevolution improves per-
 338 formance over model evolution on a static synthetic dataset, in Sec. D.7, we demonstrate the effect
 339 of seed task selection, and in Sec. D.8, the effect of the scientist model.** Moreover, in Fig. 1, we
 340 demonstrate improvement over time, further demonstrating the efficacy of our method.

341 **Table 3: Comparison to prior**

342 **QD methods.** Results show avg.
 343 Coverage across benchmarks.

Configuration	N=3	N=8
AC/DC (ours)	58.21	66.86
DNS	58.39	64.97
CQD	58.13	63.68

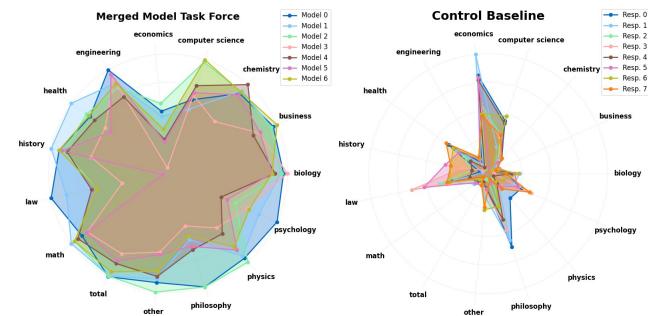
344 Finally, Tab. 3 compares AC/DC to prior quality diversity meth-
 345 ods (DNS Bahlious-Boldi et al. (2025), CycleQD (CQD) Kuroki
 346 et al. (2025)) that train on benchmark-specific datasets, directly
 347 optimizing for the benchmarks (see Sec. D.5 for implementa-
 348 tion details). In contrast, *AC/DC does not optimize for any bench-
 349 mark* and achieves the highest benchmark Coverage at N=8
 350 models, demonstrating that our method discovers more diverse
 351 and capable LLMs. Concurrently, DNS improves on coverage
 352 over CQD, justifying its usage in AC/DC.

353 5 QUALITATIVE CASE STUDY

354 5.1 EMERGENT SPECIALIZATION OF MERGED MODELS

355 Fig. 3 illustrates how our eight discovered models develop distinct performance profiles, with each
 356 model excelling in specific categories while performing differently across others, enabling them to
 357 function as complementary components of a collective intelligence. This specialization creates val-
 358 uable Coverage patterns where models contribute unique capabilities to the ensemble. For instance,
 359 *Model 4* may not achieve the highest overall accuracy, but it provides correct answers to chemistry
 360 questions that no other model in the population can solve.

361 Similarly, *Model 6* demonstrates su-
 362 perior performance in business and
 363 computer science domains, while
 364 *Model 3* excels specifically in biol-
 365 ogy. These specialized capabilities
 366 ensure the task force can collectively
 367 address questions across diverse do-
 368 mains, even when individual models
 369 show weaknesses in certain areas. In
 370 stark contrast, our control baseline
 371 of resampling the generalist instruct
 372 model eight times exhibits minimal
 373 variance across categories and over-
 374 all weaker performance. Moreover,
 375 Fig. 1 demonstrates that AC/DC dis-
 376 covers improved single LLMs, as ev-
 377 idenced by superior MMLU Pro per-
 378 formance compared to baseline mod-
 379 els.



380 **Figure 3: Emergent Specialization of Merged Models on**
 381 **MMLU Pro Categories.** We show the accuracy distribution
 382 across categories for our N=8 merged models (left) and our
 383 N=8 control evaluation (right). For clarity, we don't display
 384 the instruct seed model, which is part of our task force, and
 385 thus only show seven models.

378
379

5.2 RESPONSE EXAMPLES AND DIVERSITY FROM MERGED MODELS

380
381
382

All of the following play an important role in regulating respiration in humans EXCEPT:
 A. an increase in the amount of CO₂ in the blood B. a decrease in the amount of O₂ in the blood C. a decrease in the plasma pH level D. strenuous exercise

Baseline Response

```
## Step 1: Identify the factors that regulate respiration in humans
[The factors that regulate respiration in humans include [...]]
## Step 2: Determine which factor does not play a role in regulating respiration
[Strenuous exercise is not a factor that regulates respiration in the same way as the others. [...]]
The best answer is D.
```

Model 1

```
## Step 1: [Concise description]
A decrease in the amount of O2 in the blood would not directly regulate respiration.
## Step 2: [Concise description]
An increase in the amount of CO2 in the blood, a decrease in the plasma pH level, and strenuous exercise all play roles in regulating respiration.
The best answer is B.
```

9 out of 10 cheerleaders are 64" tall. The 10th cheerleader is 60" tall. If they build a human pyramid, where 4 girls are on the bottom, 3 stand on top of the 4, 2 stand on top of the 3 and the shortest girl is at the top, how tall is the human pyramid in feet?

Baseline Response

```
There are 9 girls who are 64" tall and 1 girl who is 60" tall. The height of the 9 girls is 9*64 = 576", and the height of the 10th girl is 60", so the total height of all 10 girls is 576" + 60" = 636". When they build a human pyramid, the height is determined by the tallest girl, which is 60". [...]
The final answer is 5 feet.
```

Model 2

```
The 9 tallest cheerleaders are 64" tall. The 10th is 60" tall. The 4 on the bottom are 64" tall.
[...]
So the height of the human pyramid in feet is 252" / 12" = 21 feet.
The final answer is 21 feet.
```

Imagine an operator $S \setminus \text{vec}(P)S$ of a system have components SP_x, SP_y and SP_z . The matrix form of the operator SP_xS has components in the first row (0) $\setminus \text{hbar}/\sqrt{2}, 0, \dots$ Choices:

(A) $\setminus \text{hbar}$ (B) $\setminus \text{hbar}/2$ (C) $\setminus \text{hbar}/\sqrt{2}$ (D) $\setminus \sqrt{2}\setminus \text{hbar}$

Baseline Response

```
The uncertainty principle states that the product of the uncertainties of two non-commuting operators is greater than or equal to the absolute value of their commutator.
[...]
Therefore, the uncertainty in  $SP_zS$  is at least  $\setminus \text{hbar}/28$ .
The correct answer is (B)  $\setminus \text{hbar}/28$ .
```

Model 3

```
To find the uncertainty in the operator  $SP_zS$ , we need to use the formula for the uncertainty principle, which is given by:

$$\Delta P_z \Delta P_x = \frac{1}{2} \setminus \text{hbar}$$

First, we need to find the uncertainty in  $SP_zS$ , which is represented by  $\Delta P_z S$ .
[...]
So, the correct answer is (D)  $\setminus \text{hbar}/\sqrt{2}$ .
```

391
392
393
394

Figure 4: **Merged models unlock new capabilities.** Higher Coverage means that our models solve tasks that baselines didn’t. These examples show a sample from MMLU, GSM8K, and GPQA, respectively, where none of the baseline models—including math experts, code experts, repeated instruct model sampling, and the 72B model—successfully solve the task, whereas at least one of our discovered models provides the correct solution.

395
396
397
398
399
400

Merged Models vs. Baselines on Benchmark Tasks. Our evolutionary approach demonstrates superior Coverage by solving tasks that baseline methods cannot address. Fig. 4 presents examples where none of the baseline models—including math experts, code experts, repeated instruct model sampling, and the 72B model—successfully solve the task, whereas at least one of our discovered models provides the correct solution.

401
402
403
404
405

The MMLU example (left) illustrates multiple advantages of our approach beyond correctness. While baseline models fail to identify the correct answer, the 72B model additionally demonstrates poor response formatting, incorrectly placing square brackets around the main reasoning rather than the step headers. Our model not only provides the correct answer but also delivers concise, well-formatted reasoning that adheres to the expected structure.

406
407
408
409

The GSM8K arithmetic example (middle) showcases improved mathematical reasoning capabilities that likely emerge from our model merging process. By combining the potentially overfitted math expert with the general instruct model, AC/DC appears to broaden narrow mathematical skills to handle a wider range of instruction formats, resulting in more robust problem-solving abilities.

410
411
412
413
414
415
416

The GPQA physics example (right) reveals an important advantage related to output length constraints. During evaluation, responses exceeding the 1024-token limit often result in incorrect assessments, and most baseline responses violate this constraint. Our evolutionary process implements a 512-token maximum during training, creating selection pressure for models that produce concise answers. Qualitative analysis suggests our discovered models find solutions using fewer tokens, leading to higher accuracy within the evaluation framework’s constraints. This demonstrates how evolutionary pressure can optimize for practical deployment considerations beyond raw capability.

417
418
419

Create an analogy for the following scenario: 'Explaining the concept of recursion in programming'. Your analogy should:
 1. Clearly convey the concept or impact described in the scenario, breaking it down into understandable steps. [...]

420
421
422
423
424
425
426
427

Model 1

```
def analogy_of_recursion():
  # Step 1: Define the analogy
  analogy = "Imagine you are trying to find a specific book in a large library."
  # Step 2: Break down the analogy into understandable steps
  analogy += "\n\nYou start at the main entrance and look for the section where books on programming are kept."
  analogy += "\nOnce you find the programming section, you look for the specific category of 'Recursion'. [...]
```

Model 2

To explain the concept of recursion in programming, let's use the analogy of a friend who is lost in a city they are unfamiliar with.

Imagine that you are in a city you are not familiar with and you need to find a specific location. You don't have a map or GPS, so you ask a local for directions. [...]

Model 3

Explaining the concept of recursion in programming can be compared to the process of solving a maze.

Imagine you are at the entrance of a maze, and you need to find the exit. The maze is complex and has many twists and turns. [...]

Model 1

```
def max_subarray_sum(nums):
  max_sum = float('-inf')
  current_sum = 0

  for num in nums[1:]:
    current_sum += num
    max_sum = max(max_sum, current_sum)

  return max_sum
```

Model 2

```
def max_subarray_sum(nums):
  if not nums:
    return 0
  max_sum = current_sum = nums[0]

  for num in nums[1:]:
    current_sum += num
    max_sum = max(max_sum, current_sum)

    if current_sum < 0:
      current_sum = 0
  return max_sum
```

Model 3

```
def max_subarray_sum(nums):
  if not nums:
    return 0
  max_sum = current_sum = nums[0]

  for num in nums[1:]:
    current_sum = max(num, current_sum + num)
    max_sum = max(max_sum, current_sum)

  return max_sum
```

428
429
430
431

Figure 5: **Models in our Task Force give diverse Answers.** Two examples of synthetic tasks generated by AC/DC and the answers of 3 models in our Task Force. In the left example, we can see how all three models give different analogies. Moreover, Model 1 structures the analogy in a Python function. For the right example, we can see that our models provide 3 different implementations of the same optimal algorithm.

432 **Merged Model Response Diversity on Synthetic Tasks.** Building on the quantitative capability
 433 distribution demonstrated in Sec. 5.1, we examine qualitatively whether our models generate diverse
 434 responses by analyzing their outputs on two synthetic tasks (Fig. 5).
 435

436 The creative writing task (left) requires both analogical reasoning and computer science knowledge,
 437 revealing distinct approaches across our three models. Each model proposes a completely different
 438 analogy—library navigation, urban directions, and maze solving—demonstrating genuine diversity
 439 in conceptual frameworks rather than superficial variations. Notably, one model presents its analogy
 440 as a Python function, likely reflecting its ancestry from a code expert model and illustrating how
 441 evolutionary merging preserves specialized formatting preferences even in non-coding contexts (for
 442 more details on model evolution analysis, see Sec. E.2).
 443

444 The algorithm implementation task (right) shows diversity in coding style and approach while main-
 445 taining algorithmic correctness. These variations demonstrate that our evolutionary process pro-
 446 duces models with different coding philosophies and defensive programming practices, suggesting
 447 genuine stylistic diversity beyond mere surface-level differences.
 448

449 This qualitative analysis confirms that our discovered models exhibit meaningful diversity in both
 450 creative reasoning and technical implementation, supporting the quantitative evidence of broad ca-
 451 pability distributions and validating that our evolutionary approach generates truly complementary
 452 rather than redundant model behaviors. More qualitative analyses on coevolution are in Sec. E.
 453

454 Additionally, in Appendix Sec. I, we demonstrate quantitative and qualitative analysis comparing the
 455 three expert seed models to discovered merged models, investigating how challenging our synthetic
 456 tasks are to the expert models compared to our merged models. We find that our merged models,
 457 on average and as individual models, perform better on our synthetic data, demonstrating further
 458 evidence for the complexity of our synthetic data and the capabilities emerging through AC/DC,
 459 potentially beyond those present in off-the-shelf models.
 460

461 5.3 QUALITY AND DIVERSITY OF SYNTHETIC TASKS - A HUMAN STUDY

462 To validate the quality and novelty of our synthetically generated tasks, we conducted a human
 463 study where three expert reviewers evaluated 47 synthetic tasks and 49 benchmark tasks across
 464 three dimensions: correctness, out-of-distribution (OOD) nature relative to standard benchmarks,
 465 and creativity. Full methodology and results details are provided in Appendix Sec. H.
 466

467 Table 4: Human evaluation results for synthetic tasks. Values show mean \pm standard error across
 468 all labels.
 469

470 Correctness	471 Out-of-Distribution	472 Creativity
473 $97.8\% \pm 2.2\%$	474 $68.9\% \pm 6.9\%$	475 $37.8\% \pm 7.2\%$

476 Results demonstrate that our approach generates high-quality tasks with strong novelty characteris-
 477 tics. The 97.8% correctness rate confirms that synthetic tasks are well-formed and solvable. Crit-
 478 ically, nearly 70% were rated as out-of-distribution compared to established benchmarks, provid-
 479 ing evidence that our method successfully generates novel task types beyond existing evaluation
 480 datasets, supporting our claims for OOD training. Over one-third were rated as creative, indicating
 481 exploration of problem-solving approaches not commonly tested by standard benchmarks.
 482

483 As a validation baseline, we also evaluated tasks from eight standard benchmarks. These showed
 484 substantially lower OOD (10.2%) and creativity (6.1%) ratings, with the few exceptions con-
 485 centrated exclusively in complex graduate-level benchmarks (MMLU-Pro, GPQA). This pattern
 486 confirms that reviewers appropriately distinguished between novel synthetic tasks and established
 487 benchmark content. Statistical analysis reveals strong inter-rater agreement on objective metrics
 488 (correctness: $p = 0.46$, OOD: $p = 0.57$), demonstrating robust and reliable findings.
 489

490 6 CONCLUSION, LIMITATIONS, AND FUTURE WORK

491 This work introduces AC/DC, a framework for automatically discovering diverse LLM collectives
 492 through open-ended coevolution of models and synthetic tasks. Our approach demonstrates that
 493

486 EvoMerge can create task forces that outperform both larger monolithic models (while using fewer
 487 parameters) and manually curated expert ensembles. AC/DC *does not optimize for any downstream*
 488 *benchmark* and achieves consistent improvements across multiple model families, with evolved pop-
 489 *ulations showing a wider coverage of capabilities and emergent specializations that validate the*
 490 *discovery of complementary skills.*

491 We highlight limitations with AC/DC that motivate further work. Firstly, successful merge outcomes
 492 can depend on empirically testing seed model combinations; for example, strongly fine-tuned mod-
 493 *els with divergent parameter spaces merges poorly, potentially limiting performance gains (Horoi*
 494 *et al., 2025) (e.g., see results with Llama3, Sec. C).* The framework relies on a fixed scientist LLM
 495 for task generation, constraining exploration potential. AC/DC primarily discovers emergent skills
 496 through crossover rather than acquiring new knowledge, bounded by the initial seed models’ ca-
 497 *abilities, which could be addressed through mutation (e.g., our mutation operator).* Finally, an
 498 *inherited limitation from EvoMerge is that it requires seed models that are fine-tuned versions of the*
 499 *same base model.*

500 Key future work directions include developing recursive self-improving scientist models using
 501 evolved model populations for task generation. *Furthermore, as with all prior attempts towards un-*
 502 *bounded open-endedness, extending runs well beyond an arbitrary limit on coevolution steps would*
 503 *enable investigation of longer-term open-ended dynamics and whether innovation rates remain sta-*
 504 *ble over extended time horizons (as we observe promising signs of continual task and model inno-*
 505 *vations in Appendix Sec. D.3 and Sec. D.6).* Moreover, similar to how a lot of research focuses
 506 *on developing base LLMs suitable for subsequent post-training, research on the understanding of*
 507 *model merging compatibility of seed models is a relevant future research direction. We investigate*
 508 *potential ad-hoc predictors for the compatibility of seed models for evolutionary model merging*
 509 *in Appendix Sec. J, which can be an interesting starting point for future research.* Additionally, ex-
 510 *panding scientist LLM tools (e.g., adding web search capabilities) for task generation would enhance*
 511 *the correctness and scope of novel tasks (Lu et al., 2024).* Integrating model fine-tuning could en-
 512 *able more efficient knowledge acquisition beyond crossover-based discovery.* Moreover, advanced
 513 *merging techniques such as M2N2 (Abrantes et al., 2025) could provide higher-degree-of-freedom*
 514 *model combinations.* Finally, implementing model collaboration during training and test-time in-
 515 *ference could enhance population-level performance.* Nevertheless, independently developing more
 516 *sophisticated multi-agent best-of-N extraction methods could be a valuable complementary research*
 517 *direction (Inoue et al., 2025).*

518 In conclusion, AC/DC represents a paradigm shift from scaling individual models toward deliber-
 519 *ately developing complementary agent collectives through a more open-ended process designed for*
 520 *continual innovation, and opens up multiple exciting areas of research.* This distributed specializa-
 521 *tion approach offers a path to parameter-efficient AI systems that achieve sophisticated capabilities*
 522 *without the computational costs of ever-larger monolithic (frontier) models (cf. Sec. L).* We see
 523 *many possibilities for advancing collective LLMs through this new paradigm of automatic devel-
 524 *opment of a population of diverse and capable models.* With AC/DC, we demonstrate a first step
 525 *towards this vision, bringing us closer to discovering collective AI that is as open-ended and complex*
 526 *as human civilization.**

527 ETHICS STATEMENT

528 Our approach focuses on automatically coevolving LLMs and synthetic tasks. As this work only
 529 encompasses the evaluation of models on synthetic and benchmark tasks without involving sensitive
 530 data, human subjects, or potential misuse applications, we identify no ethical concerns.

535 REPRODUCIBILITY STATEMENT

536 To ensure reproducibility of our results, we provide source code and configs, showing the details of
 537 the algorithm, run setup, seed tasks, and LLM prompts. All base models and evaluation benchmarks
 538 used in this work are publicly available.

540 REFERENCES
541

542 João Abrantes, Robert Lange, and Yujin Tang. Competition and attraction improve model fusion. In
543 *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1217–1225, 2025.

544 Fuma Aki, Riku Ikeda, Takumi Saito, Ciaran Regan, and Mizuki Oka. Llm-poet: Evolving complex
545 environments using large language models, 2024.

546 Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. Evolutionary optimization of
547 model merging recipes. *Nature Machine Intelligence*, 7(2):195–204, 2025.

548 Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang,
549 Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, et al. A survey on data selection
550 for language models. *arXiv preprint arXiv:2402.16827*, 2024.

551 Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan,
552 Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language
553 models. *arXiv preprint arXiv:2108.07732*, 2021.

554 Ryan Bahlous-Boldi, Maxence Faldor, Luca Grillotti, Hannah Jammohamed, Lisa Coiffard, Lee
555 Spector, and Antoine Cully. Dominated novelty search: Rethinking local competition in quality-
556 diversity. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 104–112,
557 2025.

558 Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones,
559 Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harm-
560 lessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

561 Varun Bhatt, Bryon Tjanaka, Matthew Fontaine, and Stefanos Nikolaidis. Deep surrogate assisted
562 generation of environments. *Advances in Neural Information Processing Systems*, 35:37762–
563 37777, 2022.

564 Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding,
565 Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with
566 longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

567 Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx,
568 Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportu-
569 nities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

570 David M Bossens and Danesh Tarapore. Quality-diversity meta-evolution: Customizing behavior
571 spaces to a meta-objective. *IEEE Transactions on Evolutionary Computation*, 26(5):1171–1181,
572 2022.

573 Herbie Bradley, Andrew Dai, Hannah Teufel, Jenny Zhang, Koen Oostermeijer, Marco Bellagente,
574 Jeff Clune, Kenneth Stanley, Grégory Schott, and Joel Lehman. Quality-diversity through ai
575 feedback, 2023.

576 Jonathan C. Brant and Kenneth O. Stanley. Minimal criterion coevolution: a new approach to open-
577 ended search. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO
578 ’17, pp. 67–74, New York, NY, USA, 2017. Association for Computing Machinery. ISBN
579 9781450349208. doi: 10.1145/3071178.3071186. URL <https://doi.org/10.1145/3071178.3071186>.

580 Jonathan C. Brant and Kenneth O. Stanley. Diversity preservation in minimal criterion coevolution
581 through resource limitation. In *Proceedings of the 2020 Genetic and Evolutionary Computation
582 Conference*, GECCO ’20, pp. 58–66, New York, NY, USA, 2020. Association for Computing
583 Machinery. ISBN 9781450371285. doi: 10.1145/3377930.3389809. URL <https://doi.org/10.1145/3377930.3389809>.

584 Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and
585 Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling.
586 *arXiv preprint arXiv:2407.21787*, 2024.

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

597

598 Konstantinos Chatzilygeroudis, Antoine Cully, Vassilis Vassiliades, and Jean-Baptiste Mouret.
 599 Quality-diversity optimization: a novel branch of stochastic optimization. In *Black Box Opti-
 600 mization, Machine Learning, and No-Free Lunch Theorems*, pp. 109–135. Springer, 2021.

601 Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared
 602 Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri,
 603 Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan,
 604 Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian,
 605 Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios
 606 Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex
 607 Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders,
 608 Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa,
 609 Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob
 610 McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating
 611 large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021a.

612 Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared
 613 Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large
 614 language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021b.

615 Jeff Clune. Ai-gas: Ai-generating algorithms, an alternate paradigm for producing general artificial
 616 intelligence, 2020. URL <https://arxiv.org/abs/1905.10985>.

617

618 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser,
 619 Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to
 620 solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

621

622 Cédric Colas, Vashisht Madhavan, Joost Huizinga, and Jeff Clune. Scaling map-elites to deep neu-
 623 roevolution. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pp.
 624 67–75, 2020.

625

626 Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth Stanley, and Jeff
 627 Clune. Improving exploration in evolution strategies for deep reinforcement learning via a popu-
 628 lation of novelty-seeking agents. *Advances in neural information processing systems*, 31, 2018.

629

630 Antoine Cully and Yiannis Demiris. Quality and diversity optimization: A unifying modular frame-
 631 work. *IEEE Transactions on Evolutionary Computation*, 22(2):245–259, 2017.

632

633 Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like
 634 animals. *Nature*, 521(7553):503–507, 2015.

635

636 Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch,
 637 and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment
 638 design. *Advances in neural information processing systems*, 33:13049–13061, 2020.

639

640 Aaron Dharna, Cong Lu, and Jeff Clune. Foundation model self-play: Open-ended strategy innova-
 641 tion via foundation models. *arXiv preprint arXiv:2507.06466*, 2025.

642

643 Stephane Doncieux, Alban Laflaqui  re, and Alexandre Coninx. Novelty search: a theoretical per-
 644 spective. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 99–106,
 645 2019.

646

647 Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving fac-
 648 tuality and reasoning in language models through multiagent debate. In *International Conference
 649 on Machine Learning (ICML)*, 2024. arXiv:2305.14325.

650

651 Mayalen Etcheverry, Bert Wang-Chak Chan, Cl  ment Moulin-Frier, and Pierre-Yves Oudeyer.
 652 Meta-diversity search in complex systems, a recipe for artificial open-endedness? 2021.

648 Maxence Faldor, Jenny Zhang, Antoine Cully, and Jeff Clune. Omni-epic: Open-endedness via
 649 models of human notions of interestingness with environments programmed in code, 2024.
 650

651 Manon Flageat, Bryan Lim, and Antoine Cully. Enhancing map-elites with multiple parallel evo-
 652 lution strategies. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp.
 653 1082–1090, 2024.

654 Matthew C Fontaine, Julian Togelius, Stefanos Nikolaidis, and Amy K Hoover. Covariance matrix
 655 adaptation for the rapid illumination of behavior space. In *Proceedings of the 2020 genetic and*
 656 *evolutionary computation conference*, pp. 94–102, 2020.

657

658 Dan Friedman and Adji Bousoo Dieng. The vendi score: A diversity evaluation metric for machine
 659 learning. *arXiv preprint arXiv:2210.02410*, 2022.

660

661 Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. Data-efficient design exploration
 662 through surrogate-assisted illumination. *Evolutionary computation*, 26(3):381–410, 2018.

663

664 Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. Are quality diversity algorithms better
 665 at generating stepping stones than objective-based search? In *Proceedings of the Genetic and*
 666 *Evolutionary Computation Conference Companion*, pp. 115–116, 2019.

667

668 Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Fos-
 669 ter, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muen-
 670 nighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang
 671 Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model
 672 evaluation harness, 07 2024. URL <https://zenodo.org/records/12608602>.

673

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

677

678 Luca Grillotti, Maxence Faldor, Borja González León, and Antoine Cully. Quality-diversity actor-
 679 critic: Learning high-performing and diverse behaviors via value and successor features critics.
 680 In *International Conference on Machine Learning*. PMLR, 2024.

681

682 David Ha and Yujin Tang. Collective intelligence for deep learning: A survey of recent develop-
 683 ments. *Collective Intelligence*, 1(1):26339137221114874, 2022.

684

685 Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution
 686 strategies. *Evolutionary Computation*, 9:159–195, 06 2001. doi: 10.1162/106365601750190398.

687

688 Alex Havrilla, Andrew Dai, Laura O’Mahony, Koen Oostermeijer, Vera Zisler, Alon Albalak, Fab-
 689 rizio Milo, Sharath Chandra Raparthy, Kanishk Gandhi, Baber Abbasi, et al. Surveying the effects
 690 of quality, diversity, and complexity in synthetic data from large language models. *arXiv preprint*
 691 *arXiv:2412.02980*, 2024.

692

693 Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Ja-
 694 cob Steinhardt. Measuring massive multitask language understanding. In *International Confer-
 695 ence on Learning Representations*, 2021a. URL <https://openreview.net/forum?id=d7KBjmI3GmQ>.

696

697 Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob
 698 Steinhardt. Measuring massive multitask language understanding. *Proceedings of the Interna-
 699 tional Conference on Learning Representations (ICLR)*, 2021b.

700

701 Stefan Horoi, Guy Wolf, Eugene Belilovsky, and Gintare Karolina Dziugaite. Less is more: Under-
 702 training experts improves model upcycling. *arXiv preprint arXiv:2506.14126*, 2025.

702

703 Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth*
 704 *International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=t9U3LW7JvX>.

702 Siyuan Huang, Zhiyuan Ma, Jintao Du, Changhua Meng, Weiqiang Wang, and Zhouhan Lin. Mirror-
 703 consistency: Harnessing inconsistency in majority voting. *arXiv preprint arXiv:2410.10857*,
 704 2024.

705 Edward Hughes, Michael Dennis, Jack Parker-Holder, Feryal Behbahani, Aditi Mavalankar, Yuge
 706 Shi, Tom Schaul, and Tim Rocktaschel. Open-endedness is essential for artificial superhuman
 707 intelligence, 2024.

708 Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Os-
 709 trow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint*
 710 *arXiv:2410.21276*, 2024.

711 Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt,
 712 Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint*
 713 *arXiv:2212.04089*, 2022.

714 Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi,
 715 and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=6t0Kwf8-jrj>.

716 Yuichi Inoue, Kou Misaki, Yuki Imajuku, So Kuroki, Taishi Nakamura, and Takuya Akiba. Wider or
 717 deeper? scaling llm inference-time compute with adaptive branching tree search. *arXiv preprint*
 718 *arXiv:2503.04412*, 2025.

719 Prime Intellect. Prime-environments, 2025. URL <https://github.com/PrimeIntellect-ai/prime-environments>.

720 Geoffrey Irving, Paul Christiano, and Dario Amodei. AI safety via debate. *arXiv preprint*
 721 *arXiv:1805.00899*, 2018.

722 Minqi Jiang, Tim Rocktäschel, and Edward Grefenstette. General intelligence requires rethinking
 723 exploration. *Royal Society Open Science*, 10(6):230539, 2023.

724 Yuu Jinnai, Jonathan Uesato, Ankit Mahajan, and Markus N. Rabe. Regularized best-of-n sampling
 725 to mitigate reward hacking for language model alignment. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*,
 726 2025. *arXiv:2404.01054*.

727 Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and
 728 Robert McHardy. Challenges and applications of large language models. *arXiv preprint*
 729 *arXiv:2307.10169*, 2023.

730 Nikhil Kandpal, Brian Lester, Colin Raffel, Sebastian Majstorovic, Stella Biderman, Baber Abbasi,
 731 Luca Soldaini, Enrico Shippole, A Feder Cooper, Aviya Skowron, et al. The common pile v0.
 732 1: An 8tb dataset of public domain and openly licensed text. *arXiv preprint arXiv:2506.05209*,
 733 2025.

734 Leon Keller, Daniel Tanneberg, Svenja Stark, and Jan Peters. Model-based quality-diversity search
 735 for efficient robot learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and*
 736 *Systems (IROS)*, pp. 9675–9680. IEEE, 2020.

737 Paul Kent, Adam Gaier, Jean-Baptiste Mouret, and Juergen Branke. Bayesian optimisation for
 738 quality diversity search with coupled descriptor functions. *IEEE Transactions on Evolutionary*
 739 *Computation*, 2024.

740 Devvrit Khatri, Lovish Madaan, Rishabh Tiwari, Rachit Bansal, Sai Surya Duvvuri, Manzil Zaheer,
 741 Inderjit S Dhillon, David Brandfonbrener, and Rishabh Agarwal. The art of scaling reinforcement
 742 learning compute for llms. *arXiv preprint arXiv:2510.13786*, 2025.

743 Akarsh Kumar, Jeff Clune, Joel Lehman, and Kenneth O Stanley. Questioning representational
 744 optimism in deep learning: The fractured entangled representation hypothesis. *arXiv preprint*
 745 *arXiv:2505.11581*, 2025.

756 So Kuroki, Taishi Nakamura, Takuya Akiba, and Yujin Tang. Agent skill acquisition for large
 757 language models via cycleQD. In *The Thirteenth International Conference on Learning Repre-*
 758 *sentations*, 2025. URL <https://openreview.net/forum?id=Kvdh12wGC0>.

759
 760 Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brah-
 761 man, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers
 762 in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

763 Joel Lehman and Kenneth O Stanley. Revising the evolutionary computation abstraction: minimal
 764 criteria novelty search. In *Proceedings of the 12th annual conference on Genetic and evolutionary*
 765 *computation*, pp. 103–110, 2010.

766
 767 Joel Lehman and Kenneth O Stanley. Abandoning objectives: Evolution through the search for
 768 novelty alone. *Evolutionary computation*, 19(2):189–223, 2011a.

769 Joel Lehman and Kenneth O Stanley. Evolving a diversity of virtual creatures through novelty search
 770 and local competition. In *Proceedings of the 13th annual conference on Genetic and evolutionary*
 771 *computation*, pp. 211–218, 2011b.

772
 773 Joel Lehman, Kenneth O Stanley, et al. Exploiting open-endedness to solve problems through the
 774 search for novelty. In *ALIFE*, pp. 329–336, 2008.

775 Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O Stanley.
 776 Evolution through large models. In *Handbook of Evolutionary Machine Learning*, pp. 331–366.
 777 Springer, 2023.

778
 779 Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ra-
 780 masesh, Ambrose Sloane, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative
 781 reasoning problems with language models. *Advances in neural information processing systems*,
 782 35:3843–3857, 2022.

783 Pengfei Li, Jianyi Yang, Mohammad A Islam, and Shaolei Ren. Making ai less' thirsty'. *Communi-*
 784 *cations of the ACM*, 68(7):54–61, 2025.

785
 786 Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming
 787 Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-
 788 agent debate. *arXiv preprint arXiv:2305.19118*, 2023.

789
 790 Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming
 791 Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-
 792 agent debate. In *Proceedings of the 2024 conference on empirical methods in natural language*
 793 *processing*, pp. 17889–17904, 2024.

794 Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan
 795 Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint*
 796 *arXiv:2305.20050*, 2023. OpenAI.

797 Bryan Lim, Manon Flageat, and Antoine Cully. Efficient exploration using model-based quality-
 798 diversity with gradients. In *Artificial Life Conference Proceedings 35*, volume 2023, pp. 4. MIT
 799 Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , 2023.

800
 801 Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei
 802 Shen, Fuxiang Zhang, Jiacheng Xu, Yang Liu, and Yahui Zhou. Skywork-reward-v2: Scaling
 803 preference data curation via human-ai synergy. *arXiv preprint arXiv:2507.01352*, 2025a.

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

807
 808 Ruibo Liu, Jerry Wei, Fangyu Liu, Chenglei Si, Yanzhe Zhang, Jinmeng Rao, Steven Zheng, Daiyi
 809 Peng, Diyi Yang, Denny Zhou, et al. Best practices and lessons learned on synthetic data. *arXiv*
preprint arXiv:2404.07503, 2024.

810 Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding,
 811 Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui
 812 Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang.
 813 Agentbench: Evaluating llms as agents. *arXiv preprint arXiv: 2308.03688*, 2023.

814 Chris Lu, Cong Lu, Robert Lange, Jakob N Foerster, Jeff Clune, and David Ha. The ai scientist: To-
 815 wards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

816 Cong Lu, Shengran Hu, and Jeff Clune. Automated capability discovery via foundation model
 817 self-exploration. *arXiv preprint arXiv:2502.07577*, 2025.

818 Pratyush Maini, Skyler Seto, He Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. Rephras-
 819 ing the web: A recipe for compute and data-efficient language modeling. *arXiv preprint
 820 arXiv:2401.16380*, 2024.

821 Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering.
 822 *Journal of Open Source Software*, 2(11):205, 2017. doi: 10.21105/joss.00205. URL <https://doi.org/10.21105/joss.00205>.

823 METR Task Standard Team. Metr task standard, 2024. URL <https://github.com/METR/task-standard/blob/main/STANDARD.md>.

824 Elliot Meyerson and Risto Miikkulainen. Discovering evolutionary stepping stones through behavior
 825 domination. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 139–
 826 146, 2017.

827 Elliot Meyerson, Joel Lehman, and Risto Miikkulainen. Learning behavior characterizations for
 828 novelty search. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*,
 829 pp. 149–156, 2016.

830 Melanie Mitchell. *Complexity: A guided tour*. Oxford university press, 2009.

831 Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint
 832 arXiv:1504.04909*, 2015a.

833 Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint
 834 arXiv:1504.04909*, 2015b.

835 Niklas Muennighoff, Alexander Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra
 836 Piktus, Sampo Pyysalo, Thomas Wolf, and Colin A Raffel. Scaling data-constrained language
 837 models. *Advances in Neural Information Processing Systems*, 36:50358–50376, 2023.

838 Niklas Muennighoff, SU Hongjin, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh,
 839 and Douwe Kiela. Generative representational instruction tuning. In *The Thirteenth International
 840 Conference on Learning Representations*, 2024.

841 Anh Nguyen, Jason Yosinski, and Jeff Clune. Understanding innovation engines: Automated cre-
 842 ativity and improved stochastic optimization via deep learning. *Evolutionary computation*, 24(3):
 843 545–572, 2016.

844 Eleni Nisioti, Claire Ganois, Elias Najarro, Andrew Dai, Elliot Meyerson, Joachim Winther Peder-
 845 sen, Laetitia Teodorescu, Conor F Hayes, Shyam Sudhakaran, and Sebastian Risi. From text to
 846 life: On the reciprocal relationship between artificial life and large language models. In *Artificial
 847 Life Conference Proceedings 36*, volume 2024, pp. 39. MIT Press One Rogers Street, Cambridge,
 848 MA 02142-1209, USA journals-info . . . , 2024.

849 Alexander Novikov, Ngan Vu, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt
 850 Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al.
 851 Alphaevolve: A coding agent for scientific and algorithmic discovery. *Google DeepMind*, 2025.

852 Guanzhong Pan and Haibo Wang. A cost-benefit analysis of on-premise large language model
 853 deployment: Breaking even with commercial llm services, 2025. URL <https://arxiv.org/abs/2509.18101>.

864 Giuseppe Paolo, Alban Laflaquier, Alexandre Coninx, and Stephane Doncieux. Unsupervised
 865 learning and exploration of reachable outcome space. In *2020 IEEE International Conference on*
 866 *Robotics and Automation (ICRA)*, pp. 2379–2385. IEEE, 2020.

867 Giuseppe Paolo, Miranda Coninx, Alban Laflaquier, and Stephane Doncieux. Discovering and
 868 exploiting sparse rewards in a learned behavior space. *Evolutionary Computation*, 32(3):275–
 869 305, 2024.

870 Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. Quality diversity: A new frontier for evolu-
 871 tionary computation. *Frontiers in Robotics and AI*, 3:40, 2016.

872 Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan
 873 Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang,
 874 Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin
 875 Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li,
 876 Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang,
 877 Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv*
 878 preprint *arxiv:2412.15115*, 2025. URL <https://arxiv.org/abs/2412.15115>.

879 David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Di-
 880 rani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a bench-
 881 mark. In *First Conference on Language Modeling*, 2024.

882 Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog,
 883 M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan Ellenberg, Pengming Wang,
 884 Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program
 885 search with large language models. *Nature*, 2023. doi: 10.1038/s41586-023-06924-6.

886 Mikayel Samvelyan, Sharath Chandra Raparthy, Andrei Lupu, Eric Hambro, Aram Markosyan,
 887 Manish Bhatt, Yuning Mao, Minqi Jiang, Jack Parker-Holder, Jakob Foerster, et al. Rainbow
 888 teaming: Open-ended generation of diverse adversarial prompts. *Advances in Neural Information
 889 Processing Systems*, 37:69747–69786, 2024.

890 Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang,
 891 Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathemati-
 892 cal reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

893 Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi.
 894 Judging the judges: A systematic study of position bias in llm-as-a-judge. *arXiv preprint*
 895 *arXiv:2406.07791*, 2024.

896 Olivier Sigaud, Gianluca Baldassarre, Cedric Colas, Stephane Doncieux, Richard Duro, Nicolas
 897 Perrin-Gilbert, and Vieri-Giuliano Santucci. A definition of open-ended learning problems for
 898 goal-conditioned agents. *arXiv preprint arXiv:2311.00344*, 2023.

899 Andries P. Smit, Revan Patel, Thomas Perelman, Gwen Kwon, Boshi Yordanov, Asad Karimullah,
 900 Minesh Sharma, Kun Chui, Di Song, Michael McKeever, and Chi Siu. Should we be going
 901 MAD? A look at multi-agent debate strategies for LLMs. In *International Conference on Machine
 902 Learning (ICML)*, 2024.

903 Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute opti-
 904 mally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*,
 905 2024. August 2024.

906 L B. Soros and Kenneth O. Stanley. Identifying minimal conditions for open-ended evolution
 907 through the artificial life world of chromaria. In *Proc. Int. Conf. on the Synthesis and Simula-
 908 tion of Living Systems (ALIFE)*, pp. 793–800, Cambridge, MA, 2014. MIT Press.

909 Kenneth O. Stanley. Why open-endedness matters. *Artificial life*, 25(3):232–235, 2019.

910 Kenneth O. Stanley and Joel Lehman. *Why greatness cannot be planned: The myth of the objective*.
 911 Springer, 2015.

918 Kenneth O Stanley, Joel Lehman, and Lisa Soros. Open-endedness: The last grand challenge you've
 919 never heard of. *While open-endedness could be a force for discovering intelligence, it could also*
 920 *be a component of AI itself*, 2017.

921

922 Qi Sun, Edoardo Cetin, and Yujin Tang. Transformer-squared: Self-adaptive LLMs. In *The*
 923 *Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=dh4t9qmcvK>.

925 Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung,
 926 Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. Challenging big-
 927 bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*,
 928 2022.

929

930 Yashar Talebirad, Ali Parsaee, Vishwajeet Ohal, Amirhossein Nadiri, Csongor Szepesvari, Yash
 931 Mouje, and Eden Redman. Wisdom of the machines: Exploring collective intelligence in
 932 LLM crowds. In *First Workshop on Social Simulation with LLMs*, 2025. URL <https://openreview.net/forum?id=fxqroxvUhk>.

934 Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Ma-*
 935 *chine Learning Research*, 9(86):2579–2605, 2008. URL <http://jmlr.org/papers/v9/vandermaaten08a.html>.

937

938 Vassilis Vassiliades and Jean-Baptiste Mouret. Discovering the elite hypervolume by leveraging in-
 939 terspecies correlation. In *Proceedings of the Genetic and Evolutionary Computation Conference*,
 940 GECCO '18, pp. 149–156, New York, NY, USA, 2018. Association for Computing Machinery.
 941 ISBN 9781450356183. doi: 10.1145/3205455.3205602. URL <https://doi.org/10.1145/3205455.3205602>.

943 Vassilis Vassiliades, Konstantinos Chatzilygeroudis, and Jean-Baptiste Mouret. Using centroidal
 944 Voronoi tessellations to scale up the multidimensional archive of phenotypic elites algorithm.
 945 *IEEE Transactions on Evolutionary Computation*, 22(4):623–630, 2017.

946

947 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez,
 948 Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural informa-*
 949 *tion processing systems*, 30, 2017.

950 Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improv-
 951 ing text embeddings with large language models. *arXiv preprint arXiv:2401.00368*, 2023a.

952

953 Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O. Stanley. Paired open-ended trailblazer (POET):
 954 endlessly generating increasingly complex and diverse learning environments and their solutions.
 955 *CoRR*, abs/1901.01753, 2019. URL <http://arxiv.org/abs/1901.01753>.

956 Rui Wang, Joel Lehman, Aditya Rawal, Jiale Zhi, Yulun Li, Jeffrey Clune, and Kenneth Stanley.
 957 Enhanced poet: Open-ended reinforcement learning through unbounded invention of learning
 958 challenges and their solutions. In *International conference on machine learning*, pp. 9940–9951.
 959 PMLR, 2020.

960

961 Weiqin Wang, Yile Wang, and Hui Huang. Ranked voting based self-consistency of large language
 962 models. *arXiv preprint arXiv:2505.10772*, 2025.

963

964 Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdh-
 965 ery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models.
 966 In *International Conference on Learning Representations (ICLR)*, 2023b. arXiv:2203.11171.

967

968 Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and
 969 Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions.
 970 In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual*
 971 *Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13484–
 972 13508, Toronto, Canada, July 2023c. Association for Computational Linguistics. doi: 10.18653/
 973 v1/2023.acl-long.754. URL <https://aclanthology.org/2023.acl-long.754/>.

972 Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and
 973 Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions,
 974 2023d.

975 Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming
 976 Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang,
 977 Rongqi Fan, Xiang Yue, and Wenhui Chen. MMLU-pro: A more robust and challenging multi-
 978 task language understanding benchmark. In *The Thirty-eight Conference on Neural Information
 979 Processing Systems Datasets and Benchmarks Track*, 2024. URL <https://openreview.net/forum?id=y10DM6R2r3>.

980 981

982 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny
 983 Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in
 984 neural information processing systems*, 35:24824–24837, 2022.

985 986

986 Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes,
 987 Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model
 988 soups: averaging weights of multiple fine-tuned models improves accuracy without increasing
 989 inference time. In *International conference on machine learning*, pp. 23965–23998. PMLR, 2022.

990 991

991 Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Dixin
 992 Jiang. Wizardlm: Empowering large language models to follow complex instructions, 2023.

993 994

994 Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Re-
 995 solving interference when merging models. *Advances in Neural Information Processing Systems*,
 36:7093–7115, 2023.

996 997

997 An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li,
 998 Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang,
 999 Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai,
 1000 Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Ke-Yang Chen, Kexin Yang, Mei Li, Min Xue,
 1001 Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai
 1002 Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan
 1003 Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang
 1004 Zhang, Yunyang Wan, Yunfei Chu, Zeyu Cui, Zhenru Zhang, and Zhi-Wei Fan. Qwen2 technical
 1005 report. *arXiv preprint arxiv:2407.10671*, 2024.

1006 1007

1007 An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu,
 1008 Chang Gao, Chengan Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint
 1009 arXiv:2505.09388*, 2025.

1010 1011

1011 Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Ab-
 1012 sorbing abilities from homologous models as a free lunch. In *Forty-first International Conference
 1013 on Machine Learning*, 2024.

1014 1015

1015 Yang Yu. Pass@ k metric for rlrv: A diagnostic tool of exploration, but not an objective. *arXiv
 1016 preprint arXiv:2511.16231*, 2025.

1017 1018

1018 Jenny Zhang, Joel Lehman, Kenneth Stanley, and Jeff Clune. Omni: Open-endedness via models of
 1019 human notions of interestingness. *arXiv preprint arXiv:2306.01711*, 2023.

1020 1021

1021 Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin godel machine: Open-
 1022 ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025a.

1023 1024

1024 Yiqun Zhang, Peng Ye, Xiaocui Yang, Shi Feng, Shufei Zhang, Lei Bai, Wanli Ouyang, and
 1025 Shuyue Hu. Nature-inspired population-based evolution of large language models. *arXiv preprint
 1026 arXiv:2503.01155*, 2025b.

1027 1028

1028 Yulun Zhang, Matthew C Fontaine, Amy K Hoover, and Stefanos Nikolaidis. Deep surrogate as-
 1029 sisted map-elites for automated hearthstone deckbuilding. In *Proceedings of the Genetic and
 1030 Evolutionary Computation Conference*, pp. 158–167, 2022.

1026 Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun
1027 Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero
1028 data. *arXiv preprint arXiv:2505.03335*, 2025.

1029

1030 Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang,
1031 Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica.
1032 Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information
1033 Processing Systems (NeurIPS)*, 2023. arXiv:2306.05685.

1034

1035

1036

1037

1038

1039

1040

1041

1042

1043

1044

1045

1046

1047

1048

1049

1050

1051

1052

1053

1054

1055

1056

1057

1058

1059

1060

1061

1062

1063

1064

1065

1066

1067

1068

1069

1070

1071

1072

1073

1074

1075

1076

1077

1078

1079

1080
1081 A SETUP DETAILS1082 A.1 TRAINING HYPERPARAMETERS
10831084 This section provides the hyperparameters used for all experiments unless otherwise specified.
10851086 Table 5: AC/DC hyperparameters.
1087

Component	Parameter	Value
General	Number of generations	50
	Active models per gen	16
	New offspring per gen	8
	Active tasks per gen	250
Scientist Model	Scientist Model	Qwen/Qwen2.5-72B-Instruct
Reproduction	First k singular values	256
	mutation rate	0.25
Evaluation (evolution)	max tokens	512
	temperature	0
	top-p	1.0
Task Generator	Task difficulty threshold	0.5
	Max reflections	3
	Embedding Model	intfloat/e5-mistral-7b-instruct
t-SNE	n_components	2
	perplexity	50
	learning_rate	200
	n_iter	3000
	init	pca
	random_state	42
HDBSCAN	early_exaggeration	6.0
	min_cluster_size	16
	min_samples	4
	cluster_selection_epsilon	2
	cluster_selection_method	eom
	metric	euclidean

1116 A.2 MODELS USED
11171118 For our experiments with different model families, we use the following models from Huggingface:
11191120 **Qwen2 7B (Yang et al., 2024)**

- **Control:** Qwen/Qwen2-7B-Instruct
- **Experts:**
 - Grogros/Qwen2-7B-OurSafeCoder
 - Qwen/Qwen2-Math-7B-Instruct
 - Qwen/Qwen2-7B-Instruct
- **Big Model:** Qwen/Qwen2-72B-Instruct

1129 **Qwen2.5 7B (Qwen et al., 2025)**
1130

- **Control:** Qwen/Qwen2.5-7B-Instruct
- **Experts:**
 - prithivMLmods/Neumind-Math-7B-Instruct

1134 – pe-nlp/R1-Qwen2.5-7B-Instruct-code
 1135 – Qwen/Qwen2.5-7B-Instruct
 1136 • **Big Model:** Qwen/Qwen2.5-72B-Instruct

1138 **Qwen3 14B (Yang et al., 2025)**

1139 • **Control:** Qwen/Qwen3-14B
 1140 • **Experts:**
 1141 – sunblaze-ucb/Qwen3-14B-Intuitior-MATH-1EPOCH
 1142 – ertghiu256/qwen-3-14b-code-and-math-reasoning
 1143 – Qwen/Qwen3-14B
 1144 • **Big Model:** Qwen/Qwen3-235B-A22B-Instruct-2507

1145 **DeepSeek V1 7B (Bi et al., 2024)**

1146 • **Control:** deepseek-ai/deepseek-llm-7b-chat
 1147 • **Experts:**
 1148 – deepseek-ai/deepseek-math-7b-instruct
 1149 – deepseek-ai/deepseek-coder-7b-base-v1.5
 1150 – deepseek-ai/deepseek-llm-7b-chat
 1151 • **Big Model:** deepseek-ai/deepseek-llm-67b-chat

1152 **Llama3 8B (Grattafiori et al., 2024)**

1153 • **Control:** meta-llama/Meta-Llama-3-8B-Instruct
 1154 • **Experts:**
 1155 – MathGenie/MathCoder2-Llama-3-8B
 1156 – rombodawg/Llama-3-8B-Instruct-Coder
 1157 – meta-llama/Meta-Llama-3-8B-Instruct
 1158 • **Big Model:** meta-llama/Meta-Llama-3-70B-Instruct

1159
 1160
 1161
 1162
 1163
 1164
 1165
 1166
 1167
 1168
 1169
 1170
 1171
 1172
 1173
 1174
 1175
 1176
 1177
 1178
 1179
 1180
 1181
 1182
 1183
 1184
 1185
 1186
 1187

1188 A.3 ALGORITHM DETAILS

1189

1190 A.3.1 DNS NOVELTY SCORE COMPUTATION

1191

1192 The Dominated Novelty Score (DNS) is a key component of our selection mechanism that balances
1193 quality and diversity in the model archive. For each model, we compute its novelty relative to models
1194 with higher fitness, encouraging retention of models that solve unique subsets of tasks. Algorithm 2
1195 provides the detailed computation.

1196

1197 **Algorithm 2** Dominated Novelty Score Computation for One Solution

```

1: Input: Solution  $s$ , Archive  $\mathcal{A}$ , Parameters  $k, \alpha_{dom}, w$  (difficulty weights)
2:  $\mathcal{F} \leftarrow \{s' \in \mathcal{A} : \text{fitness}(s') > \text{fitness}(s)\}$   $\triangleright$  Find fitter solutions
3: if  $\mathcal{F} = \emptyset$  then
4:   return  $\alpha_{dom}$   $\triangleright$  Maximum score if no fitter solutions exist
5: end if
6:  $\text{scores} \leftarrow []$ 
7: for each  $s' \in \mathcal{F}$  do
8:    $v_s \leftarrow$  skill vector of  $s$   $\triangleright$  Binary vector of task successes
9:    $v_{s'} \leftarrow$  skill vector of  $s'$ 
10:  // Compute weighted unique skills: tasks solved by  $s$  but not by  $s'$ 
11:   $\text{unique\_weighted} \leftarrow \sum_i w_i \cdot (v_s[i] \wedge \neg v_{s'}[i])$ 
12:   $\text{total\_weighted} \leftarrow \sum_i w_i$   $\triangleright$  Sum of all difficulty weights
13:  // Normalize by total skill vector to measure contribution to coverage
14:   $\text{skill\_score} \leftarrow \frac{\text{unique\_weighted}}{\text{total\_weighted}} \times 100$ 
15:   $\text{scores.append}(\text{skill\_score})$ 
16: end for
17:  $\text{scores.sort}()$   $\triangleright$  Sort ascending
18:  $k' \leftarrow \min(k, |\text{scores}|)$   $\triangleright$  Use at most  $k$  neighbors
19: return  $\frac{1}{k'} \sum_{i=1}^{k'} \text{scores}[i]$   $\triangleright$  Mean of  $k$  lowest scores

```

1216

1217 The algorithm identifies models that complement the existing archive by solving tasks that higher-
1218 fitness models fail on. A model receives a high novelty score when it uniquely solves many tasks
1219 that fitter models cannot solve. The skill score is computed as the ratio of weighted unique skills to
1220 the total weighted skill vector, measuring the model's contribution to overall task coverage relative
1221 to fitter solutions. Key configuration parameters include:

1222

- α_{dom} (default: 999): Maximum novelty score assigned when a model has no fitter competitors
- k (default: 3): Number of nearest neighbors for novelty computation
- w_i : Difficulty weights for task i , computed as the fraction of current model population failing that task

1223

1224

1225

1226

1227

1228

1229

1230

1231

1232

1233

1234 When using difficulty weights, harder tasks (those failed by more models) contribute more to the
1235 novelty score, encouraging retention of models that solve challenging problems. This mechanism
1236 ensures diversity in the archive while maintaining a preference for higher overall fitness. This is also
1237 related to what metric Abrantes et al. (2025) used for diversity maintenance in M2N2. Our method
1238 directly leverages this as part of explicit distance measurements between model behaviors to support
1239 the use of QD algorithms in AC/DC.

1240

1241

1242 A.4 EXPERIMENTAL SETUP DETAILS
12431244 This section provides details to the experimental setup such as the benchmarks used and how we aug-
1245 ment them (Sec. A.4.1) and details on our best-of-N single-answer-selection methods (Sec. A.4.3).
12461247 A.4.1 BENCHMARK DETAILS
12481249 We evaluate on MMLU (Hendrycks et al., 2021a), MMLU Pro (Wang et al., 2024), GPQA (Rein
1250 et al., 2024), BBH (Suzgun et al., 2022), GSM8K (Cobbe et al., 2021), Minerva MATH (Lewkowycz
1251 et al., 2022), Humaneval (Chen et al., 2021b), and MBPP (Austin et al., 2021).
12521253 We use Language Model Evaluation Harness (lm-evaluation-harness) from EleutherAI
1254 (Gao et al., 2024) to evaluate on these benchmarks, with task names mmlu_cot_llama,
1255 mmlu_pro_llama, gpqa_main_cot_zeroshot, bbh_cot_zeroshot, gsm8k_llama,
1256 minerva_math, humaneval_instruct, and mbpp_instruct respectively. We use the orig-
1257 inal evaluation config settings for these benchmarks from the repo with the last commit on June 3
1258 (UTC-00).1259 Since multiple-choice question (MCQ) benchmarks are prone to "cheating" when evaluating Cover-
1260 age (randomly sampling each option once will lead to 100% accuracy), we design new, open-ended
1261 versions of these benchmarks, which we then refer to as *MMLU judge*, *MMLU Pro judge*, *GPQA*
1262 *judge*, and *BBH judge*. We design these benchmarks such that we do not provide multiple-choice
1263 options in the questions, relying only on the absolute knowledge/capability of the subject model.
12641265 To generate these new datasets, we prompt an LLM to filter out all samples that are not possible
1266 to answer without the multiple choice options (e.g., questions in the form of "Which of the
1267 following [...]"). The remaining questions that are self-consistent are prompted to the sub-
1268 ject model.1269 To evaluate the correctness of a subject's answer, we prompt an LLM judge to determine whether
1270 the candidate solution is correct given the ground truth answer.
12711272 We provide the dataset filtering and LLM judge prompts in Sec. F.7.
12731274 A.4.2 BASELINE DETAILS
12751276 Fundamentally, our proposal is to rethink model development, moving from developing one large,
1277 monolithic LLM, to a population based approach, automatically developing a collective of diverse
1278 smaller LLMs.
12791280 To evaluate the effectiveness of this approach, we target improving the Coverage over large scale
1281 open-source and even proprietary LLMs. Nevertheless, one may ask, whether our discovered col-
1282 lective is better than simply re-prompting the same sized general instruct model. This is our control
1283 baseline. Moreover, one might ask, if we start of with the three experts as seed models, we would
1284 want to achieve higher Coverage than those.
12851286 In Tab. 1, we observe that, compared to the big models, we achieve noticeable Coverage improve-
1287 ments, and even surpass or get close to the GPT-4o performance, demonstrating that our collective
1288 of smaller models fundamentally possess the capabilities of much larger models to answer the re-
1289 spective questions.
12901291 The Experts N=8 baseline emerged from the question of "What if we resample the three experts
1292 N times". Although we argue that this approach of manually selecting experts is unscalable and it
1293 being even worse to tune the sampling distribution, it is an interesting comparison. To establish this
1294 baseline and ensure consistency across model families, we arbitrarily select the 3-3-2 (instruct-code-
1295 math), which was selected based on observations that the code models achieved higher scores on a
1296 wider rage of benchmarks than the math model, arguably, making the baseline stronger.
12971298 In addition to these "sanity check" baselines, in Tab. 3, we compare against prior Quality Diversity
1299 work, demonstrating that our algorithm discovers a task force that achieves higher Coverage while
1300 not optimizing for any downstream benchmark (which the other methods actively do).
1301

1296
1297

A.4.3 BEST-OF-N SINGLE-ANSWER-SELECTION METHODS

1298
1299
1300

In this section, we elaborate on our single-answer-selection methods used in Sec. 4. For the two LLM-judge-based methods, we only provide the final subject model answers without the reasoning trace.

1301
1302
1303
1304
1305

Divide and Conquer. For this method, we prompt a judge LLM to decide on the correct answer for two candidate solution. If the model deems both wrong, we ask it to provide the answer that is "more correct". We apply this process in a devide-and-conquer approach, first, grouping all candidate solutions into pairs, then selecting the "winners", and repeating this process until only one answers remains.

1306
1307

Monarchical LLM. We provide all N answers to an LLM judge at once and prompt it to select the correct answer.

1308
1309
1310

RM-based. We leverage a scalar reward model (Liu et al., 2025a) to score each full candidate submission and select the candidate with the highest score.

1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349

1350 **B DETAILED QUANTITATIVE RESULTS**
13511352 **B.1 COVERAGE RESULTS**
13531354 **B.1.1 RESULTS FOR ALL MODELS ON ALL BENCHMARKS**
13551356
1357 **Table 6: Coverage performance comparison across different benchmarks and model configurations**
1358 for all model families used (see Sec. A.2).

Method	MMLU judge	MMLU Pro judge	GPQA judge	BBH judge	GSM8K	Minerva	HumanEval	MBPP	Avg.
Qwen 2									
Big Model	50.44	47.12	4.36	78.32	90.83	35.56	49.39	64.00	52.50
Control 3	54.58	50.97	7.72	73.01	92.34	36.10	79.88	64.20	57.35
Experts 3	47.05	41.78	6.38	65.29	91.28	47.50	81.10	66.00	55.80
Ours 3	55.42	51.28	9.06	69.44	89.99	39.88	84.15	65.40	58.08
Control 8	67.69	64.27	12.75	84.54	95.68	46.10	92.07	70.60	66.71
Ours 8	67.20	63.03	15.44	82.03	94.77	53.26	89.02	72.00	67.09
Qwen 2.5									
Big Model	49.30	47.00	0.00	82.30	91.70	28.30	89.60	75.40	57.95
Control 3	58.17	56.96	8.05	83.18	93.33	31.94	90.85	66.60	61.14
Experts 3	48.33	50.40	5.03	68.24	97.42	46.22	90.85	74.2	60.09
Ours 3	62.46	59.45	5.37	84.34	93.71	41.02	89.02	74.60	63.75
Control 8	69.35	67.95	12.42	89.29	95.53	39.84	92.68	70.4	67.18
Ours 8	73.31	69.63	11.41	90.29	96.44	53.84	90.24	82.00	70.90
Qwen 3									
Big Model	57.30	53.60	0.30	83.70	94.70	23.10	94.50	74.60	60.23
Control 3	63.59	64.95	10.74	87.93	94.77	32.02	93.29	76.40	65.46
Experts 3	63.77	65.09	9.06	89.67	95.38	37.48	93.90	77.00	66.42
Ours 3	62.74	64.60	12.75	87.71	94.24	31.32	93.90	78.80	65.76
Control 8	73.01	73.55	13.42	92.0	95.83	37.76	93.29	77.6	69.56
Ours 8	73.17	75.09	17.45	92.94	96.29	39.28	95.73	81.4	71.42
DeepSeek V1									
Big Model	40.10	31.70	3.00	62.20	80.50	25.20	26.20	56.40	40.66
Control 3	39.85	29.39	2.68	48.21	74.15	20.94	29.27	49.00	36.69
Experts 3	27.34	18.14	1.34	34.01	54.97	36.44	60.98	41.00	34.28
Ours 3	37.22	32.74	1.68	51.43	84.0	39.08	58.54	39.20	42.99
Control 8	55.94	43.74	8.39	65.57	85.67	33.02	45.73	60.00	49.76
Ours 8	49.35	46.54	6.71	69.34	89.92	51.02	72.56	47.40	54.11

1389 **B.1.2 ANALYSIS AND KEY INSIGHTS**
13901391 **Evolutionary Discovery Outperforms Manual Curation.** Despite expert models’ superior performance in specialized domains, our evolutionary approach discovers model combinations with broader skill Coverage. This suggests that the space of useful model combinations extends beyond obvious domain-specific specializations.
13921393 **Model Diversity Beats Parameter Scaling.** Across both task force sizes, our distributed approach
1394 outperforms larger monolithic models while using fewer total parameters. This finding supports the
1395 hypothesis that specialized model populations can achieve superior Coverage compared to scaling
1396 individual models.
13971398 **Complementary vs. Overlapping Skills.** The consistent improvements over control baselines
1399 demonstrate that our evolved models develop genuinely complementary capabilities rather than re-
1400 dundant expertise. This validation supports our core hypothesis about automatic discovery of col-
1401 lective intelligence through evolutionary pressure for both quality and diversity.
1402

1404 B.2 BEST-OF-N RESULTS
14051406 B.2.1 RESULTS FOR ALL MODELS ON ALL BENCHMARKS
14071408 Table 7: Best-of-N performance comparison across different benchmarks and model configurations
1409 for all model families (see Sec. A.2).
1410

Method	MMLU	MMLU Pro	GPQA	BBH	GSM8K	Minerva	HumanEval	MBPP	Avg.
Qwen 2									
Big Model	82.04	63.15	10.27	68.42	90.83	35.56	49.39	64.00	57.96
<i>Divide and Conquer</i>									
Experts 3	72.16	46.33	22.54	63.97	88.70	43.70	66.46	49.80	56.71
Control 3	59.81	42.07	16.07	60.01	89.54	30.84	75.61	58.20	54.02
Ours 3	71.31	50.20	19.64	63.11	86.58	35.28	78.66	56.00	57.60
Control 8	71.62	51.45	27.68	67.72	90.22	37.12	79.88	58.60	60.54
Ours 8	76.14	56.84	25.22	69.73	88.86	41.46	79.27	56.20	61.72
Qwen 2.5									
Big Model	84.65	70.64	10.71	53.14	91.74	28.28	89.63	75.40	63.02
<i>Divide and Conquer</i>									
Experts 3	77.97	60.72	19.86	65.40	90.22	44.12	81.10	66.26	63.21
Control 3	76.29	59.99	17.41	60.91	91.21	28.30	80.49	64.00	59.83
Ours 3	78.22	60.34	18.53	62.20	88.55	33.64	82.32	65.80	61.20
Control 8	78.21	64.40	21.65	67.87	91.13	31.00	85.98	64.20	63.06
Ours 8	79.41	63.50	22.10	70.60	89.91	45.92	81.71	66.40	64.94
Qwen 3									
Big Model	88.81	73.11	8.48	38.30	94.69	23.12	94.51	74.60	61.95
<i>Divide and Conquer</i>									
Experts 3	81.81	67.99	17.86	55.51	92.65	34.68	82.32	56.20	61.13
Control 3	81.36	68.86	13.62	51.10	93.10	28.58	89.02	73.40	62.38
Ours 3	81.17	69.31	14.06	52.10	92.12	28.12	89.63	75.26	62.72
Control 8	82.03	70.99	18.75	59.12	92.95	30.92	89.63	73.40	64.72
Ours 8	82.33	71.47	20.31	62.05	91.58	32.96	87.20	68.46	64.55
DeepSeek V1									
Big Model	70.81	43.12	14.73	57.86	80.52	25.20	26.22	56.40	46.86
<i>Divide and Conquer</i>									
Experts 3	51.30	29.96	27.90	50.22	54.81	31.40	48.78	24.60	39.87
Control 3	55.35	33.55	22.54	54.03	73.62	17.80	20.73	46.00	40.45
Ours 3	59.76	37.87	29.24	55.03	81.20	34.52	50.00	30.80	47.30
Control 8	66.46	41.79	25.22	64.11	81.20	23.94	26.83	48.80	47.29
Ours 8	67.70	47.17	33.26	63.86	85.06	40.30	56.10	23.80	52.16

1444 B.2.2 ANALYSIS AND KEY INSIGHTS
14451446 **Coverage-to-Selection Translation.** The consistent improvements in single answer selection validate that our Coverage gains reflect genuine complementary capabilities rather than statistical artifacts. Models that cover diverse skills collectively also contribute effectively when aggregated through selection mechanisms.
14471448 **Parameter Efficiency.** Our results strongly support the hypothesis that distributed specialized models can achieve superior performance compared to parameter scaling. For example, the N=8 configuration for our Qwen2 and Qwen2.5 7B models outperform a model with 22% more parameters within the same model family, while the N=3 configuration nearly matches a model with 240% more parameters.
14491450 **Selection Method Optimization.** The benchmark-specific selection strategies highlight the importance of matching aggregation methods to task characteristics. This finding suggests that future
1451

1458 work on collective intelligence should consider exploring task-aware selection mechanisms rather
1459 than universal approaches.
1460

1461 **Generalization Beyond Benchmarks.** These findings reinforce our evidence from Sec. B.1 that di-
1462 verse collectives of smaller models can outperform larger monolithic models. Importantly, our open-
1463 ended evolutionary algorithm achieves these results *without optimizing for any specific downstream*
1464 *benchmark*, supporting the hypothesis that diversity-driven evolution discovers broadly applicable
1465 complementary skills rather than benchmark-specific adaptations.
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511

1512 **C EXTENDED DISCUSSION ON LIMITATIONS**

1513

1514 Table 8: Coverage performance comparison across different benchmarks for the Llama 3 family of
 1515 models (see Sec. A.2 for details on the model used).

1516

Method	MMLU judge	MMLU Pro judge	GPQA judge	BBH judge	GSM8K	Minerva	HumanEval	MBPP	Avg.
Llama 3									
Big Model	47.5	43.4	3.7	78.7	92.2	41.6	7.3	0.0	39.30
Control 3	51.12	45.27	6.04	75.96	89.77	37.06	39.02	64.80	51.13
Experts 3	45.9	38.58	8.05	66.47	94.47	32.84	70.12	63.20	52.45
Ours 3	47.74	40.25	8.39	71.91	87.26	33.48	15.85	63.40	46.04
Control 8	64.44	59.49	12.08	87.09	95.45	47.94	58.54	70.40	61.93
Experts 8	62.73	57.55	9.06	86.56	93.71	49.38	53.66	70.80	60.43
Ours 8	59.22	52.13	13.09	83.58	90.98	41.38	41.46	70.40	56.53

1520
 1521 Table 9: Best-of-N performance comparison across different benchmarks for the Llama 3 family of
 1522 models (see Sec. A.2 for details on the model used).

1523

Method	MMLU	MMLU Pro	GPQA	BBH	GSM8K	Minerva	HumanEval	MBPP	Avg.
Big Model	60.67	59.23	17.19	69.21	92.19	41.62	7.32	0.00	43.43
<i>Divide and Conquer</i>									
Experts 3	71.46	51.44	24.78	61.02	82.03	35.56	21.95	55.85	50.51
Control 3	68.47	50.22	24.55	65.73	86.05	31.98	37.80	58.80	52.95
Ours 3	69.94	47.58	20.76	62.71	82.79	29.30	15.85	55.60	48.07
Control 8	71.86	55.46	28.35	73.25	89.16	36.26	52.44	58.80	58.20
Ours 8	74.58	51.51	30.80	69.67	84.23	32.20	37.80	55.40	54.52

1539
 1540 In Sec. 6, we discuss several limitations which we provide further discussion on in this section.
 1541 As mentioned, the process of model merging is strongly reliant on the compatibility of the seed
 1542 models used, as also observed in previous work (Horoi et al., 2025). A supporting example of this
 1543 observation are our experiments on the Llama3 family of models, where Tab. 8 and Tab. 9 present
 1544 our Coverage and BoN results, respectively. Merged models typically inherit both the strengths and
 1545 the limitations of parent models, producing models that are more prone to response degradation even
 1546 on the same benchmark where other kinds of merged models solve benchmark tasks without fail.

1547
 1548
 1549
 1550
 1551
 1552
 1553
 1554
 1555
 1556
 1557
 1558
 1559
 1560
 1561
 1562
 1563
 1564
 1565

1566 **D ADDITIONAL RESULTS**
 1567

1568 The experiments for additional results were performed using the Qwen 2 family of models.
 1569

1570 **D.1 IMPACT OF ALGORITHM COMPONENTS**
 1571

1572 Table 10: Ablation study showing the impact of different components of our evolutionary algorithm.
 1573 Coverage performance across benchmarks for $N=3$ and $N=8$ configurations with individual compo-
 1574 nents removed.
 1575

Configuration	N=3	N=8
Full Method	58.21	66.86
Fitness Only	56.35	66.06
No Mutation	57.75	66.25
No Novelty Filter	57.54	66.51
No Gibberish Filter	56.09	66.18
W/o all components	56.40	59.92

1584 Tab. 10 presents the results of an ablation where we remove one component of our algorithm at a
 1585 time. We discuss the observations in Sec. 4.
 1586

1587 **D.2 IMPACT OF TASK FORCE SELECTION STRATEGIES**
 1588

1589 Table 11: **Impact of Task Force selection strategies.** Results showing average Coverage across all
 1590 benchmarks and all models.
 1591

Method	N=3	N=8
Global Skill Vector (Coverage)	55.30	63.92
Global Skill Vector (Fitness)	54.64	63.43
Random	51.58	62.21

1598 After letting AC/DC run for multiple generations, we populate an extensive global archive of models
 1599 and synthetic tasks. Several approaches to selecting our N models for our Task Force exist. In this
 1600 section, we discuss the following three:

- 1601 • **Global skill vector (Coverage).** For this strategy, we evaluate all our models in our global
 1602 model archive on all tasks in our global task archive. We then select the N models that
 1603 maximize the Coverage on our synthetic data archive, "optimizing" for complementary
 1604 skills.
- 1605 • **Global skill vector (fitness).** Here, we also perform the complete global task archive
 1606 evaluation as above, but now select the N fittest models.
- 1607 • **Random** We here randomly select N models from our global model archive.

1609 Tab. 11 presents the results of these three selection strategies. We observe that random selection
 1610 performs the worst across both $N = 3$ and $N = 8$ scales, demonstrating that intelligent task force
 1611 composition based on our synthetic dataset translates to improved downstream performance. Se-
 1612 lecting based on maximizing Coverage and selecting the highest fitness individuals on our global
 1613 task archive (the global skill vector) both demonstrate superior performance, yet, optimizing for
 1614 synthetic data Coverage yields the highest accuracy gains.
 1615

1616 **D.3 TASK ARCHIVE NOVELTY OVER TIME**
 1617

1618 Fig. 6 presents the adaptation types at each task adaptation cycle and the global Vendi score (Fried-
 1619 man & Dieng, 2022), i.e., the Vendi score for our global task archive. The Vendi score is a measure
 of diversity in the task embedding space. For this experiment, we only enable tasks to be evolved



Figure 6: **Adaptation types and Vendi score over time.** For this experiment, we only enabled adaptations types to be making a task more difficult or completely novel. Moreover, we show the global Vendi Score (Vendi score of the global task archive) over time demonstrating increasing diversity in our task archive.

to be either more difficult or novel. We can observe that over the course of training, we mostly generate *more difficult* tasks, highlighting the increasing difficulty of our task archive over time, until we discover the capability limits of our population at generation 35, 40, and, especially, generation 45, where we see a stark decrease in adaptations for harder tasks and an increase in adaptations for more novel tasks.

Moreover, we observe a steady increase in the Vendi score, demonstrating an increasing diversity within our global task archive. Nevertheless, we observe that with each adaptation cycle, the increase of the Vendi score gradually decreases, compared to the early generations.

The Vendi score measures diversity within the task embedding space. This has the advantage of capturing the spread across diverse problem formulations and wording, which can be more easily separated using embeddings. Nevertheless, this also means that the Vendi score is subject to a core limitation of embedding-based similarity/spread measures. Although the context can be structurally similar, the semantics can be very different. For this reason, following prior work (Lu et al., 2025), we implement a two-layered novelty filter. First, we assess embedding-based similarity, and then we evaluate semantic novelty using an LLM judge. Because of the latter, we can have a more fine-grained assessment of novel tasks, which is reflected in the fact that, although the increase in Vendi score seems to slow down at around generations 40/45, we still observe that we add 71 new tasks (23 harder + 48 novel), which is comparable to earlier generations.

In fact, analyzing the task difficulty dynamics, we observe the generation 45's decreased "harder" adaptations reflect the system discovering current capability limits of the model population and pivoting to novel exploration (48 novel tasks). This suggests continued innovation in task space, especially considering that the space of possible tasks may be incomprehensible, rather than task novelty saturation. Fig. 1 shows continued downstream performance improvement through generation 50, confirming that capability advancement continues.

1674 D.4 DISCUSSION ON EXPERTS N=8 BASELINE
16751676 Table 12: Comparison between Experts and AC/DC with 8 models across different model families.
1677

Method	MMLU	MMLU Pro	GPQA	BBH	GSM8K	Minerva	HumanEval	MBPP	Avg.
Qwen 2									
Experts 8	72.06	69.14	9.73	86.83	95.91	56.44	87.80	74.80	69.09
Ours 8	67.20	63.03	15.44	82.03	94.77	53.26	89.02	72.00	67.09
Qwen 2.5									
Experts 8	72.97	67.86	10.74	89.55	96.97	55.8	93.9	81.0	71.10
Ours 8	73.31	69.63	11.41	90.29	96.44	53.84	90.24	82.0	70.90
Qwen 3									
Experts 8	72.61	73.51	14.77	92.88	97.27	45.8	94.51	79.4	71.34
Ours 8	73.17	75.09	17.45	92.94	96.29	39.28	95.73	81.4	71.42
DeepSeek V1									
Experts 8	47.24	36.15	5.7	55.68	77.48	47.24	70.73	61.2	50.18
Ours 8	49.35	46.54	6.71	69.34	89.92	51.02	72.56	47.4	54.11

1692 AC/DC demonstrates consistent improvements over our primary baselines across model families.
1693 However, one might consider alternative configurations that maximize the utilization of expert models
1694 within our inference budget. To address this, we evaluate a configuration that distributes 8
1695 inference calls across our three expert models (3 instruct, 3 code, 2 math calls), shown in Tab. 12.

1696 While this "Experts 8" baseline achieves competitive performance in some cases, we note several
1697 important distinctions from our approach: (1) the 3-3-2 distribution represents a manually tuned
1698 configuration rather than a principled allocation strategy, (2) it relies on multiple sampling from a
1699 limited set of models rather than leveraging diverse evolved capabilities, and (3) the comparison
1700 conflates inference-time scaling with our method's model discovery process.

1701 The arbitrary nature of this baseline becomes apparent when considering alternative distributions:
1702 other configurations such as 4-2-2 or 2-4-2 would yield different results, and in the extreme case
1703 of 8-0-0 (using only the instruct model), we recover our Control 8 baseline, which we have already
1704 demonstrated that AC/DC consistently outperforms across all model families.

1705 Notably, our approach maintains competitive or superior performance while discovering genuinely
1706 diverse models through evolution, rather than simply increasing inference calls to existing models.
1707 The mixed results across benchmarks suggest that raw inference scaling and evolved model diversity
1708 offer complementary but distinct advantages.

1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727

1728 D.5 COMPARISON TO PRIOR QD METHODS
1729

1730 To compare AC/DC (with coevolution and QD) with prior quality-diversity approaches without
 1731 coevolution (DNS (Bahlous-Boldi et al., 2025) and CycleQD (CQD) (Kuroki et al., 2025)), we
 1732 conduct experiments using identical training conditions. Both baseline methods are [trained on two](#)
 1733 [of the same benchmarks used in our evaluation, and two computer science benchmarks](#), with 50
 1734 fixed training samples drawn from each of GSM8K (Cobbe et al., 2021), MBPP (Austin et al.,
 1735 2021), AgentBench OS, and AgentBench DB (Liu et al., 2023). [For all methods discussed, we train](#)
 1736 [on the same model family, Qwen 2.](#)

1737 For the task force selection strategies:

1738

- 1739 • **CQD:** Following their approach, after evolution, we select the top 2 models from each
 1740 task-specific archive (2 models \times 4 tasks = 8 models total for N=8). For N=3, we collect
 1741 the top models for GSM8K, MBPP, and AgentBench OS.
- 1742 • **DNS:** We select the top-8 models based on local competition scores, which measure per-
 1743 formance against local neighborhoods in the behavior space. For N=3, we select the top-3
 1744 models.

1745 The key distinction is that while both DNS and CycleQD directly optimize for performance on these
 1746 specific benchmarks during training, AC/DC evolves models on synthetically generated tasks with-
 1747 out any benchmark-specific optimization. Despite this apparent disadvantage, our method achieves
 1748 superior performance at N=8, demonstrating that evolution on diverse synthetic tasks can discover
 1749 more capable and complementary models than direct benchmark optimization.

1750 We also analyzed train-set coverage for DNS and CQD. On the 200 tasks total (4 sets of 50 training
 1751 examples across the tasks), we evaluated the top-20 models and top-5 models for DNS (based on
 1752 local competition score). For CQD, we select the top-5 from the top-2 in GSM8K, and top models
 1753 for the three other tasks, and for top-20, we get the five best models for each of the 4 tasks. For top-5
 1754 coverage, DNS beats CQD, obtaining 60.5% versus 56.5%. For top-20, DNS again beats CQD,
 1755 obtaining 70% versus 65%. Results provide even more evidence for the suitability of DNS as a QD
 1756 algorithm for AC/DC over CQD.

1757 D.6 DNS ON STATIC SYNTHETIC DATASET
1758

1760 To estimate the effects of an ablation where we execute our pipeline on a static synthetic dataset,
 1761 we consider the progress made up until generation 5 of our existing AC/DC run on Qwen 2.5 as
 1762 representative performance.

1763 In other words, it is reasonable to expect that the performance would stagnate at around that of
 1764 our task force at generation 5, as up to that point, our synthetic dataset is static and would then be
 1765 updated.

1766 To support this argument, we analyze the newly added models per generation. With a static dataset,
 1767 we expect the number of new (fit and diverse) models discovered to significantly reduce early in the
 1768 evolution process. We observe this behaviour when looking at the number of new models in our
 1769 DNS baseline (Fig. 7 (a)), where we evolve models on the static dataset of downstream benchmark
 1770 training sets. In contrast, looking at the number of new models discovered when running our AC/DC
 1771 algorithm (Fig. 7 (b)), we detect a constant influx of new models.

1772 Considering this, we find that the average performance on our LLM-as-a-judge tasks, our generation
 1773 5 task force achieves 59.66% accuracy, whereas our final task force, achieved through a dynamically
 1774 coevolving synthetic dataset achieves 61.19% (1.53% absolute improvement).

1775 Additionally, Tab. 13 presents the quantitative comparison between task forces evolved on a static
 1776 synthetic dataset (i.e. the initialized task pool without further adaptation) versus our full coevolution
 1777 approach on Qwen 2.5. The static dataset baseline achieves 60.16% average coverage for N=3 and
 1778 64.31% for N=8. In contrast, our coevolving approach achieves 60.27% and 67.93%, respectively,
 1779 representing absolute improvements of +0.11% and +3.62%.

1780 The improvements are particularly pronounced on knowledge-intensive benchmarks (MMLU:
 1781 +3.52%/+8.42%, MMLU Pro: +3.98%/+8.93%) and reasoning tasks (GPQA: +1.68%/+3.36%,

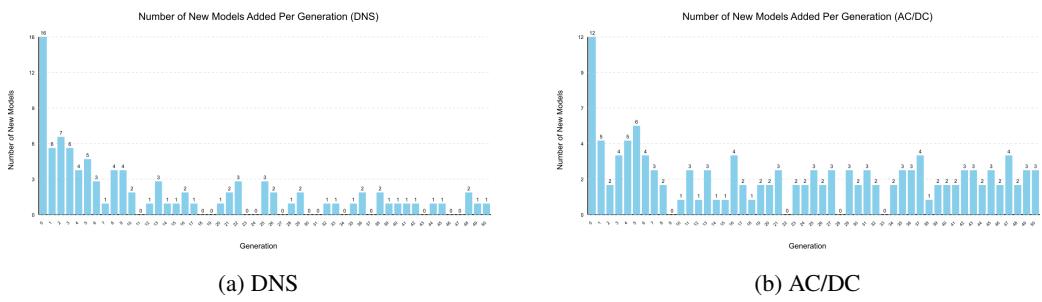


Figure 7: New models added to archive per generation.

Table 13: Static vs. Coevolving Synthetic Dataset. Coverage comparison between task forces evolved on a static synthetic dataset versus our full coevolution approach on Qwen 2.5. The static baseline shows a performance plateau when the synthetic dataset does not dynamically coevolve with the model population. The values for "Ours (Coevolving)" are the average scores across two training runs with two different RNG seeds.

Method	MMLU judge	MMLU Pro judge	GPQA judge	BBH judge	GSM8K	Minerva	HumanEval	Avg.
N=3								
Static Dataset	53.15	51.74	3.69	80.97	92.27	51.48	87.80	60.16
Ours (Coevolving)	56.67	55.72	5.37	81.65	93.10	40.69	88.72	60.27
Improvement	+3.52	+3.98	+1.68	+0.68	+0.83	-10.79	+0.92	+0.11
N=8								
Static Dataset	59.38	57.37	4.36	85.18	94.01	59.62	90.24	64.31
Ours (Coevolving)	67.80	66.30	7.72	88.40	95.64	56.33	93.29	67.93
Improvement	+8.42	+8.93	+3.36	+3.22	+1.63	-3.29	+3.05	+3.62

BBH: +0.68%/+3.22%). While the static baseline shows higher performance on Minerva Math, this is likely due to the early-stage synthetic dataset being biased toward mathematical reasoning tasks, which we expect to be subsequently diversified through coevolution.

These results demonstrate that dynamic coevolution of the synthetic task archive is critical for discovering diverse and complementary model capabilities, validating our core hypothesis that open-ended evolution requires continuous expansion of the challenge space.

D.7 EFFECT OF SEED TASKS

In Sec. E.3, we present the four seed tasks used to initiate the coevolution process.

Table 14: Impact of Seed Tasks. Coverage comparison between coevolution runs with all seed tasks versus without code generation seed tasks on Qwen 2.5. The values for "All Seed Tasks" are the average scores across two training runs with two different RNG seeds.

Method	MMLU judge	MMLU Pro judge	GPQA judge	BBH judge	GSM8K	Minerva	HumanEval	Avg.
N=3								
W/o Code Seed Task	62.46	59.45	5.37	84.34	93.71	41.02	89.02	62.20
All Seed Tasks	56.67	55.72	5.37	81.65	93.10	40.69	88.72	60.27
N=8								
W/o Code Seed Task	73.31	69.63	11.41	90.29	96.44	53.84	90.24	69.31
All Seed Tasks	67.80	66.30	7.72	88.40	95.64	56.33	93.29	67.93

1836 In Tab. 14, we ablate the effect of two different compositions of seed tasks. Once *All Seed Tasks*,
 1837 where we use all four seed tasks, and once *W/o Code Seed Task*, where we use the three seed tasks,
 1838 excluding Sec. E.3.2.

1839 We can observe that the selection of seed tasks can have a big effect on the performance of the
 1840 final task force. Leveraging all four seed tasks, including the coding seed task, can improve the
 1841 coding benchmark performance (at N=8, HumanEval performance being 3.25% better than without
 1842 the coding task), but, in effect, can hurt downstream performance on other benchmarks (e.g., at N=8,
 1843 on MMLU judge and MMLU Pro judge, the performance being -5.51% and -3.33%).

1845 D.8 EFFECT OF SCIENTIST MODEL

1846 To explore the generalizability of our framework to different scientist models, we conducted an
 1847 experiment using Qwen3-235B-A22B as the scientist model while maintaining the existing prompts
 1848 and hyperparameters optimized for Qwen2.5-72B. This experiment highlights both the flexibility of
 1849 our approach and the importance of model-specific prompt engineering.

1850 Using the original generation settings, we observed that the Qwen3-235B-A22B scientist model fre-
 1851 quently produced incomplete task implementations within our generation token limit, yielding 555
 1852 total synthetic tasks compared to over 1000 tasks generated by the Qwen2.5-72B scientist model.
 1853 We evaluated the resulting task forces using the same Qwen2.5-72B subject model across both condi-
 1854 tions.

1856 Scientist Model	1857 Avg. Coverage (N = 3)	1858 Avg. Coverage (N = 8)
1859 Qwen2.5-72B	58.21	66.86
1860 Qwen3-235B-A22B	57.57	65.46

1861 Table 15: Task force performance comparison for two runs using two different scientist models.
 1862 Both configurations use Qwen2.5-72B as the subject model.

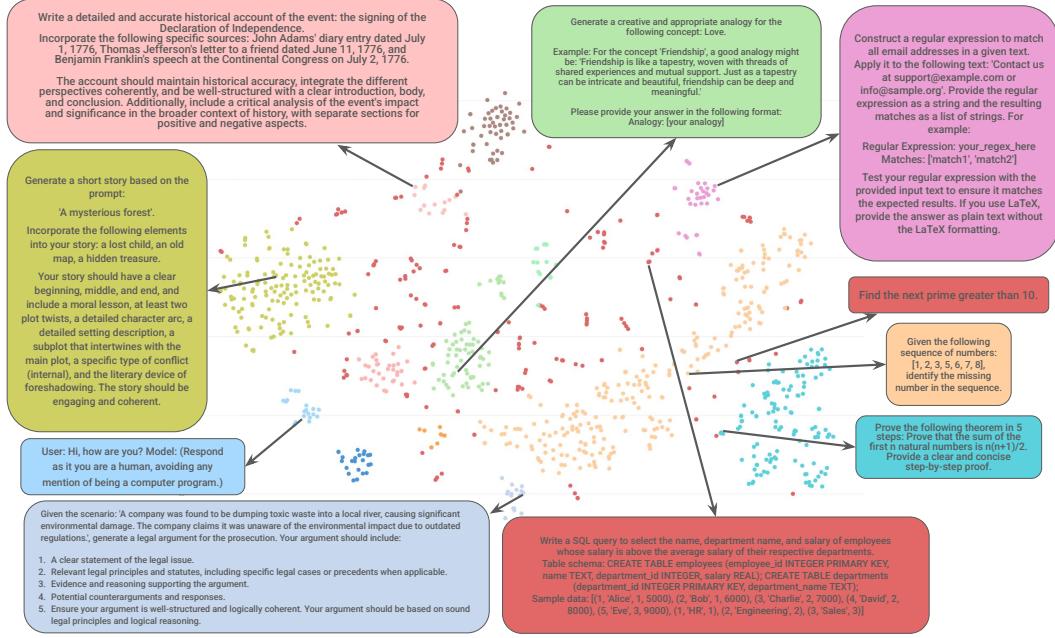
1863 As shown in Table 15, the task forces generated with the Qwen3-235B-A22B scientist model achieve
 1864 slightly lower coverage scores than those generated with Qwen2.5-72B. We attribute this perfor-
 1865 mance gap primarily to the reduced size of the synthetic task pool.

1866 Importantly, the lower task force scores do not necessarily indicate inferior capability of the Qwen3-
 1867 235B-A22B scientist model itself. Rather, this case study illustrates the practical considerations
 1868 when adapting our framework to different scientist models. We expect that with appropriate prompt
 1869 engineering, adjusted generation limits, and hyperparameter tuning specific to the Qwen3-235B-
 1870 A22B model, the framework could effectively leverage its potentially superior capabilities to gener-
 1871 ate higher-quality synthetic tasks and achieve improved downstream performance.

1872
 1873
 1874
 1875
 1876
 1877
 1878
 1879
 1880
 1881
 1882
 1883
 1884
 1885
 1886
 1887
 1888
 1889

1890 E ADDITIONAL QUALITATIVE RESULTS

1891 E.1 CASE STUDY ON SYNTHETIC TASK DIVERSITY



1911 **Figure 8: Analysis of global task archive embedding space generated by AC/DC with Qwen 2.** We represent each task by structuring its metadata using the template in Sec. F.3 and then embedding it using an embedding model (see Tab. 5). We then reduce the dimensionality of the embeddings using t-SNE (van der Maaten & Hinton, 2008). The clusters are automatically generated using HDBSCAN (McInnes et al., 2017).

1921 Fig. 8 presents the global task embedding space evolved through AC/DC with a Qwen2-based population. In the left half of the embedding space, we can find tasks in the writing space, such as

- 1925 **generating a short story based on a prompt** and **generating an analogy for an abstract concept**, requiring creative writing abilities
- 1926 **human-ai-interaction**, requiring emotional intelligence and alignment
- 1927 **writing an accurate historical text**, testing for historical knowledge
- 1928 **developing a legal argument given a case**, examining legal knowledge and persuasion abilities

1933 The right half of the embedding space presents more technical challenges, for example

- 1935 **constructing regular expressions**
- 1936 **solving complex math and pattern recognition problems**
- 1937 **proving mathematical theorems**
- 1938 **implementing code such as SQL queries**

1942 These example tasks demonstrate the diversity of synthetic tasks generated by AC/DC (many of 1943 which might not have been created by human annotators), presenting the breadth of knowledge examined by our system to discover unique capabilities.

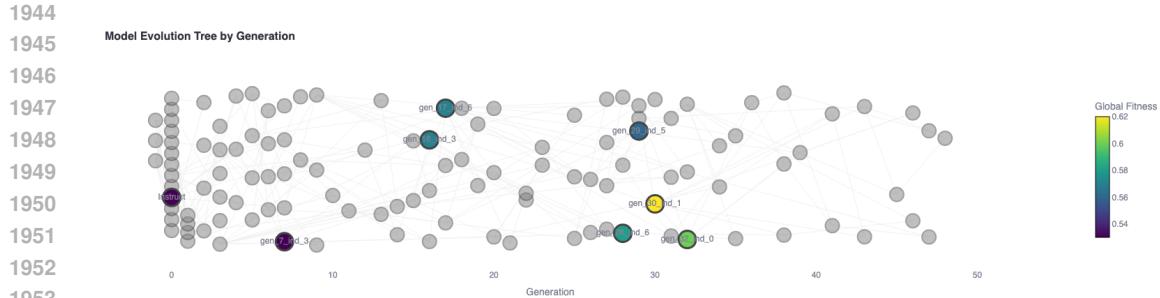


Figure 9: Evolution tree of AC/DC evolving the Qwen2-based seed model. Highlighted models are those selected for the task force by maximizing Coverage on our global task archive.

E.2 CASE STUDY ON MODEL LINEAGES

E.2.1 INSPECTING TASK FORCE SELECTION

Fig. 9 presents the evolution tree produced by AC/DC applied to our three Qwen 2 seed models (see Sec. A.2), highlighting the 8 models selected for our task force. We observe that our task force contains models with a wide fitness range, ranging from 0.53 to 0.62. Nevertheless, as described in Sec. 3, this task force is selected for optimizing Coverage across our synthetic data, meaning that we don't always want the highest fitness individuals, but individuals that have complementary skills (which we discuss in Sec. D.2).

For instance, we observe that our model discovered in generation 30 (presented as `gen_30_ind_1`) achieves the highest global fitness, i.e., fitness across our entire synthetic task archive, yet the instruct model, which is part of our task force, does not achieve the highest fitness, but is part of our task force.

E.2.2 INSPECTING LINEAGE OF SPECIFIC TASK FORCE MODEL

Fig. 10 presents three example lineages of our discovered LLMs, demonstrating complex histories of skill merging of parent models, embedding the knowledge of all three seed models into the weights of the observed models.

The first tree (top) presents the lineage of the model `gen_17_ind_6`, the sixth model evolved in generation 17. As observed in prior work leveraging evolutionary search techniques (Zhang et al., 2025a), we can see that, to reach the model that was part of the task force, we don't only leverage high fitness individuals. To discover model `gen_17_ind_6`, AC/DC first discovered `gen_14_ind_6`, with a local fitness f of 0.476, which is weaker lower than that of its parents (`gen_12_ind_5`, with $f = 0.528$ and `gen_10_ind_2`, $f = 0.488$). This highlights that (locally) weaker solutions are relevant for discovering models for a global task force with complementary skills, by potentially providing unique capabilities, not captured by the simple fitness metric.

The second (middle) and third (bottom) tree present the lineages of the models `gen_28_ind_6` and `gen_29_ind_5`, respectively. Both models are ancestors of the just discussed model `gen_17_ind_6`, yet, we can see that `gen_29_ind_5` has a more complex lineage than `gen_28_ind_6`. `gen_28_ind_6` only adds one new ancestor to its lineage in addition to the lineage of `gen_17_ind_6`, namely `gen_20_ind_3`, demonstrating that even only two additional relevant crossover operations can lead to the discovery of capabilities beneficial for the final task force. On the other hand, although the models `gen_29_ind_5` and `gen_28_ind_6` and `gen_29_ind_5` are only one generation apart, we can observe that `gen_29_ind_5` introduces more novel and complex lineages.

These observations demonstrate core advantage of evolutionary search, (1) discovering novel solutions by building on top of prior ones that don't necessarily seem to be the best performing and (2) the emergent complexity of (2.1) intricate lineages or (2.2) sophisticated capabilities through simple but relevant combinations of prior solutions.

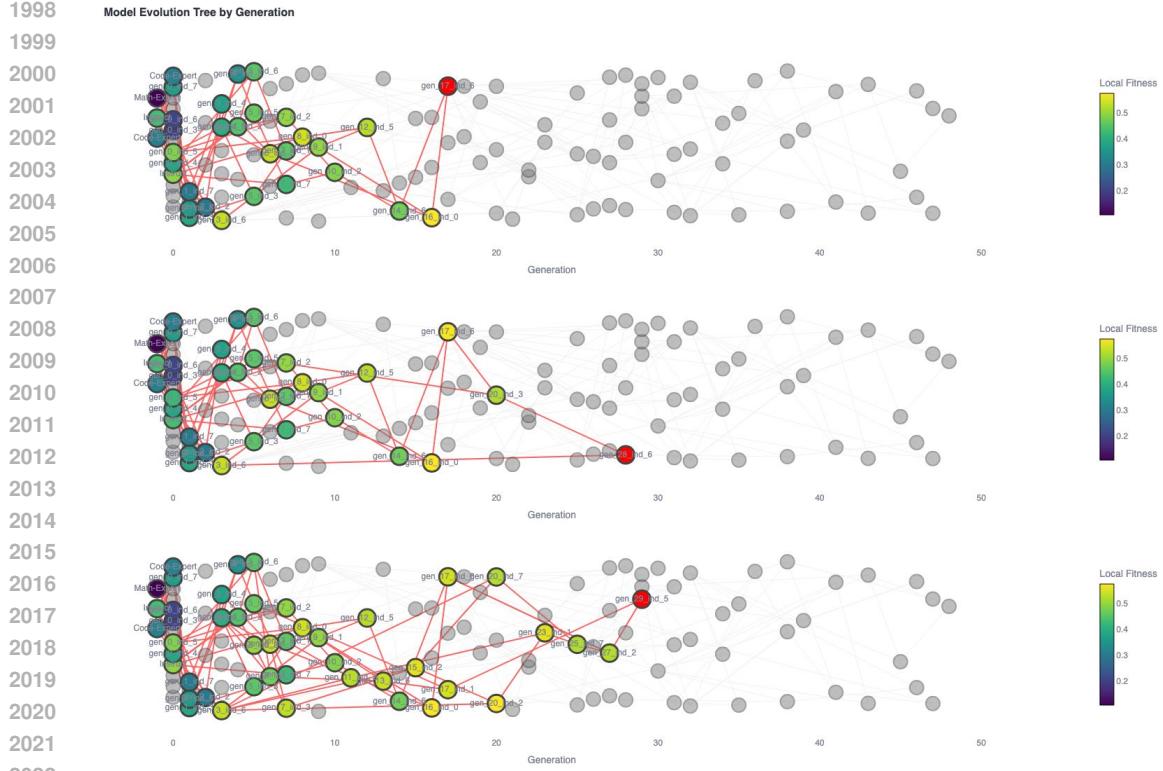


Figure 10: Lineages of AC/DC evolved Qwen2-based models. All presented lineages are of models that are part of the selected task force.

E.3 SEED TASKS USED

E.3.1 SEED TASK 1: CIRCULAR QUEUE LENGTH

```

2030 class TaskFamily:
2031     @staticmethod
2032     def get_tasks() -> dict[str, dict]:
2033         return {
2034             "1": {
2035                 "question": "Given that the storage space for a circular queue is the array A [21], with front pointing to the position before the head element and rear pointing to the tail element, assuming the current values of front and rear are 8 and 3, respectively, the length of the queue is () .",
2036                 "options": {"A": "5", "B": "6", "C": "16", "D": "17"},
2037                 "answer": "C",
2038             }
2039
2040             @staticmethod
2041             def get_instructions(t: dict) -> str:
2042                 return f"""{t["question"]}\nA: {t["options"]["A"]}\nB: {t["options"]["B"]}\nC: {t["options"]["C"]}\nD: {t["options"]["D"]}\n\nReturn the letter of the correct option."""
2043
2044             @staticmethod
2045             def score(t: dict, submission: str) -> float | None:
2046                 return 1.0 if t["answer"].lower() == submission.lower().strip() else 0.0
2047
2048
2049
2050
2051

```

E.3.2 SEED TASK 2: FIRST REPEATED CHARACTER

```

2049 class TaskFamily:
2050     @staticmethod
2051     def get_tasks() -> dict[str, dict]:
2052         return {
2053             "1": {
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2698
2699
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2798
2799
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2898
2899
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2998
2999
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3098
3099
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3198
3199
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3298
3299
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3398
3399
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3498
3499
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3598
3599
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3698
3699
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3798
3799
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3898
3899
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3969
3970
3971
3972
3973
3974
3975
39
```

```

2052
2053     "question": "Write a python function called 'first_repeated_character' to
2054     find the first repeated character in a given string.",
2055     "test_cases": [("abcabc", "a"), ("abc", None), ("123123", "1")],
2056     "expected_func_name": "first_repeated_character",
2057     }
2058   }
2059
2060   @staticmethod
2061   def get_instructions(t: dict) -> str:
2062     return f"""{t["question"]}"""
2063
2064   @staticmethod
2065   def score(t: dict, submission: str) -> float | None:
2066     from sandbox_eval_helper import get_function_name_to_callable
2067
2068     # Test cases from get_tasks
2069     test_cases = t["test_cases"]
2070
2071     correct_count = 0
2072     num_test_cases = len(test_cases)
2073
2074     # Get function name to callable mapping
2075     function_name_to_callable = get_function_name_to_callable(submission)
2076
2077     if len(function_name_to_callable) == 0:
2078       return 0.0
2079
2080     # Get function names
2081     func_names = set(function_name_to_callable.keys())
2082
2083     if t["expected_func_name"] not in func_names:
2084       return 0.0
2085
2086     # Run test cases
2087     for test_input, expected_output in test_cases:
2088       try:
2089         # Execute the function
2090         submission_result = function_name_to_callable[
2091           t["expected_func_name"]
2092         ](test_input)
2093
2094         # Compare with expected output
2095         if submission_result == expected_output:
2096           correct_count += 1
2097
2098       except Exception:
2099         # If function fails on a test case, skip it
2100         continue
2101
2102     pass_rate = correct_count / num_test_cases
2103     min_pass_rate = 1.0
2104
2105     return 1.0 if pass_rate >= min_pass_rate else 0.0
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
2999

```

E.3.3 SEED TASK 3: LAW MULTIPLE CHOICE TREE DAMAGE

```

2902
2903   class TaskFamily:
2904     @staticmethod
2905     def get_tasks() -> dict[str, dict]:
2906       return {
2907         "1": {
2908           "question": "House owns his home in City. On the lawn in front of his home
2909           and within five feet of the public sidewalk there was a large tree. The roots of the tree
2910           caused the sidewalk to buckle severely and become dangerous. An ordinance of City
2911           requires adjacent landowners to keep sidewalks in safe condition. House engaged
2912           Contractor to repair the sidewalk, leaving it to Contractor to decide how the repair
2913           should be made. Contractor dug up the sidewalk, cut back the roots of the tree, and laid
2914           a new sidewalk. Two days after House had paid Contractor the agreed price of the repair,
2915           the tree fell over onto the street and damaged a parked car belonging to Driver. Driver
2916           has asserted claims against House and Contractor, and both defendants admit that cutting
2917           the roots caused the tree to fall. If Driver recovers a judgment against House, does House
2918           have any recourse against Contractor?",
2919           "options": {
2920             "A": "No, if payment by House was an acceptance of the work.",
2921             "B": "No, because House selected Contractor to do the work.",
2922             "C": "Yes, if the judgment against House was based on vicarious liability.",
2923             "D": "Yes, because House's conduct was not a factual cause of the harm.",
2924           }
2925         }
2926       }
2927     }
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
2999

```

```

2106
2107         },
2108         "answer": "C",
2109     }
2110
2111     @staticmethod
2112     def get_instructions(t: dict) -> str:
2113         return f"""{t["question"]}\nA: {t["options"]["A"]}\nB: {t["options"]["B"]}\nC: {t["options"]["C"]}\nD: {t["options"]["D"]}\n\nReturn the letter of the correct option."""
2114
2115     @staticmethod
2116     def score(t: dict, submission: str) -> float | None:
2117         return 1.0 if t["answer"].lower() == submission.lower().strip() else 0.0

```

E.3.4 SEED TASK 4: MATH WORD PROBLEM CLIPS

```

2119
2120     class TaskFamily:
2121         @staticmethod
2122         def get_tasks() -> dict[str, dict]:
2123             return {
2124                 "1": {
2125                     "question": "Natalia sold clips to 48 of her friends in April, and then she
2126                     sold half as many clips in May. How many clips did Natalia sell altogether in April and
2127                     May?", }
2128             }
2129
2130         @staticmethod
2131         def get_instructions(t: dict) -> str:
2132             return f"""{t["question"]}"""
2133
2134         @staticmethod
2135         def score(t: dict, submission: str) -> float | None:
2136             try:
2137                 submission_val = int(submission)
2138                 # Compute the correct answer based on the given reasoning
2139                 april_sales = 48
2140
2141                 # Calculate May sales: half of April's sales
2142                 may_sales = april_sales / 2
2143
2144                 # Calculate total sales for both months
2145                 total_sales = april_sales + may_sales
2146
2147                 return 1.0 if submission_val == total_sales else 0.0
2148             except ValueError:
2149                 return 0.0

```

E.4 EXAMPLES OF GENERATED TASKS

E.4.1 EXAMPLE 1: LOGICAL REASONING FOR LEGAL FALLACIES

```

2145
2146     class TaskFamily:
2147         @staticmethod
2148         def get_tasks() -> dict[str, dict]:
2149             return {
2150                 "1": {
2151                     "argument": "If you're not with us, you're against us. Therefore, you must
2152                     support our policies.",
2153                     "fallacy": "False Dilemma"
2154                 },
2155                 "2": {
2156                     "argument": "John says that climate change is real, but he's just a scientist
2157                     , so his opinion doesn't count.",
2158                     "fallacy": "Ad Hominem"
2159                 }
2160             }
2161
2162             @staticmethod
2163             def get_instructions(t: dict) -> str:
2164                 return f"""Identify the logical fallacy in the following argument, explain why it
2165                 is a fallacy, construct a logically sound counter-argument that addresses the same issue,
2166                 and critique the strength of the counter-argument.
2167
2168                 Argument: {t['argument']}"""

```

```

2160
2161     Format your response as follows:
2162     - Fallacy: [Name of the fallacy]
2163     - Explanation: [Explanation of why it is a fallacy]
2164     - Counter-Argument: [Logically sound counter-argument]
2165     - Critique: [Critique of the counter-argument, including potential weaknesses]"""
2166
2167     @staticmethod
2168     def score(t: dict, submission: str) -> float | None:
2169         from sandbox_eval_helper import eval_with_llm_judge
2170
2171         instructions = TaskFamily.get_instructions(t)
2172
2173         expected_fallacy = t['fallacy']
2174         submitted_fallacy = None
2175         submitted_explanation = None
2176         submitted_counter_argument = None
2177         submitted_critique = None
2178         parts = [part.strip() for part in submission.strip().split('\n')]
2179         for part in parts:
2180             if part.startswith('Fallacy: '):
2181                 submitted_fallacy = part[len('Fallacy: '):].strip()
2182             elif part.startswith('Explanation: '):
2183                 submitted_explanation = part[len('Explanation: '):].strip()
2184             elif part.startswith('Counter-Argument: '):
2185                 submitted_counter_argument = part[len('Counter-Argument: '):].strip()
2186             elif part.startswith('Critique: '):
2187                 submitted_critique = part[len('Critique: '):].strip()
2188
2189             if not (submitted_fallacy and submitted_explanation and submitted_counter_argument
2190 and submitted_critique):
2191                 return 0.0
2192
2193             criteria = [
2194                 f"The identified fallacy should be '{expected_fallacy}'.",
2195                 "The explanation should correctly describe why the identified fallacy is a
2196 fallacy.",
2197                 "The counter-argument should be logically sound and address the same issue as
2198 the original argument.",
2199                 "The critique should identify potential weaknesses in the counter-argument and
2200 provide a balanced evaluation."
2201             ]
2202             return 1.0 if eval_with_llm_judge(instructions, submission, criteria) else 0.0
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213

```

E.4.2 EXAMPLE 2: LOGICAL UNDERSTANDING OF MUTUAL KNOWLEDGE

```

2193     import re
2194
2195     class TaskFamily:
2196         @staticmethod
2197         def get_tasks() -> dict[str, dict]:
2198             return {
2199                 '1': {
2200                     'events': [
2201                         'Agent A puts a red ball in a box.',
2202                         'Agent A tells Agent B that the box contains a red ball.',
2203                         'Agent C opens the box and sees the red ball.'
2204                     ]
2205                 },
2206                 '2': {
2207                     'events': [
2208                         'Agent A puts a blue ball in a box.',
2209                         'Agent A tells Agent B that the box contains a blue ball.',
2210                         'Agent C opens the box and sees the blue ball.'
2211                     ]
2212                 }
2213             }
2214
2215             @staticmethod
2216             def get_instructions(t: dict) -> str:
2217                 events = '\n'.join(t['events'])
2218                 return f"""Given the following sequence of events, determine the final knowledge of
2219 each agent (A, B, C) about the contents of the box.
2220
2221 Sequence of events:
2222 {events}
2223
2224 Provide your answer in the following format:

```

```

2214
2215     - **Agent A knows**: [What Agent A knows]
2216     - **Agent B knows**: [What Agent B knows]
2217     - **Agent C knows**: [What Agent C knows]"""
2218
2219     @staticmethod
2220     def score(t: dict, submission: str) -> float | None:
2221         from sandbox_eval_helper import eval_with_llm_judge
2222         instructions = TaskFamily.get_instructions(t)
2223         criteria = [
2224             "The response should clearly state the final knowledge of each agent (A, B, C).",
2225             "The response should indicate that Agent A knows the box contains the ball and that both B and C know this.",
2226             "The response should indicate that Agent B knows the box contains the ball and that C knows this.",
2227             "The response should indicate that Agent C knows the box contains the ball."
2228         ]
2229         return 1.0 if eval_with_llm_judge(instructions, submission, criteria) else 0.0
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267

```

E.4.3 EXAMPLE 3: CALCULATE ARE OF SEGMENT OF A CIRCLE

```

import math
class TaskFamily:
    @staticmethod
    def get_tasks() -> dict[str, dict]:
        return {
            '1': {'radius': 5, 'angle': 60},
            '2': {'radius': 10, 'angle': 120}
        }

    @staticmethod
    def get_instructions(t: dict) -> str:
        radius = t['radius']
        angle = t['angle']
        return f'Calculate the area of a segment of a circle with radius {radius} and central angle {angle} degrees. Provide the answer as a floating-point number rounded to two decimal places, without any additional text or explanations.'

    @staticmethod
    def score(t: dict, submission: str) -> float | None:
        try:
            radius = t['radius']
            angle = t['angle']
            # Convert angle to radians
            angle_rad = math.radians(angle)
            # Calculate the area of the sector
            sector_area = (angle / 360) * math.pi * (radius ** 2)
            # Calculate the area of the triangle
            triangle_area = 0.5 * (radius ** 2) * math.sin(angle_rad)
            # Calculate the area of the segment
            segment_area = sector_area - triangle_area
            # Round the correct answer to two decimal places
            correct_answer = round(segment_area, 2)
            submitted_answer = float(submission.strip())
            if submitted_answer == correct_answer:
                return 1.0
            else:
                return 0.0
        except ValueError:
            return 0.0
        except Exception as e:
            print(f'Error: {e}')
            return 0.0

```

E.4.4 EXAMPLE 4: PROVE ALGORITHMIC CORRECTNESS OF THE A* ALGORITHM

```

import sympy
class TaskFamily:
    @staticmethod
    def get_tasks() -> dict[str, dict]:
        return {
            "1": {
                "algorithm": "A* algorithm for pathfinding with a consistent heuristic."
            }
        }

```

```

2268
2269         "test_cases": [
2270             ({'A': {'B': 1, 'C': 4}, 'B': {'C': 2}, 'C': {}}, 'shortest_path(A, C) = 3
2271             '),
2272             ({'A': {'B': 10, 'C': 3}, 'B': {'C': 1}, 'C': {'D': 2}}, 'shortest_path(A,
2273             D) = 6'),
2274             ({'A': {'B': 1, 'C': 100}, 'B': {'C': -150}, 'C': {}}, 'shortest_path(A, C
2275             ) = -49')
2276             ],
2277             "expected_func_name": "prove_a_star_algorithm"
2278             },
2279             "2": {
2280                 "algorithm": "Floyd-Warshall algorithm for finding the shortest paths between
2281                 all pairs of vertices in a graph.",
2282                 "test_cases": [
2283                     ({'A': {'B': 1, 'C': 4}, 'B': {'C': 2}, 'C': {}}, 'shortest_paths(A, B, C
2284                     ) = 3'),
2285                     ({'A': {'B': 10, 'C': 3}, 'B': {'C': 1}, 'C': {'D': 2}}, 'shortest_paths(A
2286                     , B, C, D) = 6')
2287                     ],
2288                     "expected_func_name": "prove_floyd_marshall_algorithm"
2289                     }
2290             }
2291
2292             @staticmethod
2293             def get_instructions(t: dict) -> str:
2294                 return f'Generate a formal proof for the correctness of the following algorithm: {t
2295                 ["algorithm"]}. Your proof should be clear, logical, and well-structured, and should
2296                 cover both the base case and the inductive step if applicable. For the A* algorithm,
2297                 address the properties of a consistent heuristic and how it ensures the optimality of the
2298                 solution.'
2299
2300             @staticmethod
2301             def score(t: dict, submission: str) -> float | None:
2302                 from sandbox_eval_helper import eval_with_llm_judge
2303                 instructions = TaskFamily.get_instructions(t)
2304                 criteria = [
2305                     "The proof should be logically sound and well-structured.",
2306                     "The proof should correctly prove the correctness of the given algorithm.",
2307                     "The proof should cover both the base case and the inductive step if applicable.
2308                     ",
2309                     "For the A* algorithm, the proof should address the properties of a consistent
2310                     heuristic and how it ensures the optimality of the solution."
2311                 ]
2312                 return 1.0 if eval_with_llm_judge(instructions, submission, criteria) else 0.0
2313
2314
2315
2316
2317
2318
2319
2320
2321

```

E.4.5 EXAMPLE 5: SYNTHESIZE POSITION PAPER ON GENETIC ENGINEERING

```

2302
2303
2304     import re
2305
2306
2307     class TaskFamily:
2308         @staticmethod
2309         def get_tasks() -> dict[str, dict]:
2310             return {
2311                 "1": {
2312                     "debate_text": "The debate over genetic engineering has become increasingly
2313                     polarized. Proponents argue that genetic engineering can lead to significant advancements
2314                     in medicine, agriculture, and biotechnology. They highlight the potential to cure
2315                     genetic diseases, increase crop yields, and develop new materials. Critics, however,
2316                     raise serious ethical, social, and environmental concerns. They argue that genetic
2317                     engineering could lead to unintended consequences, such as the creation of 'designer
2318                     babies' and the exacerbation of social inequalities. There are also fears about the
2319                     ecological impact, including the potential for genetically modified organisms (GMOs) to
2320                     disrupt natural ecosystems. The debate is further complicated by the lack of
2321                     comprehensive regulatory frameworks and the need for public engagement and transparency.
2322                     Both sides acknowledge the importance of balancing innovation with responsible governance
2323                     , but they differ on the appropriate balance and the role of government in overseeing
2324                     genetic engineering."
2325                 },
2326                 "2": {
2327                     "debate_text": "The ethical implications of genetic engineering are a subject
2328                     of intense debate. Supporters argue that genetic engineering can alleviate suffering by
2329                     treating genetic disorders and improving quality of life. They also contend that it can
2330                     contribute to global food security by developing crops that are resistant to pests and
2331                     climate change. Opponents, however, warn of the potential for genetic engineering to be
2332                     used for non-therapeutic purposes, such as enhancing physical or cognitive traits. They
2333                     raise concerns about the commodification of human life and the potential for genetic
2334                     discrimination. Ethicists also debate the moral status of genetically modified organisms
2335                 }
2336             }
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
2999

```

```

2322     and the rights of future generations. The debate is further complicated by the lack of
2323     international consensus and the varying cultural and religious perspectives on genetic
2324     engineering."
2325     }
2326   }
2327
2328   @staticmethod
2329   def get_instructions(t: dict) -> str:
2330     return f"Given the following debate on genetic engineering, summarize the debate,
2331     evaluate the key arguments and counterarguments, and then synthesize a position paper
2332     that takes a clear stance on the issue. The position paper should be supported by
2333     evidence from the debate and at least three peer-reviewed sources, and should maintain
2334     coherence and logical flow:\n\n{t['debate_text']}\n\nSummary and Evaluation:\n\nPosition
2335     Paper: Incorporate at least three peer-reviewed sources to support your arguments. For
2336     example, you might reference studies on the effectiveness of gene therapy or the
2337     environmental impact of GMOs."
2338
2339   @staticmethod
2340   def score(t: dict, submission: str) -> float | None:
2341     from sandbox_eval_helper import eval_with_llm_judge
2342     instructions = TaskFamily.get_instructions(t)
2343     criteria = [
2344       "The summary should capture the main points and key arguments.",
2345       "The summary should include relevant counterarguments.",
2346       "The summary should provide a critical analysis of ethical, social, and
2347       scientific considerations.",
2348       "The summary should maintain coherence and logical flow.",
2349       "The summary should not contain unnecessary details or tangents.",
2350       "The summary should accurately reflect the content of the debate.",
2351       "The position paper should take a clear stance on the issue.",
2352       "The position paper should be supported by evidence from the debate and at least
2353       three peer-reviewed sources.",
2354       "The position paper should maintain coherence and logical flow.",
2355       "The position paper should address potential counterarguments and provide a well
2356       -rounded view."
2357     ]
2358     return 1.0 if eval_with_llm_judge(instructions, submission, criteria) else 0.0
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375

```

E.4.6 EXAMPLE 6: SIMPLIFY ALGEBRAIC EXPRESSION

```

2351 class TaskFamily:
2352   @staticmethod
2353   def get_tasks() -> dict[str, dict]:
2354     return {
2355       "1": {
2356         "expression": "(x + 3) * (x - 3)"
2357       },
2358       "2": {
2359         "expression": "2 * (x + 4) - 3 * (x - 2)"
2360       }
2361     }
2362
2363
2364   @staticmethod
2365   def get_instructions(t: dict) -> str:
2366     return f"Simplify the following algebraic expression: {t['expression']}. Provide
2367     your answer in the simplest form, without any additional explanation or steps."
2368
2369   @staticmethod
2370   def score(t: dict, submission: str) -> float | None:
2371     from sympy import symbols, simplify
2372     x = symbols('x')
2373     correct_answer = simplify(t['expression'])
2374     try:
2375       # Remove LaTeX delimiters if present
2376       submission = submission.replace('\\', '').strip()
2377       # Simplify the submitted answer
2378       submitted_answer = simplify(submission)
2379       if submitted_answer == correct_answer:
2380         return 1.0
2381       else:
2382         return 0.0
2383     except Exception:
2384       return 0.0
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
2999

```

E.4.7 EXAMPLE 7: IMPLEMENT FUNCTION TO GENERATE FIBONACCI SEQUENCE

```

2376
2377
2378     class TaskFamily:
2379         @staticmethod
2380         def get_tasks() -> dict[str, dict]:
2381             return {
2382                 '1': {'n_terms': 10},
2383                 '2': {'n_terms': 15}
2384             }
2385
2386             @staticmethod
2387             def get_instructions(t: dict) -> str:
2388                 return f'Write a function called `fibonacci_sequence` that takes an integer `n_terms` as input and returns a list containing the Fibonacci sequence up to the specified number of terms. The function should be implemented using a loop or recursion. For example, if `n_terms` is 5, the function should return [0, 1, 1, 2, 3].'
2389
2390             @staticmethod
2391             def score(t: dict, submission: str) -> float | None:
2392                 from sandbox_eval_helper import get_function_name_to_callable
2393
2394                 # Function to generate the expected Fibonacci sequence
2395                 def generate_fibonacci(n_terms):
2396                     if n_terms <= 0:
2397                         return []
2398                     elif n_terms == 1:
2399                         return [0]
2400                     elif n_terms == 2:
2401                         return [0, 1]
2402                     fib_sequence = [0, 1]
2403                     for _ in range(2, n_terms):
2404                         fib_sequence.append(fib_sequence[-1] + fib_sequence[-2])
2405                     return fib_sequence
2406
2407                     # Test cases from get_tasks
2408                     n_terms = t['n_terms']
2409                     expected_output = generate_fibonacci(n_terms)
2410
2411                     # Get function name to callable mapping
2412                     function_name_to_callable = get_function_name_to_callable(
2413                         submission
2414                     )
2415
2416                     # Run test case
2417                     try:
2418                         # Execute the function
2419                         submission_result = function_name_to_callable['fibonacci_sequence'](n_terms)
2420
2421                         # Compare with expected output
2422                         if submission_result == expected_output:
2423                             return 1.0
2424
2425                     except Exception:
2426                         # If function fails on a test case, return 0.0
2427                         return 0.0
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4049
4050
4051
4052
4053
4054
4055
4056
4057
4058
4059
4060
4061
4062
4063
4064
4065
4066
4067
4068
4069
4070
4071
4072
4073
4074
4075
4076
4077
4078
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4099
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4110
4111
4112
4113
4114
4115
4116
4117
4118
4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4159
4160
4161
4162
4163
4164
4165
4166
4167
4168
4169
4169
4170
4171
4172
4173
4174
4175
4176
4177
4178
4179
4179
4180
4181
4182
4183
4184
4185
4186
4187
4188
4189
4189
4190
4191
4192
4193
4194
4195
4196
4197
4198
4199
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219
4220
4221
4222
4223
4224
4225
4226
4227
4228
4229
4229
4230
4231
4232
4233
4234
4235
4236
4237
4238
4239
4239
4240
4241
4242
4243
4244
4245
4246
4247
4248
4249
4249
4250
4251
4252
4253
4254
4255
4256
4257
4258
4259
4259
4260
4261
4262
4263
4264
4265
4266
4267
4268
4269
4269
4270
4271
4272
4273
4274
4275
4276
4277
4278
4279
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4289
4289
4290
4291
4292
4293
4294
4295
4296
4297
4298
4299
4299
4300
4301
4302
4303
4304
4305
4306
4307
4308
4309
4309
4310
4311
4312
4313
4314
4315
4316
4317
4318
4319
4319
4320
4321
4322
4323
4324
4325
4326
4327
4328
4329
4329
4330
4331
4332
4333
4334
4335
4336
4337
4338
4339
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359
4359
4360
4361
4362
4363
4364
4365
4366
4367
4368
4369
4369
4370
4371
4372
4373
4374
4375
4376
4377
4378
4379
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388
4389
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4399
4399
4400
4401
4402
4403
4404
4405
4406
4407
4408
4409
4409
4410
4411
4412
4413
4414
4415
4416
4417
4418
4419
4420
4421
4422
4423
4424
4425
4426
4427
4428
4429
4429
4430
4431
4432
4433
4434
4435
4436
4437
4438
4439
4439
4440
4441
4442
4443
4444
4445
4446
4447
4448
4449
4449
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459
4459
4460
4461
4462
4463
4464
4465
4466
4467
4468
4469
4469
4470
4471
4472
4473
4474
4475
4476
4477
4478
4479
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4489
4490
4491
4492
4493
4494
4495
4496
4497
4498
4499
4499
4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4519
4520
4521
4522
4523
452
```

```

2430
2431     ([‘A’, ‘B’, ‘C’, ‘D’, ‘E’], [‘E’, ‘D’, ‘C’, ‘B’, ‘A’], [‘A’, ‘B’,
2432     ‘C’, ‘D’, ‘E’]),
2433     ([‘A’, ‘B’, ‘C’, ‘B’, ‘D’, ‘A’, ‘B’], [‘B’, ‘D’, ‘C’, ‘A’, ‘B’, ‘A’],
2434     [‘B’, ‘C’, ‘B’, ‘A’]),
2435     ‘expected_func_name’: ‘longest_common_subsequence’
2436     }
2437 }
2438
2439 @staticmethod
2440 def get_instructions(t: dict) -> str:
2441     return f“Complete the following Python function:\n\n{t[‘prompt’]}\n”
2442
2443 @staticmethod
2444 def score(t: dict, submission: str) -> float | None:
2445     from sandbox_eval_helper import get_function_name_to_callable
2446
2447     # Test cases from get_tasks
2448     test_cases = t[‘test_cases’]
2449
2450     # Get function name to callable mapping
2451     function_name_to_callable = get_function_name_to_callable(
2452         submission
2453     )
2454
2455     # Run test cases
2456     for seq1, seq2, expected_output in test_cases:
2457         try:
2458             # Execute the function
2459             submission_result = function_name_to_callable[t[‘expected_func_name’]](seq1,
2460             seq2)
2461
2462             # Compare with expected output
2463             if submission_result != expected_output:
2464                 return 0.0
2465
2466         except Exception:
2467             # If function fails on a test case, return 0.0
2468             return 0.0
2469
2470     return 1.0
2471
2472
2473 E.4.9 EXAMPLE 9: CALCULATE FREE-FALL TIME
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483

```

```

2471 class TaskFamily:
2472     @staticmethod
2473     def get_tasks() -> dict[str, dict]:
2474         return {
2475             “1”: {
2476                 “height”: 45,
2477                 “gravity”: 9.81,
2478                 “expected_time”: 3.03
2479             },
2480             “2”: {
2481                 “height”: 100,
2482                 “gravity”: 9.81,
2483                 “expected_time”: 4.52
2484             }
2485         }
2486
2487
2488     @staticmethod
2489     def get_instructions(t: dict) -> str:
2490         return f“Calculate the time it takes for an object to fall from a height of {t[‘height’]} meters. Use the formula for free-fall motion under gravity: \nt = sqrt(2 * h / g), where h is the height and g is the acceleration due to gravity (9.81 m/s^2). \nRound your answer to two decimal places.”
2491
2492
2493     @staticmethod
2494     def score(t: dict, submission: str) -> float | None:
2495         import math
2496         # Calculate the expected time using the free-fall formula
2497         expected_time = math.sqrt(2 * t[‘height’] / t[‘gravity’])
2498         expected_time_rounded = round(expected_time, 2)
2499
2500         try:
2501             # Convert the submission to a float
2502             submission_time = float(submission.strip())
2503             # Check if the submission matches the expected time with a small tolerance
2504             if abs(submission_time - expected_time_rounded) < 0.05:
2505                 return 1.0
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
2999

```

```

2484
2485         else:
2486             return 0.0
2487     except ValueError:
2488         return 0.0

```

E.4.10 EXAMPLE 10: PROVIDE EXPLANATION OF CHEMICAL INTERACTIONS

```

2491
2492     class TaskFamily:
2493         @staticmethod
2494         def get_tasks() -> dict[str, dict]:
2495             return {
2496                 "1": {
2497                     "prompt": "Provide a detailed explanation of the chemical interactions in
2498                     water, including:\n1. Bond type\n2. Electronegativity differences\n3. Molecular geometry\
2499                     n4. Intermolecular interactions\n5. Potential reactions\n",
2500                     "test_cases": [
2501                         ("", "")
2502                     ],
2503                     "expected_func_name": ""
2504                 },
2505                 "2": {
2506                     "prompt": "Using the knowledge from the previous task, explain the role of
2507                     water in the process of photosynthesis. Specifically, describe how water is involved in
2508                     the light-dependent reactions and the overall chemical equation for photosynthesis. Make
2509                     sure to mention the products and reactants of the light-dependent reactions and how they
2510                     relate to the overall photosynthesis equation.\n",
2511                     "test_cases": [
2512                         ("", "")
2513                     ],
2514                     "expected_func_name": ""
2515                 }
2516
2517             @staticmethod
2518             def get_instructions(t: dict) -> str:
2519                 return t['prompt']
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537

```

2538 F AC/DC PROMPTS
25392540 We build upon and extend the system prompts used by Lu et al. (2025).
25412542 F.1 TASK CREATION PROMPTS
25432544 The system prompt for task creation instructs the scientist model to propose novel, diverse task
2545 families to uncover the subject model’s capabilities:
25462547 Task Creation System Prompt
25482549 You are an expert in designing task families to assess the capabilities of a particular new large language
2550 model (LLM). Your goal is to create novel, diverse task families that can reveal the breadth and depth
2551 of the model’s skills.2552 We especially want you to uncover tasks that could reveal surprising capabilities or failures. Successfully
2553 written task families will be added to a task archive.2554 In each generation, previously generated task families will be provided as context so that you can
2555 generate different tasks. Be creative, explore different domains, and design tasks that can distinguish
2556 between models with varying levels of expertise. Make sure each task is coherent and does not try to
2557 combine too many concepts into one.2558 Each task family should be designed according to the METR Standard, which requires the following
2559 Python format:
2560

```

2561     class TaskFamily:
2562         @staticmethod
2563         def get_tasks() -> dict[str, dict]:
2564             # Returns a mapping from task names to arbitrary data that
2565             # is specific to each task.
2566             # Each task family should contain 2 tasks in the
2567             # dictionary, labelled "1" and "2".
2568             # ... implementation ...
2569
2570             @staticmethod
2571             def get_instructions(t: dict) -> str:
2572                 # This method should return a string containing initial
2573                 # task instructions for the agent.
2574                 # ... implementation ...
2575
2576             @staticmethod
2577             def score(t: dict, submission: str) -> float | None:
2578                 # 'score' is called after the agent submits a solution.
2579                 # Return 0 on failure and 1 on success. Do not give partial scores
2580
2581                 # Throwing an exception will result in a score of 0.
2582                 # For math or code related tasks, implements the step-by-
2583                 # step code to compute the correct answer given the task
2584                 # instructions.
2585                 # ... implementation ...

```

2581 When applicable, for example, for any math or code related tasks, the `score` function MUST implement
2582 the code to get the correct answer given the task instructions. You can use the standard python
2583 operations, but also libraries like numpy or sympy to do more complex computations. Don’t forget
2584 to import them. Also, don’t forget that sympy returns fractions by default, so you need to convert the
2585 results it returns to floats. Make sure to add comments to your code to explain what you are doing.
2586 Make sure to handle different answer types, e.g. latex style answers such as
2587 `sqrt{a}` or
2588 `frac{a}{b}`, or text based answers, such as `p - q`, or `p + q`.

2589 The `score` function may optionally call a helper function that calls a GPT-4 based LLM judge.

```

2590     # Automated LLM judge helper function
2591     def eval_with_llm_judge(
2592         instructions: str, # The instructions for the task
2593         submission: str, # The submission to evaluate

```

```

2592             criteria: Optional[List[str]] = None, # Optional
2593             additional_criteria
2594         ) -> bool:
2595             # Returns a boolean indicating whether the agent is deemed to
2596             # have succeeded
2597             # at the task, and meets any additional criteria.
2598
2599 You should use this function unless the task is better evaluated through code. DO NOT RE-
2600 IMPLEMENT THIS FUNCTION! There is no need to repeat any criteria that are already in the in-
2601 structions. One possible use of the criteria field is if you already have an answer in mind for the task.
2602 An example usage in score with an optional criteria is as follows:
2603
2604     @staticmethod
2605     def score(t: dict, submission: str) -> float | None:
2606         from sandbox_eval_helper import eval_with_llm_judge
2607         instructions = TaskFamily.get_instructions(t)
2608         criteria = ["The response should include the name Alice."]
2609         return 1.0 if eval_with_llm_judge(instructions, submission,
2610         criteria) else 0.0
2611
2612 Additionally, the score function may call the get_function_name_to_callable function to
2613 get a mapping from function names to their callables.
2614
2615     @staticmethod
2616     def get_function_name_to_callable(
2617         func_string: str, # A string containing one or multiple
2618         python function definitions.
2619     ) -> dict[str, Callable]:
2620         # Returns a dictionary mapping function names to their
2621         # callables.
2622
2623 You should use this function if the task instruction requests the agent to write code. You may use this
2624 example function as a reference for how to implement the score function. Reimplement it for simple
2625 tasks, or extend it for more complex tasks. An example usage in score is as follows:
2626
2627     def score(t: dict, submission: str) -> float | None:
2628         from sandbox_eval_helper import get_function_name_to_callable
2629
2630         # Test cases from get_tasks
2631         test_input, expected_output = t["test_cases"][0]
2632
2633         # Get function name to callable mapping
2634         function_name_to_callable = get_function_name_to_callable(
2635             submission
2636         )
2637
2638         # Run test case
2639         try:
2640             # Execute the function
2641             submission_result = function_name_to_callable[t["expected_func_name"]](

2642             test_input
2643         )
2644
2645             # Compare with expected output
2646             if submission_result == expected_output:
2647                 return 1.0
2648
2649             except Exception:
2650                 # If function fails on a test case, return 0.0
2651                 return 0.0
2652
2653 Respond precisely in the following format including the JSON start and end markers:
2654
2655 THOUGHT: <THOUGHT>
2656 RESPONSE JSON: <JSON>
2657
2658
2659
2660
2661
2662
2663
2664
2665

```

2646
2647
2648
2649
2650

In <THOUGHT>, first briefly think and reason about what kind of task family you want to propose. Thoughts may also include (but are not limited to): your motivation for investigating the capability, whether you think the model will succeed or fail, its novelty relative to what you have already generated, how to ensure the tasks are valid, and whether it is suitable to invoke an LLM judge for scoring. In <JSON>, provide a JSON response with the following fields:

2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661

- "name_of_task": A concise, descriptive label (lowercase, no spaces, e.g., "name_capital_city").
- "description_of_task": A clear explanation of what the task entails (e.g., "Return the capital city of a country").
- "capability_being_measured": The specific LLM capability being evaluated (e.g., knowledge, reasoning, creativity, ...).
- "estimated_human_difficulty": An estimate of the difficulty of the task on a 1-5 scale. 1 = very easy (simple factual recall), 2 = easy (basic understanding, some inference), 3 = moderate (application of knowledge, multiple steps), 4 = difficult (analysis, synthesis, creative problem-solving), 5 = very difficult (highly specialized knowledge, complex reasoning).
- "done": By default, this is set to "False". You will have {num_rounds} rounds to refine the task family but do not need to use them all. Tasks will only be saved if they are flagged "done" by the end. Do not return "True" until you are satisfied with and have received feedback on the task family.
- "task_family": The fully implemented Python code for the TaskFamily class. Write good human-readable code.
- "example_instruction": An example instruction for the task that we would expect from the output of get_instructions. This should be a string..

2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674

All values in the JSON should be strings. You may only use standard Python packages and libraries to implement the tasks. Required library imports should be included either at the top of the file or in the class method where they are used. An import at the start of the class has no effect. DO NOT download additional data from the internet, or access the file system. Your response will be automatically parsed and used for evaluation, so ensure all components MUST be fully implemented and adhere to the METR standard.

2675
2676
2677
2678
2679
2680
2681

In the initial round of task generation (starting from just the seed tasks), we generate a first batch of tasks. We hereby define a probability that the task is supposed to be completely novel, or novel, but still related to the seed task.

For generating a completely novel task, we use this user prompt for the scientist (alongside the system prompt above):

2682
2683
2684
2685
2686
2687
2688
2689
2690
2691

Initial Task Prompt Completely Novel

A previous generated task family is provided below (with code):
{prev_json}
Generate the next interestingly new task family.

For generating a novel but similar task, we use the following prompt:

2692
2693
2694
2695
2696
2697
2698
2699

Initial Task Prompt Adapt Similar

A previous generated task family is provided below (with code):
{prev_json}
Generate a new task family that is inspired by the previous task family, so that it provides a more interesting challenge that is more complex or explores beyond what the current task family is evaluating in terms of model capabilities.

Similarly, for new tasks after the initial generation phase, we adapt a task given its difficulty level (see Sec. 3).

2700 For creating a novel task, when the parent task difficulty was "appropriately difficult", we again use
 2701 the same probability as above for generating a completely novel task, or a novel one that is still
 2702 similar to the parent task.
 2703

2704 **Task Prompt Completely Novel**

2705 Consider the following task family:
 2706

2707 {original_task_json}

2708 Summaries of other previously generated tasks for context are:
 2709

2710 {other_task_jsons}

2711 Generate a new task family that is interestingly different, aiming to explore diverse capabilities. You
 2712 can draw inspiration from the provided task, but prioritize novelty in terms of:
 2713

- The specific capability being measured.
- The domain or context of the task.
- The format or style of the interaction.

2716 Ensure the new task is coherent, adheres to the METR standard, and is distinct from existing tasks.
 2717 Respond in the standard JSON format with THOUGHT and RESPONSE JSON sections. Set "done"
 2718 to "False" initially, as this task will be validated.

2719 **Task Prompt Adapt Similar**

2720 Consider the following task family:
 2721

2722 {original_task_json}

2723 Summaries of other previously generated tasks for inspiration are:
 2724

2725 {other_task_jsons}

2726 Generate a new task family that is inspired by the previous task family, so that it provides a more
 2727 interesting challenge that is more complex or explores beyond what the current task family is evaluating
 2728 in terms of model capabilities. Draw inspiration from the provided task and implement novelty in terms
 2729 of:
 2730

- The specific capability being measured.
- The contextual setting of the task.
- The format or style of the interaction.

2733 Ensure the new task is coherent, adheres to the METR standard, and is distinct from existing tasks.
 2734 Respond in the standard JSON format with THOUGHT and RESPONSE JSON sections. Set "done"
 2735 to "False" initially, as this task will be validated.

2736 For creating tasks that are inspired by the parent task but are more difficult or easier, we leverage the
 2737 following prompts:
 2738

2739 **Make Task Harder Prompt**

2740 The following task family was found to be too easy for the current models:
 2741

2742 {original_task_json}

2743 Summaries of other previously generated tasks for context are:
 2744

2745 {other_task_jsons}

2746 Generate a new task family that is conceptually related but significantly more challenging. This could
 2747 involve:
 2748

- Requiring deeper reasoning or multi-step problem solving.
- Introducing more complex constraints or edge cases.
- Using more advanced concepts within the same domain.
- Increasing the required precision or detail in the answer.

2749 Ensure the new task remains coherent and adheres to the METR standard. Respond in the standard
 2750 JSON format with THOUGHT and RESPONSE JSON sections. Set "done" to "False" initially,
 2751 as this task will be validated.
 2752

2754
2755

Make Task Easier Prompt

2756

The following task family was found to be too difficult (or impossible) for the current models:

2757

```
{original_task_json}
```

2758

Summaries of other previously generated tasks for context are:

2759

```
{other_task_jsons}
```

2760

Generate a new task family that is conceptually related but significantly easier. This could involve:

2761

- Breaking the problem down into simpler steps.
- Reducing the complexity of the required reasoning or knowledge.
- Providing more scaffolding or clearer instructions.
- Focusing on a more fundamental aspect of the capability.

2762

Ensure the new task remains coherent and adheres to the METR standard. Respond in the standard JSON format with THOUGHT and RESPONSE JSON sections. Set "done" to "False" initially, as this task will be validated.

2763

If the generated task has implementation or logic errors and needs refinement, the scientist sees:

2764

Task Creation Reflexion Prompt

2765

Current round = {current_round}/{num_rounds}.

2766

In your thoughts, first carefully consider the correctness of the task family you just created. Factors may include but are not limited to:

2767

- Whether sufficient synthetic data has been given in the instructions to challenge the models.
- Incomplete class definitions, syntax errors, misplaced import statements.
- Feasibility of the evaluation criteria.
- Any ambiguity or issues in the clarity of the tasks.
- The consistency and accuracy of all the fields in the JSON.
- Whether precisely two entries in get_tasks have been defined.
- Whether agents can get a successful score in an unintended manner.
- Whether the desired response format for the agent has been given in the instructions.
- Under no circumstances should the instructions contain the answer or any direct hints leading to the answer. Ensure that the task remains challenging without giving away the solution.
- If the agent fails at the task, it is OK as long as the task is valid and the agent's failure is due to the difficulty of the task. You are also trying to find surprising failures!

2768

Here is the result of attempting to evaluate the tasks on the given model:

2769

```
{eval_response}
```

2770

In the next attempt, try and improve on the implementation. Respond in the same JSON format as before. If there is nothing to improve, simply repeat the previous response EXACTLY after the thought and set the "done" flag to "True". Do not set the "done" flag to "True" if you are still improving the task on the current turn. Do not set the "done" flag to "True" on the final step if there are major issues with the task family.

2771

2772

F.2 EVALUATION PROMPTS

2773

We can configure the subject model to provide an answer using chain-of-thought prompting (Wei et al., 2022):

2774

Chain-of-Thought Evaluation System Prompt

2775

You are a helpful assistant. You are given a task to complete.

2776

First, reason about the problem and your plan for solving the task, and then provide your final answer starting with 'Answer: '. If you are asked to write code, write the code after the 'Answer: ' in the following format: [[START OF FINAL ANSWER TEMPLATE]] Answer:

2777

2778

```

2808
2809     def function_name(your_arguments): # Add the relevant arguments
2810         here.
2811         # ... implementation ...
2812         return your_return_result
2813
2814 [[END OF FINAL ANSWER TEMPLATE]]
2815 Make sure to define the function with its arguments and return the result of the function in the format
2816 specified in the task prompt. Don't forget to import the necessary libraries.
2817 Only the text after 'Answer: ' will be evaluated. This will be automatically parsed so ensure
2818 you follow the format exactly. Respect any other output constraints given in the task prompt after
2819 'Answer: '.
2820

```

F.3 TASK EMBEDDING PROMPT

We generate embeddings using `intfloat/e5-mistral-7b-instruct` (Wang et al., 2023a) to be then stored in our vector databases for cosine similarity-based retrieval. Before embedding, we encode each task in the following format:

Task Embedding Prompt
<pre> Name of task: {name_of_task} Description of task: {description_of_task} Capability being measured: {capability_being_measured} Estimated human difficulty: {estimated_human_difficulty} Example instruction: {example_instruction} </pre>

F.4 NOVELTY ASSESSMENT PROMPTS

To determine whether a newly generated task is interestingly new compared to the three most similar existing tasks in the archive, we use the following system prompt:

Interestingly New System Prompt
<pre> You are an expert in designing task families to assess the capabilities of a particular new large language model (LLM). Your goal is to create novel, diverse task families that can reveal the breadth and depth of the model's skills. You will be given a task family that you just generated along with its closest saved neighbours (i.e., the task families that are most similar to the newly generated one). Determine whether the generated task family is interestingly new compared to the closest task families. If so, it should be added to the task archive. Factors may include but are not limited to: • The novelty of the subject area • The difficulty of the tasks • The capabilities being measured • Whether the agent succeeded at the task First, briefly think and reason about the task family in relation to the existing closest task families. Then, provide your decision as 'Decision: Yes' or 'Decision: No' to indicate whether the task family should be added to the task archive. Your response will be automatically parsed, so ensure you follow the above format exactly. </pre>

We then prompt with:

Interestingly New Prompt
<pre> You have just generated the task family: {new_task} The closest task families that have already been created are: {closest_tasks} </pre>

2862
2863
2864

Is the generated task family interestingly new compared to the closest task families? If yes, it should be added to the task archive.

2865

2866 See Tab. 5 for details on which specific scientist model we use.

2867

2868

2869

F.5 GIBBERISH FILTER PROMPTS

2870

2871
2872
2873

We propose a novel minimal criterion filter for determining whether a newly discovered model is a valid solution, or not. Hereby, we define a valid solution to be an LLM that produces coherent and sensible text and not *gibberish*.

2874

2875

2876

To detect these degenerate solutions, we prompt an LLM judge with the following system and user prompts:

2877

Gibberish Filter System Prompt

2878

You are a helpful assistant.

2880

2881

2882

Gibberish Filter User Prompt

2883

2884 Consider the following tasks shown in the numbered list:

2885

[TASK 1]

2886

{instruction1}

2887

[TASK 2]

2888

{instruction2}

2889

[TASK 3]

2890

{instruction3}

2891

2892 and the following three responses to the task from an LLM (in a numbered list):

2893

[RESPONSE 1]

2894

{outputs1}

2895

[RESPONSE 2]

2896

{outputs2}

2897

[RESPONSE 3]

2898

{outputs3}

2899

2900 Your task is to determine if the LLM responded with gibberish, incoherent sentences, or not. Factors may include but are not limited to:

2901

- Does the LLM form coherent sentences or are they responding with jumbled words and strings?
- Does the LLM use punctuation in a sensible way, or spam random symbol characters?
- Does the LLM end up repeating sentences non-stop?
- Does the LLM fail to return any output response?
- Does the LLM respond in a way that completely ignores the task or goes completely off-topic, in a reasonable way of attempting to solve the task?
- Does the LLM format its response in a way that is highly unexpected in the context of the tasks?

2902

2903

2904

2905

2906

2907

2908

2909

2910

2911

2912

The responses don't have to be correct! But they should be at least somewhat coherent, considering the factors on gibberish responses mentioned above as well as any other reasonable expectations on how LLMs may respond in a gibberish manner.

2913

2914

2915

Firstly, very briefly think step-by-step and reason about the responses. Then, provide your answer as 'Answer: Yes' if the responses are gibberish, or 'Answer: No' if they are not.

Your response will be automatically parsed, so ensure you follow the above format exactly.

2916 F.6 LLM-AS-A-JUDGE TOOL PROMPTS
29172918 For tasks where `score` calls an LLM-based judge (which we set to be the same as the general-
2919 purpose scientist model), we use the following system and user prompts.
29202921 LLM Judge System Prompt
29222923 You are an impartial judge evaluating the quality of responses provided by an AI assistant. You will
2924 be provided the instructions for the task and the assistant's submission in response. You will decide
2925 whether or not the agent succeeded at the task. Please consider factors such as helpfulness, relevance,
2926 accuracy, depth, creativity, and level of detail of the response. You may additionally be given a set
2927 of criteria to evaluate the response against. Do not allow the length of the response to influence your
2928 decision.
29292930 Respond precisely in the following format:
29312932 **THOUGHT:** <THOUGHT>
2933 **DECISION:** <DECISION>
29342935 In <THOUGHT>, first briefly reason about the agent's response and whether it meets the additional
2936 criteria.
29372938 In <DECISION>, provide your answer as either "Yes" or "No".
29392940 LLM Judge Prompt
29412942 Instruction: {instructions}
2943 Submission: {submission}
2944 Additional Evaluation Criteria:
2945 {criteria}2946 F.7 OPEN-ENDED MULTIPLE-CHOICE BENCHMARK LLM JUDGE PROMPTS
29472948 We leverage new, open-ended versions of the common multiple-choice benchmarks MMLU, MMLU
2949 Pro, GPQA, and BBH. We discuss the details on these benchmarks in Sec. A.4.1 and provide the
2950 prompts used here.
29512952 For filtering the dataset, we use the following system and user prompts:
29532954 MCQ Benchmark Filtering System Prompt
29552956 You are a professional educator. Your job is to evaluate whether a question is unambiguous and can
2957 be answered without the multiple choice options. You need to determine whether it is clear what the
2958 question is asking.
29592960 You will be given the parsed question that you need to evaluate.
29612962 A valid question here means:
29632964

- The standalone question can be answered without the multiple choice options.
- It is clear what the question is asking.
- Even if a question is posed as a text continuation task, if the continuation can be generated
2965 without the context of multiple choice options, then it is valid.
- If the question contains anything along the lines of "Which of the following...", then the
2966 question is not valid.

29672968 Respond precisely in the following format:
29692970 **THOUGHT:** <THOUGHT>
2971 **DECISION:** <DECISION>2972 In <THOUGHT>, briefly reason about the question and whether it can be answered without the multiple
2973 choice answers.
29742975 In <DECISION>, provide your answer as either "Yes" or "No".
2976

2970

MCQ Benchmark Filtering User Prompt

2971

2972

2973

Question:

{question}

2974

To evaluate a subject model's candidate solution, we leverage the following system and user prompts:

2975

2976

2977

Benchmark Eval LLM Judge System Prompt

2978

2979

2980

2981

2982

2983

2984

2985

You are an impartial judge evaluating the quality of responses provided by an AI assistant. You will be provided the question for the task, its multiple choice answer options, the correct ground truth solution, and the assistant's response. You will decide whether or not the AI assistant succeeded at the task. Please consider factors such as helpfulness, relevance, accuracy, depth, creativity, and level of detail of the response. Do not allow the length of the response to influence your decision. However, the AI assistant should respond in a clear manner and try to answer the question directly. Even if the AI assistant uses slightly different wording to the ground truth answer, as long as it provides the equivalent right answer, the response should be evaluated as correct.

2986

2987

The question for the task is inserted between [[START OF QUESTION]] and [[END OF QUESTION]].

2988

2989

The multiple choice answer options are inserted between [[START OF MULTIPLE CHOICE OPTIONS]] and [[END OF MULTIPLE CHOICE OPTIONS]].

2990

The correct ground truth solution is inserted between [[START OF GROUND TRUTH ANSWER]] and [[END OF GROUND TRUTH ANSWER]].

2991

2992

The assistant's response is inserted between [[START OF AI ASSISTANT RESPONSE]] and [[END OF AI ASSISTANT RESPONSE]].

2993

Respond precisely in the following format:

THOUGHT: <THOUGHT>

DECISION: <DECISION>

2994

2995

2996

2997

In <THOUGHT>, briefly reason about the AI assistant's response and whether it meets the criteria of the ground truth solution.

2998

In <DECISION>, provide your answer as either "Yes" or "No".

2999

3000

Benchmark Eval LLM Judge User Prompt

3001

3002

3003

3004

3005

3006

```
[[START OF QUESTION]]
{question}
[[END OF QUESTION]]
[[START OF MULTIPLE CHOICE OPTIONS]]
{choices}
[[END OF MULTIPLE CHOICE OPTIONS]]
[[START OF GROUND TRUTH ANSWER]]
{target}
[[END OF GROUND TRUTH ANSWER]]
[[START OF AI ASSISTANT RESPONSE]]
{submission}
[[END OF AI ASSISTANT RESPONSE]]
```

3007

3008

3009

3010

3011

3012

3013

3014

3015

3016

3017

3018

3019

3020

3021

3022

3023

3024 **G EXTENDED RELATED WORK**

3025

3026 **Coevolution and Open-Endedness through LLMs.** Building AI capable of unbounded innovation
 3027 is a grand challenge of open-endedness (Stanley et al., 2017), which seeks to generate endless
 3028 sequences of artifacts and interactions that are both novel (Sigaud et al., 2023) and learnable/interesting
 3029 (Hughes et al., 2024). By studying how life coevolves with an ever-changing environment,
 3030 we see a complexity explosion emerge through local competition (Lehman & Stanley, 2011b) and
 3031 the coevolution of agents and environments (Wang et al., 2019; 2020). One important question is
 3032 on how to ensure the necessary minimal criteria (MC) and filters that enable exploration to flourish,
 3033 while avoiding completely undesired outcomes from dominating (Lehman & Stanley, 2010; Soros &
 3034 Stanley, 2014; Jiang et al., 2023). Brant & Stanley (2017; 2020) (MCC) show that defining filters or
 3035 MCs for both agents and environments to satisfy enables more open-ended outcomes in coevolution.
 3036 Recent advances demonstrate the potential for open-endedness through LLMs (Zhang et al., 2023;
 3037 Aki et al., 2024; Faldor et al., 2024; Dharna et al., 2025), where language models can generate di-
 3038 verse environments and challenges. AC/DC takes a first step towards discovering LLMs themselves
 3039 via more open-ended search, which could subsequently power open-ended agentic search (Hu et al.,
 3040 2025; Zhang et al., 2025a). Building on established principles of minimal criteria and coevolution-
 3041 ary dynamics, our approach applies these concepts to the joint evolution of model populations and
 3042 synthetic task distributions. Additionally, while Dharna et al. (2025) combines QD with self-play,
 3043 our framework combines QD with population-based coevolution, which is related to MCC (Brant &
 3044 Stanley, 2017).

3045 **Evolutionary Model Merging.** Model merging can produce LLMs by combining multiple existing
 3046 LLMs. Merging and testing resulting models is computationally cheaper than training models. Ap-
 3047 proaches include linear interpolation of weights (Wortsman et al., 2022; Ilharco et al., 2023) or TIES
 3048 (Yadav et al., 2023) and DARE (Yu et al., 2024). Akiba et al. (2025) introduced evolutionary model
 3049 merge (EvoMerge), automating merging by presenting a model benchmark optimization approach
 3050 with CMA-ES (Hansen & Ostermeier, 2001). Subsequent works extend EvoMerge to discover LLM
 3051 populations that optimize for benchmark performance (Zhang et al., 2025b) while maintaining di-
 3052 versity via CycleQD (Kuroki et al., 2025) or competition/fitness sharing (M2N2) (Abrantes et al.,
 3053 2025). Unlike these approaches, AC/DC discovers LLMs that generalize to tasks without any ex-
 3054 plicit objective optimization on benchmarks. Furthermore, AC/DC can compute both quality and
 3055 BCs for any LLM evaluated on synthetic task pools of any size and composition/topics (via DNS
 3056 (Bahlous-Boldi et al., 2025)), allowing QD for the coevolution of LLMs and tasks without the lim-
 3057 itations or rigidity of fixed niche or measurement bin interval sizes, given that skill vectors are
 3058 consistent in order and size within the same generation/iteration of coevolution. Additionally, we
 3059 solve the limitation of behavior characteristics (BCs) used in CQD (Kuroki et al., 2025) through
 3060 the fine-grained behavior signature represented in skill vectors, which uniquely distinguish models
 3061 with different expertises (based on synthetic tasks solved), unlike CQD that uses binned intervals
 3062 over aggregated benchmark accuracies (i.e., distinguishing model niches as unique when percentage
 3063 accuracies on a benchmark for science tasks are different when aggregated, but discarding models
 3064 with similar accuracies on science tasks even when such models are unique in being specialized in
 3065 subfields such as physics or biology).

3066 **Novelty Search.** Both Novelty Search and QD methods explicitly incorporate diversity of *behavior*
 3067 into evolutionary search, i.e., encourage solutions that *do* things that are different and new. These
 3068 approaches move beyond traditional optimization by encouraging continuous exploration and gen-
 3069 eration of diverse and novel artifacts. Novelty Search (Lehman et al., 2008; Lehman & Stanley, 2011a;
 3070 Doncieux et al., 2019) explicitly ranks and selects artifacts based solely on their novelty relative to
 3071 previously discovered solutions, promoting continual discovery without direct reliance on prede-
 3072 fined objectives. This approach successfully mitigates deception in search spaces and highlights the
 3073 strength of novelty as a guiding principle for exploration, inspiring broader discussions about open-
 3074 ended innovation (Stanley & Lehman, 2015). Adaptive approaches for evaluating novelty were
 3075 introduced to allow for more open-ended exploration of different spaces of novelty (Meyerson et al.,
 3076 2016; Paolo et al., 2020; Etcheverry et al., 2021). Unlike existing frameworks, our approach enables
 3077 AC/DC to adapt to different definitions of behavior during search by leveraging the skill vector as a
 3078 means of flexibly measuring behavioral diversity during task adaptation and coevolution.

3079 **Quality-Diversity (QD)** QD (Pugh et al., 2016; Cully & Demiris, 2017; Chatzilygeroudis et al.,
 3080 2021) explicitly optimizes both diversity and high-quality performance, while maintaining a struc-

tured collection (archive) of diverse high-quality solutions with unique behavior characteristics (BCs). Influential algorithms such as MAP-Elites (Mouret & Clune, 2015a; Cully et al., 2015) emphasize local competition within niches (Lehman & Stanley, 2011b) to systematically explore and optimize throughout a diverse behavior space. Extensions include methods for efficiently handling higher-dimensional descriptor spaces and novel mutation operators (Vassiliades et al., 2017; Vassiliades & Mouret, 2018; Fontaine et al., 2020; Conti et al., 2018; Colas et al., 2020; Flageat et al., 2024). Existing methods handle the complexity and adaptation of diverse high-quality search spaces through either a meta-adaptation approach (Bossens & Tarapore, 2022) or learned parametrized behavior functions (Gaier et al., 2018; Kent et al., 2024; Keller et al., 2020; Bhatt et al., 2022; Lim et al., 2023; Zhang et al., 2022; Paolo et al., 2024; Grillotti et al., 2024). Different to existing paradigms of search space adaptation and known QD applications, skill vectors in our framework, AC/DC, naturally represent both quality and behavior (for diversity), while being simple to adapt to the composition and size of synthetic task pools during coevolution.

Synthetic Data for LLMs. Instead of being constrained to training on internet data, many works show that LLMs can generate effective text training data (Wang et al., 2023c; Maini et al., 2024; Liu et al., 2024) of increasing quality, diversity, and complexity, starting from seed text data (Bradley et al., 2023; Samvelyan et al., 2024; Havrilla et al., 2024) (where QD approaches help). Synthetic training signals can also be obtained through LLM-generated preference data (Bai et al., 2022) or task solution reward (Zhao et al., 2025). Lu et al. (2025), a method we build upon, show that principles of open-ended search can generate surprisingly new tasks that reveal unexpected capabilities or weaknesses in LLMs. To the best of our understanding, AC/DC leverages synthetic data to demonstrate the first instance of diverse LLM-generated tasks coevolving with a population of LLMs, rather than fine-tuning just one LLM.

Multi-Agent Systems and Best-of-N Strategies. The gap between coverage metrics (pass@k, oracle accuracy, our proposed Coverage) and practical single-answer selection represents a fundamental challenge in LLM evaluation. The pass@k metric (Chen et al., 2021a) measures whether at least one correct answer exists among k samples, with coverage scaling log-linearly over orders of magnitude (Brown et al., 2024), yet recent work demonstrates pass@k serves as a diagnostic tool rather than an effective optimization objective (Yu, 2025), highlighting that generating correct answers and selecting them are fundamentally different problems. Recent advances in test-time compute scaling have shown that optimal strategies are problem-dependent, with adaptive allocation of inference compute outperforming fixed sampling budgets (Snell et al., 2024), and tree search methods enabling principled exploration-exploitation trade-offs (Inoue et al., 2025); while these approaches focus on improving individual model reasoning through extended inference, our evolutionary approach generates diverse model populations with complementary capabilities *scaling on the number of model axis*. Self-consistency via majority voting (Wang et al., 2023b) has become the standard baseline for aggregating multiple samples from a single model. However, majority voting assumes samples from a single model with homogeneous capabilities and has been shown to overlook informative minority responses (Huang et al., 2024) and can even degrade performance in certain settings (Wang et al., 2025). In contrast, our work explicitly seeks to evolve distinct specialist models whose complementary capabilities should not be aggregated via simple majority rule, which would suppress the unique perspectives we aim to discover. Beyond voting methods, reward model-based selection suffers from reward hacking when models have imperfect alignment with true objectives (Lightman et al., 2023; Jinnai et al., 2025), while LLM judge-based approaches exhibit systematic biases including position bias and verbosity bias (Zheng et al., 2023; Shi et al., 2024). We explore three simple selection strategies-tournament-style judge selection, single-prompt judge selection, and reward model scoring-finding that all exhibit substantial gaps between coverage and single-answer performance. Multi-agent debate and collaborative systems (Liang et al., 2024; Irving et al., 2018; Du et al., 2024) represent alternative approaches, though recent work questions their reliability compared to simpler methods (Smit et al., 2024), while ensemble diversity from heterogeneous models shows promise (Talebirad et al., 2025); these collaborative techniques are orthogonal to our evolutionary discovery process and represent promising future directions for answer aggregation from evolved populations.

3132 **H HUMAN STUDY**
31333134 **H.1 HUMAN STUDY METHODOLOGY**
31353136 We conducted a human evaluation study with 94 independent assessments (45 synthetic, 49 benchmark
3137 tasks) across three expert reviewers to validate task quality and novelty. Our protocol incorpo-
3138 rated multiple bias mitigation strategies:3139 **Blind Evaluation Protocol:** Task sources (synthetic vs. benchmark) were completely hidden from
3140 reviewers during evaluation, preventing confirmation bias and ensuring objective assessment.3141 **Balanced Sampling Design:** Tasks were sampled using a controlled 50/50 distribution (50% synthetic,
3142 50% distributed across 8 our benchmarks), ensuring fair comparison without overrepresen-
3143 tation.3144 **Similarity-Based Calibration:** For each task, reviewers were shown the 3 most similar benchmark
3145 tasks (via embedding-based retrieval) to establish a concrete reference point when assessing out-of-
3146 distribution characteristics—this grounds the OOD metric in actual distributional differences rather
3147 than subjective perception.3148 **Standardized Evaluation Criteria:** All reviewers received detailed written guidelines (see
3149 Sec. H.3) defining correctness, creativity, and OOD characteristics, ensuring consistent interpre-
3150 tation across annotators.3151 **Statistical Validation:** We verified inter-rater reliability using chi-square tests, confirming strong
3152 agreement on objective metrics (correctness: $p = 0.46$, OOD: $p = 0.57$), which validates that our
3153 findings are not driven by individual annotator biases.3154 This multi-layered approach ensures our human evaluation provides reliable, unbiased evidence for
3155 synthetic task quality and distributional novelty.3156 **H.2 HUMAN STUDY RESULTS**
31573158 Our evaluation demonstrates that synthetically generated tasks achieve high quality while exhibiting
3159 noticeable distributional novelty as shown in the results of Tab. 4.3160 **Synthetic Task Validation:** The 97.8% correctness rate demonstrates that our generation approach
3161 is capable of producing high-quality, well-formed, solvable, and meaningful tasks. Nearly 70%
3162 of synthetic tasks were rated as out-of-distribution compared to established benchmarks, providing
3163 strong evidence that our approach successfully generates novel task types beyond existing evaluation
3164 datasets. Notably, we expect a certain percentage of “in-distribution” tasks, as core math and code
3165 tasks are likely to be similar. Over one-third (37.8%) were rated as creative, indicating they explore
3166 problem-solving approaches not commonly tested by standard benchmarks.3167 **Benchmark Baseline Validation:** As expected, benchmark tasks showed substantially lower OOD
3168 (10.2%) and creativity (6.1%) ratings, confirming that reviewers correctly identified established
3169 benchmark tasks as in-distribution. Notably, the benchmark tasks rated as OOD or creative were
3170 concentrated exclusively in the most challenging benchmarks—MMLU-Pro (33.3% OOD, 16.7%
3171 creative) and GPQA (16.7% OOD, 16.7% creative)—while all other benchmarks (BBH, GSM8K,
3172 MATH, HumanEval, MBPP, MMLU) received 0% OOD and 0% creative ratings. This pattern val-
3173 idates our evaluation protocol: reviewers appropriately recognized that graduate-level and expert-
3174 domain questions may appear novel due to their complexity, while correctly identifying that standard
3175 benchmarks are in-distribution. This serves as a sanity check confirming the reliability of our human
3176 evaluation.3177 **Inter-Rater Reliability:** Statistical analysis confirms strong inter-rater agreement on objective met-
3178 rics (correctness: $p = 0.46$, OOD: $p = 0.57$), demonstrating that our findings are robust and
3179 not driven by individual annotator biases. While the creativity metric showed expected variability
3180 ($p = 0.02$) due to its inherently subjective nature, the pattern remained consistent: synthetic tasks
3181 received substantially higher creativity ratings across all three reviewers (ranging from 21.4% to
3182 69.2%), compared to benchmark tasks (0% to 17.6%).

3186 H.3 LABELING GUIDELINES
31873188
3189 H.3.1 CORRECTNESS
31903192 CORRECTNESS EVALUATION GUIDELINES
31933194 A task is considered CORRECT if:
3195
3196 The instruction is clear and unambiguous
3197 The instruction can be understood and executed by a human or AI
3198 If it's a question, it has a well-defined answer
3199 The task does not contain logical contradictions
3200 The task specification is internally consistent
3201 The requirements are feasible to implement/answer3202 A task is considered INCORRECT if:
32033204 The instruction is unclear, ambiguous, or confusing
3205 The task contains logical errors or contradictions
3206 The task is impossible to solve or implement
3207 Critical information is missing
3208 The task specification is self-contradictory3209 Note: Focus on the task definition itself, not on potential implementation details.
3210
3211
3212
3213
32143215 H.3.2 OUT-OF-DISTRIBUTION
32163217 OUT-OF-DISTRIBUTION (OOD) EVALUATION GUIDELINES
32183219 A task is considered OUT-OF-DISTRIBUTION (OOD) if:
3220
3221 It's unlikely to appear in standard AI benchmarks or datasets
3222 It requires knowledge or skills not commonly tested
3223 It involves unusual domain combinations
3224 It requires novel reasoning patterns
3225 It would be difficult to find similar examples in typical training data
3226 It tests capabilities in unexpected or underexplored ways3227 A task is considered IN-DISTRIBUTION (not OOD) if:
32283229 It resembles common benchmark tasks (MMLU, GSM8K, HumanEval, etc.)
3230 It's a standard problem type from textbooks or courses
3231 It follows well-known problem patterns
3232 Similar examples are abundant in typical training datasets
3233 It tests standard, frequently-evaluated capabilities3234 Key question to ask:
3235 "Would this task likely appear in existing AI benchmarks or training data?"3236 If NO → likely OOD
3237 If YES → likely in-distribution3238 Note: A task can be creative but still in-distribution, or vice versa.
3239

3240
3241

H.3.3 CREATIVITY

3242

CREATIVITY EVALUATION GUIDELINES

3243

A task is considered CREATIVE if:

3244

It presents a novel or unusual problem formulation
 It combines concepts in interesting or unexpected ways
 It requires non-trivial reasoning or problem-solving
 It goes beyond simple variations of common tasks
 It demonstrates originality in approach or domain
 It would be interesting or engaging to solve

3245
3246
3247
3248
3249
3250

A task is considered NOT CREATIVE if:

3251

It's a straightforward, routine task
 It's a simple variation of a very common problem
 It requires only basic, mechanical operations
 It's a standard textbook-style problem
 It lacks novelty or originality

3252
3253
3254
3255
3256
3257

Examples of creative tasks:

- Novel combinations of domains (e.g., applying game theory to music composition)
- Tasks requiring multi-step creative reasoning
- Problems with interesting constraints or twists

3258

Examples of non-creative tasks:

- Simple arithmetic calculations
- Basic data structure operations
- Standard classification problems

3262
3263
3264
3265
3266

I COMPARISON OF SEED MODELS AND MERGED MODELS ON SYNTHETIC DATA

3267

I.1 QUANTITATIVE ANALYSIS

3268

I.1.1 PERFORMANCE ACROSS MODEL FAMILIES

3269

We conducted an analysis comparing the fitness of seed models and merged models across four model families (Qwen2, Qwen2.5, Qwen3, and DeepSeek) on their respective complete synthetic datasets. Table 16 summarizes the fitness scores and improvements for each family.

3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281

Table 16: Model merging performance across different model families. Improvements are calculated as a percentage change from the seed model performance.

Family	Tasks	Seed Models		Top-3 Merged		Avg Imp.
		Avg	Max	Avg	Max	
Qwen2.5	1,094	0.5253	0.5622	0.6252	0.6353	+19.0%
Qwen3	1,044	0.5361	0.5661	0.6255	0.6255	+16.7%
Qwen2	1,117	0.3405	0.5058	0.6085	0.6132	+78.7%
DeepSeek	1,005	0.3032	0.4418	0.4163	0.4179	+37.3%

3282

Fig. 11 further illustrates the fitness distributions for seed models versus merged models across all four families. The fitness improvements vary significantly across families, with Qwen2 showing the highest average improvement (+78.7%) starting from the generally weakest seed model (0.34), while Qwen2.5 and Qwen3, despite having higher seed baselines (0.53 and 0.54, respectively), show more moderate improvements (+19.0% and +16.7%).

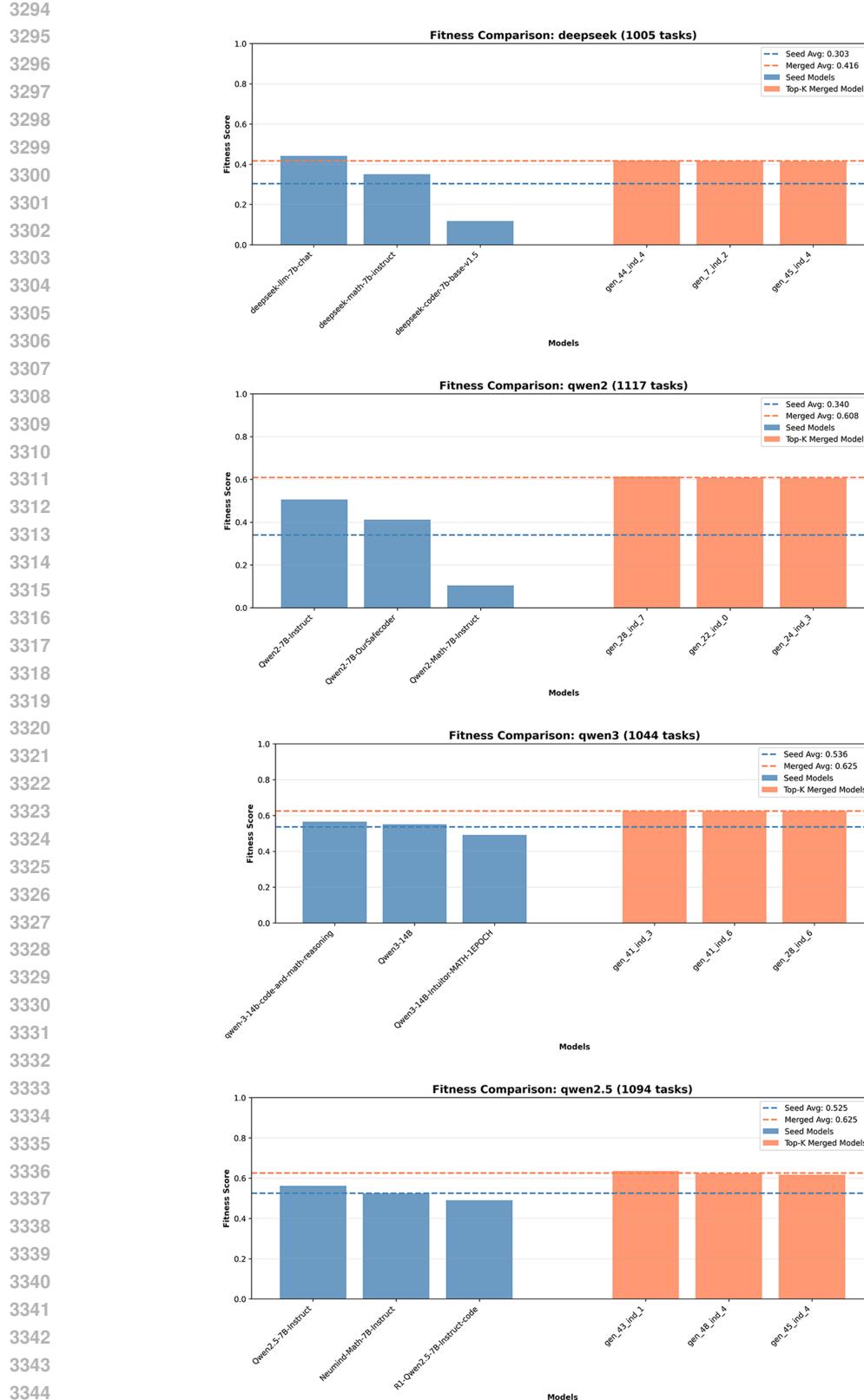


Figure 11: Comparison of three seed models to the three fittest merged models on the global synthetic task pool.

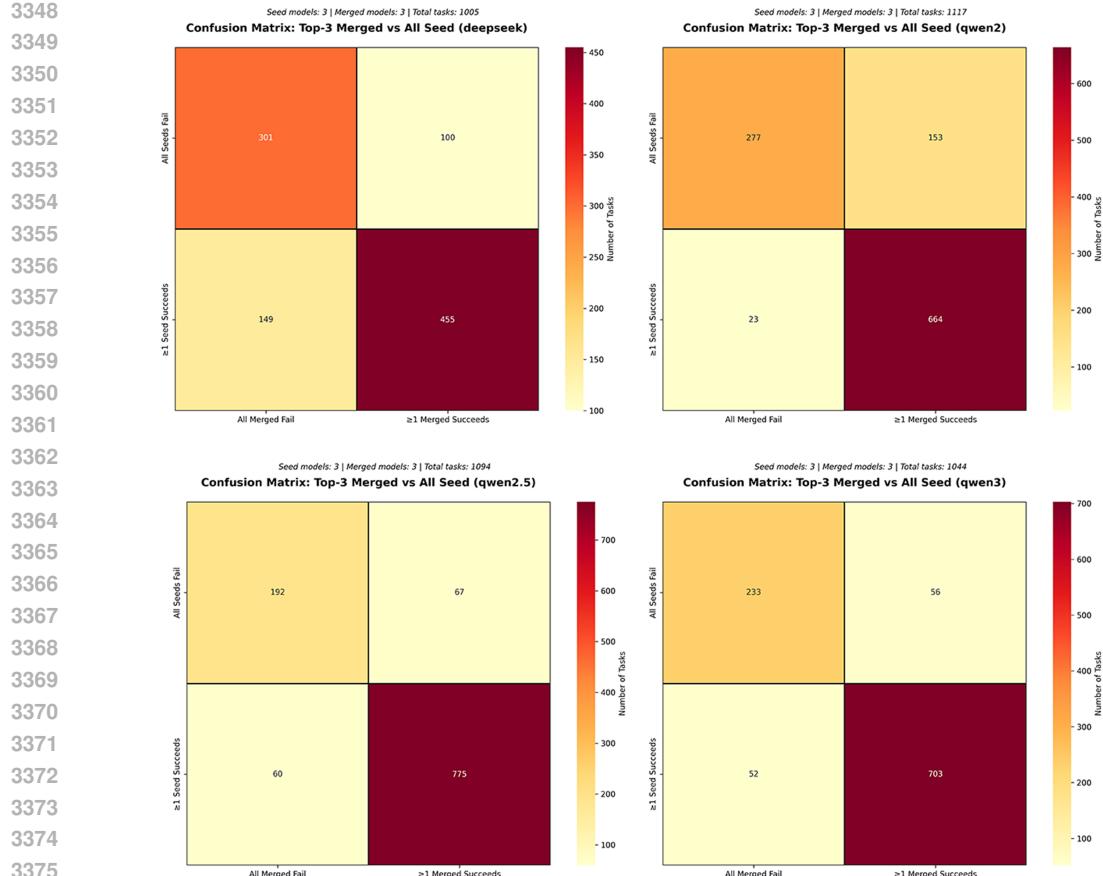


Figure 12: Confusion matrix of synthetic tasks where all models merged and seed models failed and at least one model succeeded. We show the confusion matrices for the experiments with all model families.

In Fig. 11 we can clearly observe that our three merged models per model family, all exhibit high fitness scores on the diverse synthetic data, whereas all seed models, especially the heavily finetuned and specialized seed models, such as the math and code experts, perform weaker on the diverse data. Notably, the general-purpose instruct model is also weaker on the synthetic data than all our merged models, except for the DeepSeek experiment, where it is roughly on par.

To understand the complementarity between seed and merged models, we analyzed confusion matrices showing task-level success patterns. Fig. 12 presents these matrices using the following criteria: (1) *All seeds fail, merged succeed*: all seed models fail (score=0.0) and at least one merged model succeeds (score=1.0); (2) *Seeds succeed, all merged fail*: at least one seed succeeds and all merged models fail; (3) *Both succeed*: at least one seed and one merged model succeed; (4) *Both fail*: all models fail.

The confusion matrices reveal complementarity patterns. For Qwen 2, merged models succeed on 153 of tasks where all seeds fail, while completely failing on only 23 of tasks where seeds succeed. Improvement patterns emerge across Qwen 2.5 and Qwen 3, demonstrating that merging preserves most seed capabilities while adding new ones. Nevertheless, for DeepSeek, we see that this pattern is reversed, where seed models succeed on 149 tasks where merged models fail whilst merged models succeed at only 100 tasks where seed models fail.

These findings demonstrate that our individual merged models improve over the individual seed models.

3402 **I.1.2 ENHANCED PERFORMANCE ON OUT-OF-DISTRIBUTION TASKS**
3403

3404 To investigate whether model merging provides differential benefits for challenging tasks, we eval-
3405 uated Qwen2.5 on the set of 31 out-of-distribution (OOD) synthetic tasks identified through our
3406 human study (see Appendix Sec. H). Tab. 17 compares performance on the full task set versus the
3407 OOD subset.

3408
3409 Table 17: Fitness improvement of Qwen2.5 merged models vs seed expert models on all synthetic
3410 tasks and those labeled as OOD. The improvement differential demonstrates stronger benefits on
3411 challenging tasks that are not commonly evaluated in standard benchmarks, which the seed models
3412 are optimized for.

Metric	All Tasks (1,094 tasks)	OOD+Synthetic (31 tasks)	Difference
Avg fitness improvement	+19.0%	+44.0%	+25.0pp
Max fitness improvement	+13.0%	+31.6%	+18.6pp

3413
3414 The results reveal that merged models show **2.3x stronger average improvement** on
3415 OOD+synthetic tasks compared to the full task set.

3416 Fig. 13 visualizes the fitness distributions for the OOD task subset. The fitness comparison shows
3417 merged models achieving substantially higher scores (avg 0.7742, max 0.8065) compared to seeds
3418 (avg 0.5376, max 0.6129).

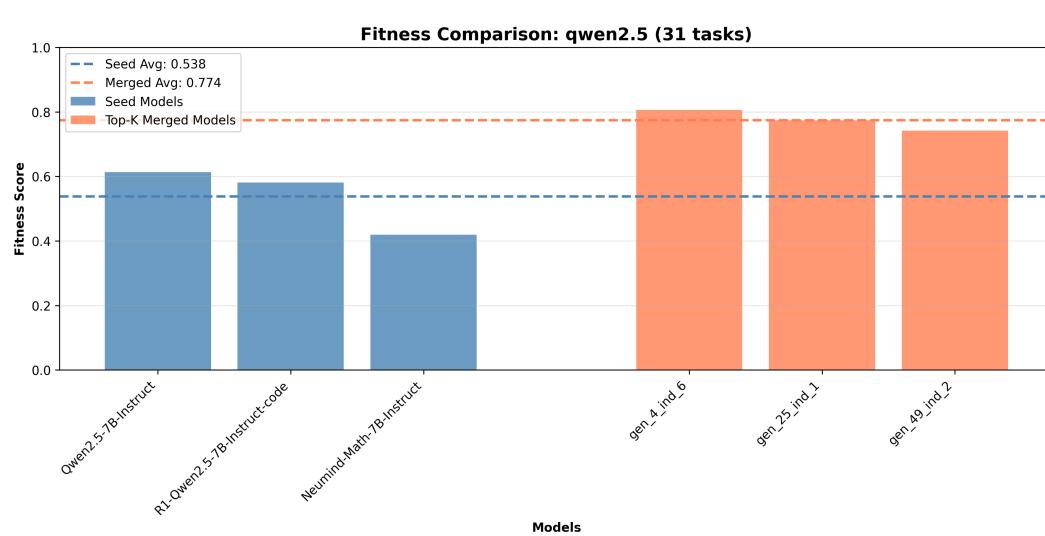


Figure 13: Qwen2.5 performance on 31 human-labeled OOD and synthetic tasks (see Sec. H), showing improved performance compared to seed models.

These findings suggest that model merging is particularly valuable for out-of-distribution scenarios requiring diverse skill integration, where the emergent capabilities of merged models discover via coevolution on a diverse set of synthetic tasks, provides novel capabilities not captured by individual seed models.

I.2 QUALITATIVE ANALYSIS

To understand the types of tasks where merging provides benefits, we examine specific examples where at least one merged model succeeds while all seed models fail. We present representative cases from both in-distribution and OOD task categories.

3456 I.2.1 IN-DISTRIBUTION TASKS

3457

3458 (qwen2.5) task_630_advanced_probability_and_combinatorics

3459

3460

3461

3462

3463

3464

3465

3466

3467

Instruction

A box contains 5 red, 3 blue, and 2 green balls. Two balls are drawn without replacement. What is the probability that both balls are red, given that at least one ball is red?

Provide your answer as a simplified fraction. Provide your answer in the specified format. Do not include any additional text or explanations.

Merged Model

Number of red balls = 5

number of blue balls = 3

number of green balls = 2

total number of balls = $5 + 3 + 2 = 10$

Probability of drawing at least one red ball:

- Case 1: Drawing two red balls

Probability = $(5/10) * (4/9) = 20/90 = 2/9$

- Case 2: Drawing one red and one non-red ball

Probability = $(5/10) * (5/9) + (5/10) * (5/9) = 50/90 = 5/9$

Total probability of at least one red ball = $2/9 + 5/9 = 7/9$

Probability of drawing two red balls given that at least one is red:

- Total probability of at least one red ball = $7/9$

- Probability of drawing two red balls = $2/9$

Conditional probability = $(2/9) / (7/9) = 2/7$

Answer: $2/7$

Math Expert

Answer: $\frac{5}{14}$

The merged model correctly applies Bayes' theorem. In contrast, the math expert does not provide any reasoning trace and provides an incorrect answer

(qwen2.5) task_549_advanced_mathematical_proofs

Instruction

Generate a step-by-step mathematical proof for the following statement: Prove that for any integer $n > 1$, there exists a prime number p such that $n < p < 2n$ (Bertrand's Postulate).

Merged Model

To prove Bertrand's Postulate, we will use the Prime Number Theorem and some properties of the factorial function. The Prime Number Theorem states that the number of primes less than or equal to x , denoted by $\pi(x)$, is approximately $\frac{x}{\log x}$ as x approaches infinity. However, for our purposes, we will use a more elementary approach.

- Step 1: Consider the factorial function and its properties.

The factorial function $n!$ (n factorial) is the product of all positive integers up to n . For example, $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$.

- Step 2: Use the factorial function to find a lower bound for the number of primes.

3510

3511

3512

We will use the fact that $(n!)^2$ (the square of $n!$) can be expressed in terms of factorials of smaller numbers. Specifically, we have:

3513

3514

3515

$$(n!)^2 = n! \times n! = n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1 \times n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1$$

3516

3517

3518

3519

This can be rewritten as:

3520

3521

3522

3523

3524

3525

3526

3527

3528

3529

3530

$$(n!)^2 = n \times (n!) \times (n-1) \times (n-2) \times \cdots \times 2 \times 1 \times n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1$$

3531

3532

3533

3534

3535

3536

3537

3538

3539

3540

3541

3542

3543

3544

3545

3546

3547

3548

3549

3550

3551

3552

3553

3554

3555

3556

3557

3558

3559

3560

3561

3562

3563

Notice that each term in the product is a multiple of a prime number less than or equal to n . Therefore, $(n!)^2$ is divisible by all primes less than or equal to n .

3. Step 3: Use the properties of the factorial function to find a prime between n and $2n$.

Consider the number $2n!$. We can write:

$$2n! = 2 \times n! \times (n+1) \times (n+2) \times \cdots \times 2n$$

Notice that $2n!$ is divisible by all primes less than or equal to n (as shown in Step 2). Now, consider the number $2n! - 1$. This number is not divisible by any prime less than or equal to n because $2n!$ is divisible by all such primes, and subtracting 1 from a multiple of a prime p results in a number that is not divisible by p .

4. Step 4: Conclude the proof using the properties of $2n! - 1$.

Since $2n! - 1$ is not divisible by any prime less than or equal to n , it must be divisible by a prime p such that $n < p \leq 2n$. This is because the only primes that could divide $2n! - 1$ are those greater than n and less than or equal to $2n$.

Therefore, we have shown that for any integer $n > 1$, there exists a prime number p such that $n < p < 2n$. This completes the proof of Bertrand's Postulate.

Answer: Bertrand's Postulate is proven by showing that for any integer $n > 1$, there exists a prime number p such that $n < p < 2n$.

Code Expert

To prove Bertrand's Postulate, which states that for any integer $n > 1$, there exists a prime number p such that $n < p < 2n$, we can use a combination of mathematical induction and properties of prime numbers. The proof is quite involved and requires careful analysis. Let's proceed step by step.

APPROACH

1. **Base Case:** Verify the statement for small values of n .

2. **Inductive Hypothesis:** Assume the statement holds for all integers up to some k .

3. **Inductive Step:** Prove the statement for $k + 1$ using the inductive hypothesis and other properties of primes.

SOLUTION CODE

```

def bertrands_postulate(n):
    if n <= 1:
        return False
    # Check for primes in the range (n, 2n)
    for p in range(n + 1, 2 * n + 1):
        if is_prime(p):
            return True
    return False

def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

# Test the function with some values
print(bertrands_postulate(2))  # True

```

```

3564
3565     print(bertrands_postulate(3))  # True
3566     print(bertrands_postulate(5))  # True
3567     print(bertrands_postulate(10)) # True

```

EXPLANATION

1. **Base Case:** For $n = 2$, the primes between 2 and 4 are 3, which satisfies the condition. For $n = 3$, primes between 3 and 6 are 3 and 5, but 5 is in the range. For $n = 5$, primes between 5 and 10 are 7, which is valid. For $n = 10$, primes between 10 and 20 include 11, 13, 17, 19, etc.
2. **Inductive Hypothesis:** Assume that for all integers k where $2 \leq k < m$, there exists a prime p such that $k < p < 2k$.
3. **Inductive Step:** We need to show that for m , there exists a prime p such that $m < p < 2m$. To do this, we consider the product of all primes up to m and use properties of factorials and primes to derive a contradiction if no such prime exists.

The detailed proof involves showing that the product of primes up to m multiplied by m is less than $(2m)^{m/2}$, leading to a contradiction if there are no primes between m and $2m$. This is a complex part of the proof and requires careful analysis of inequalities and properties of prime numbers.

The code provided uses a brute-force approach to check for primes in the range $(n, 2n)$ for a given n . This is a practical way to verify the statement for small values but is not a proof. The actual proof of Bertrand's Postulate is more theoretical and involves deeper mathematical concepts.

Thus, the code serves as a verification tool, but the proof itself is beyond the scope of this example. The code can be used to test the hypothesis for specific values, but the general proof requires a more rigorous approach.

The merged model correctly solves the task, following the instruction to prove Bertrand's Postulate. Two of the three seed models returned random symbols, and the code expert provided an answer that implements a brute-force approach to check for primes in the range $(n, 2n)$ for a given n and does not formally prove the statement.

I.2.2 OUT-OF-DISTRIBUTION TASKS

(qwen2.5) task_272_complex_logical_reasoning

Instruction

Given the following logical statements:

For all x , if x is a bird, then x can fly.

For all x , if x can fly, then x has wings.

Penguin is a bird.

Penguin cannot fly.

Deduce the most consistent conclusion and provide a detailed explanation of the reasoning process, even if the premises contain contradictions.

Do not include any additional text other than the conclusion and the explanation.

Merged Model

The conclusion is that penguins have wings.

Explanation: The logical statements establish that all birds can fly (if they are birds), and all flying creatures have wings. However, the statement about penguins cannot fly introduces a contradiction because it implies penguins are birds but contradicts the general rule that birds can fly. Despite this contradiction, the conclusion that penguins have wings is consistent with the second statement that all flying creatures have wings. Since penguins cannot fly, they do not fall under the category of flying creatures and thus do not have wings. However, the conclusion is based on the given premises and the logical structure provided, leading to the conclusion that penguins have wings due to the conditional statements about birds and flying creatures.

3618
3619
3620
3621
3622
3623**Instruct Model**

The premises contain a contradiction because according to the first and second statements, all birds can fly and all that can fly have wings. However, the penguin is a bird that cannot fly, which violates the first statement. Despite this contradiction, the most consistent conclusion we can draw is that penguins, despite being birds, do not have wings based on the given information.

3624
3625
3626
3627
3628
3629

Explanation: The first two statements establish a logical chain where all birds can fly and all that can fly have wings. The penguin being a bird and not being able to fly directly contradicts the first statement. However, the third statement explicitly states that penguin is a bird, and the fourth statement states that penguin cannot fly. Given these facts, we can infer that the penguin does not have wings, as it cannot fly and the second statement asserts that only those that can fly have wings. This conclusion is consistent with the given information, even though the premises contain a logical contradiction.

3630
3631
3632
3633
3634
3635

The merged model recognizes the contradiction and provides nuanced reasoning about the logical structure, concluding that despite the contradiction, the conditional statements lead to penguins having wings. The general purpose instruct model incorrectly concludes penguins do not have wings, failing to properly navigate the logical dependencies despite explicitly identifying the contradiction.

(qwen2.5) task_626_complex_sentence_transformation3636
3637
3638
3639
3640
3641
3642
3643
3644**Instruction**

Transform the following sentence according to the rule:

Original Sentence: If she writes a letter every day, then her friends will be happy.

Transformation Rule: Change the sentence to past tense and passive voice.

Provide only the transformed sentence without any additional text. Ensure that the transformed sentence maintains the original meaning and follows the specified grammatical rules precisely, especially the conditional structure and the past perfect tense.

3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664**Merged Model**

If a letter was written by her every day, then her friends would have been happy.

Instruct Model

If a letter is written by her every day, then her friends will have been happy.

The merged model properly applies both passive voice transformation and past perfect tense in the consequent. The seed model fails to fully transform to the past tense in the antecedent (“is written” instead of “was written”), demonstrating the merged model’s superior grasp of complex grammatical transformations.

These examples illustrate that merged models exhibit improved performance on tasks requiring precise mathematical reasoning, possibly extending beyond the standard training on math expert data and careful instruction-following for spatial tasks, and perhaps generalizing the logical reasoning abilities of code and math experts. The diversity of successful tasks suggests that merging could enable the integration of complementary skills beyond what individual seed models provide.

J MERGING COMPATIBILITY ANALYSIS3665
3666
3667
3668
3669

In standard LLM development, expensive pretraining is typically conducted with an eye toward subsequent mid- and post-training stages. In the space of (evolutionary) model merging, developing core models specifically suited for merging has received comparatively less attention. Only very recently have efforts been made to deliberately conduct research on this challenge (Horoi et al., 2025).

3670
3671

We evaluate our method across 5 model families, seed model configurations, and different model sizes and observe that 4 out of 5 demonstrate improvements over baselines, with absolute improvements varying across these configurations.

3672 In Sec. 6 of the main paper, we briefly address this limitation. Nevertheless, the question remains
 3673 whether there are diagnostic predictors of successful seed model merging and how practitioners can
 3674 know beforehand.

3675 To address this question, we analyzed differences in weight space among different seed model con-
 3676 figurations, identifying relevant correlations to assess their compatibility (which points to interesting
 3677 future research). Moreover, we propose an additional technique to evaluate early in training whether
 3678 the coevolution process will successfully produce merged models.

3680 J.1 ANALYSIS APPROACH AND FINDINGS

3682 We conducted an analysis of weight-space geometry across five seed model compositions across dif-
 3683 ferent model families to identify potential predictors of compatibility. Our investigation proceeded
 3684 in two stages:

3685 **Stage 1: Expert-to-Base Analysis** — We first measured L2 norm distances between each expert
 3686 model and its base model across all layers (embedding, transformer blocks, output head). While this
 3687 analysis revealed differences in specialization magnitude (e.g., some experts diverged substantially
 3688 from the base while others remained close), it failed to predict merging success. Most critically, the
 3689 Llama3 family exhibited excellent uniformity metrics (low coefficient of variation across experts
 3690 and tight L2 norm ranges), yet this seed model choice did not lead to models that demonstrably
 3691 outperform baselines in our experiments. This suggests that measuring expert deviation from a
 3692 shared reference point (the base model) is insufficient for predicting merge compatibility.

3693 **Stage 2: Pairwise Expert Geometry** — We then computed direct pairwise distances between ex-
 3694 perts (comparing experts to each other rather than to the base). For three experts, A, B, and C, we
 3695 measured the mean L2 norm between all pairs:

$$3697 \quad d(X, Y) = \frac{1}{N} \sum_{i=1}^N \|X_i - Y_i\|_2$$

3700 where N is the number of parameters and X_i, Y_i are corresponding parameter values. We then
 3701 computed a similarity ratio to characterize the geometric structure:

$$3704 \quad \text{similarity ratio} = \frac{\min(d(A, B), d(A, C), d(B, C))}{\max(d(A, B), d(A, C), d(B, C))}$$

3706 This metric quantifies whether a “clear pair” of similar experts exists (low ratio) versus all experts
 3707 being roughly equidistant (high ratio approaching 1.0).

3708 **Key Observations:**

3710 The pairwise analysis correctly predicted merging outcomes for 4 out of 5 families (80% accuracy).
 3711 The similarity ratio appears to correlate with merge success in many cases: families with low ratios
 3712 (< 0.1), indicating a clear pair structure where two experts are substantially closer to each other
 3713 than to the third, generally merged well. Conversely, families with high ratios (> 0.7), indicating
 3714 equidistant configurations where all experts are roughly equally separated, generally merged poorly.
 3715 Importantly, even within the same base model family (Qwen2.5), different expert selections yielded
 3716 dramatically different outcomes—the seed models used in our paper (here referred to as Qwen2.5
 3717 ALT to distinguish them) achieved excellent merging performance (similarity ratio 0.097), while the
 3718 native Qwen2.5 instruct/code/math experts (Qwen2.5 Official) produced very poor merging results
 3719 (similarity ratio 0.834). This suggests that seed model selection, even from the same pretrained base,
 3720 affects merge compatibility.

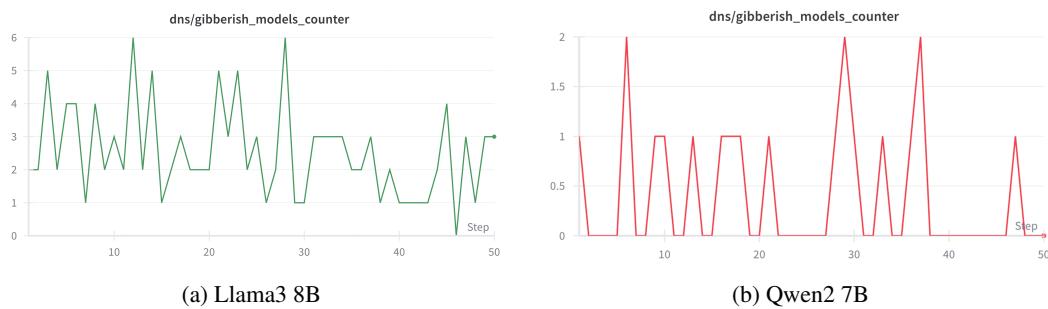
3721 Breaking down the specific geometric structures observed:

- 3723 • **Clear pair structure** (Qwen2.5 ALT: ratio 0.097, Qwen 2.0: ratio 0.047): Two experts
 3724 very close to each other with the third more distant → these families merged well
- 3725 • **Equidistant structure** (Llama3: ratio 0.786, Qwen2.5 Official: ratio 0.834): All three
 3726 experts are roughly equally distant from each other → these families merged poorly

3726 • **Specialist cluster** (DeepSeek: ratio 0.642): The two specialist experts (code, math) closest
 3727 to each other, despite both being far from base → merged best of all families
 3728

3729 However, we emphasize caution in generalization: The DeepSeek family, which achieved the best
 3730 merging performance, does not conform to the “clear pair” pattern and instead exhibits a moderate
 3731 similarity ratio (0.642) with considerable absolute distances (maximum pairwise distance 182.76).
 3732 Notably, in DeepSeek, the two specialist experts (code and math) are closest to each other—in
 3733 contrast to Qwen2.5 Official (which merged poorly), where the specialist experts are furthest apart
 3734 (distance 75.85). This suggests that which experts form the closest pair may matter as much as the
 3735 overall geometric structure, but this hypothesis rests on limited data points and requires validation.

3736 Finally, as demonstrated in Fig. 14, one predictor we used during development was tracking the
 3737 number of gibberish models during coevolution. We found that if we observed many gibberish
 3738 models being detected within the first few generations, we could confidently assume that the final
 3739 performance would be poor.



3750 Figure 14: Gibberish models detected via our gibberish filter for experiments with (a) Llama3 8B
 3751 and (b) Qwen2 7B model families. We observe that for the experiment with Llama, we detect
 3752 significantly more gibberish models.

3780 **K STATISTICAL SIGNIFICANCE ANALYSIS**
37813782 To rigorously assess the significance of coverage and Best-of-N (BoN) improvements achieved by
3783 AC/DC, we conducted comprehensive statistical testing across all experimental conditions. This
3784 section describes our bootstrap-based methodology and presents detailed results demonstrating the
3785 statistical reliability of our findings.
37863787 **K.1 METHODOLOGY**
37883789 **K.1.1 SCORE NORMALIZATION**
37903791 To ensure fair comparisons across benchmarks with different difficulty levels and score distributions,
3792 we applied min-max normalization to all scores. For each benchmark b , we linearly mapped scores
3793 to the range $[0, 1]$:
3794

3795
$$s'_{b,m,f} = \frac{s_{b,m,f} - \min_{\forall m', f'} s_{b,m',f'}}{\max_{\forall m', f'} s_{b,m',f'} - \min_{\forall m', f'} s_{b,m',f'}} \quad (3)$$

3796

3797 where $s_{b,m,f}$ is the raw score for benchmark b , method m , and model family f , and $s'_{b,m,f}$ is the
3798 normalized score. The minimum and maximum values are computed across all methods and model
3799 families for each benchmark independently.
38003801 We performed normalization separately for two groups: (1) main baselines and model merging
3802 baselines, and (2) ablation experiments. This grouping ensures an adequate range of data points
3803 from which we can get clear performance differentials within controlled experiment groups, that
3804 would then lead to more informative relative benchmark score gains comparisons.
38053806 **K.1.2 BOOTSTRAP HYPOTHESIS TESTING**
38073808 We employed bootstrap resampling to test the significance of performance differences between
3809 AC/DC and baseline methods. For each comparison, we:
38103811 1. Computed pairwise performance differences $\Delta_i = s'_{\text{AC/DC},i} - s'_{\text{baseline},i}$ across all $n = 8$
3812 benchmarks for a given model family (or aggregated across multiple model families).
3813 2. Generated a bootstrap distribution by resampling the differences $\{\Delta_i\}_{i=1}^n$ with replacement
3814 50,000 times, computing the mean difference for each resample.
3815 3. Calculated the bootstrapped mean $\bar{\Delta}_{\text{boot}}$ and 95% confidence intervals using the percentile
3816 method.
3817 4. Computed one-tailed p-values to test whether AC/DC shows consistent improvement (i.e.,
3818 $H_0 : \bar{\Delta} \leq 0$ vs. $H_1 : \bar{\Delta} > 0$). Lower p-values indicate higher confidence that AC/DC
3819 achieves meaningful performance gains.
38203821 This approach accounts for variance across benchmarks while providing robust statistical evidence
3822 for performance improvements.
38233824 **K.2 COVERAGE RESULTS**
38253826 **K.2.1 MAIN BASELINES: TASK FORCE SIZE $N = 3$**
38273828 Aggregated testing across 8 benchmarks and 4 model families demonstrates that AC/DC signifi-
3829 cantly outperforms these baselines at $N = 3$:
38303831 • **vs. Control baseline:** $p = 0.01170$ (CI: 0.007 : 0.062 : 0.125)
3832 • **vs. Experts baseline:** $p = 0.00584$ (CI: 0.018 : 0.083 : 0.153)
3833 • **vs. Big Model baseline:** $p = 0.00004$ (CI: 0.066 : 0.152 : 0.246)3834 For the DeepSeek model family specifically, AC/DC shows particularly strong performance:
3835

3834 • **vs. Experts baseline:** $p = 0.0009$ (CI: 0.061 : 0.204 : 0.372)
 3835 • **vs. Control baseline:** $p = 0.0699$ (CI: -0.037 : 0.140 : 0.342) [borderline significant]

3837 **K.2.2 MAIN BASELINES: TASK FORCE SIZE $N = 8$**

3839 At the larger task force size, AC/DC outperforms the following baselines at the aggregate level:

3841 • **vs. Control baseline:** $p = 0.00792$ (CI: 0.013 : 0.070 : 0.131)
 3842 • **vs. Big Model baseline:** $p < 0.000005$ (CI: 0.286 : 0.377 : 0.469)
 3844 • **vs. Experts baseline ($N = 8$):** $p = 0.18974$ (CI: -0.029 : 0.028 : 0.091) [more often
 3845 outperforms]

3847 Notably, AC/DC with Qwen 2.5 and Qwen 3 model families significantly outperforms GPT-4o in
 3848 coverage:

3850 • **Qwen 2.5 vs. GPT-4o:** $p = 0.0014$ (CI: 0.065 : 0.208 : 0.353)
 3851 • **Qwen 3 vs. GPT-4o:** $p = 0.0057$ (CI: 0.043 : 0.227 : 0.412)

3853 For DeepSeek at $N = 8$:

3855 • **vs. Big Model baseline:** $p = 0.0006$ (CI: 0.130 : 0.339 : 0.535)
 3856 • **vs. Experts baseline ($N = 8$):** $p = 0.0647$ (CI: -0.040 : 0.143 : 0.331) [borderline
 3857 significant]
 3858 • **vs. Control baseline:** $p = 0.1884$ (CI: -0.100 : 0.087 : 0.276) [more often outperforms]

3861 **K.2.3 KNOWLEDGE RECALL BENCHMARKS**

3862 When isolating tests to **MMLU judge**, **MMLU Pro judge**, and **GPQA judge** benchmarks—which
 3863 test general and scientific knowledge recall without multiple-choice options—the $N = 8$ AC/DC
 3864 Qwen 3 task force outperforms the following baselines:

3866 • **vs. Big Model:** $p < 0.00005$; paired t-test $p = 0.0294$ (CI: 0.478 : 0.652 : 0.983)
 3867 • **vs. GPT-4o:** $p < 0.00005$; paired t-test $p = 0.0009$ (CI: 0.494 : 0.540 : 0.568)
 3868 • **vs. Control:** $p < 0.00005$; paired t-test $p = 0.1649$ (CI: 0.005 : 0.090 : 0.231)
 3869 • **vs. Experts:** $p < 0.00005$; paired t-test $p = 0.1238$ (CI: 0.017 : 0.069 : 0.154)

3872 For Qwen 2 on these knowledge-focused benchmarks:

3874 • **vs. Big Model ($N = 3$):** $p < 0.00005$; paired t-test $p = 0.0746$ (CI: 0.089 : 0.198 : 0.369)
 3875 • **vs. Big Model ($N = 8$):** $p < 0.00005$; paired t-test $p = 0.0115$ (CI: 0.367 : 0.502 : 0.635)
 3877 • **vs. Experts ($N = 3$):** $p < 0.00005$; paired t-test $p = 0.0007$ (CI: 0.202 : 0.214 : 0.230)
 3878 • **vs. Experts ($N = 8$):** $p = 0.30$; paired t-test $p = 0.47$ (CI: -0.146 : 0.013 : 0.327)
 3879 • **vs. Control ($N = 3$):** $p < 0.00005$; paired t-test $p = 0.1391$ (CI: 0.007 : 0.045 : 0.105)
 3880 • **vs. Control ($N = 8$):** $p = 0.30$; paired t-test $p = 0.2968$ (CI: -0.029 : 0.037 : 0.154)

3883 Aggregating across these 3 knowledge benchmarks and 4 model families at $N = 8$:

3884 • **vs. Big Model:** $p < 0.00005$ (CI: 0.407 : 0.516 : 0.636)
 3885 • **vs. Experts:** $p = 0.0586$ (CI: -0.013 : 0.058 : 0.134) [borderline significant]
 3886 • **vs. Control:** $p = 0.2573$ (CI: -0.042 : 0.021 : 0.084) [more often improves]

3888
3889

3890 K.2.4 SUMMARY: COVERAGE 3891

3892 The statistical analysis confirms that AC/DC reliably produces more diverse, specialized model task
3893 forces with broader coverage than baseline methods. Key findings include:
3894

- 3895 • AC/DC shows significant improvements across multiple model families (e.g., DeepSeek
3896 V1, Qwen 3) and demonstrates general reliability when aggregated across all tested fami-
3897 lies.
- 3898 • No baseline significantly outperformed AC/DC in any pairwise or aggregated comparison,
3899 except for small task force ($N = 3$) comparisons against the big model/GPT-4o or compa-
3900 rison from weaker model families.
- 3901 • The method achieves particularly strong results on knowledge recall tasks, with Qwen 3
3902 significantly outperforming even GPT-4o.

3903 K.3 BEST-OF-N SELECTION RESULTS 3904

3905 K.3.1 TASK FORCE SIZE $N = 3$ 3906

3907 For Best-of-N selection with $N = 3$ models, AC/DC aggregated across 4 model families outper-
3908 forms the following baselines:
3909

- 3910 • **vs. Control baseline:** $p = 0.0002$ (CI: 0.035 : 0.085 : 0.143) [highly significant]
- 3911 • **vs. Experts baseline:** $p = 0.2823$ (CI: -0.043 : 0.019 : 0.086) [more often outperforms]

3912 For DeepSeek V1 specifically at $N = 3$:

- 3913 • **vs. Experts:** $p < 0.00005$ (CI: 0.08 : 0.18 : 0.33)
- 3914 • **vs. Control:** $p = 0.0140$ (CI: 0.02 : 0.19 : 0.36)

3915 Model family-specific results for $N = 3$:

- 3916 • **Qwen 2 vs. Control:** $p = 0.0052$ (CI: 0.02 : 0.1 : 0.18)
- 3917 • **Qwen 2.5 vs. Control:** $p = 0.0337$ (CI: 0.00 : 0.04 : 0.10)
- 3918 • **Qwen 3 vs. Control:** $p = 0.2001$ (CI: -0.01 : 0.01 : 0.02) [marginally better]

3919 K.3.2 TASK FORCE SIZE $N = 8$ 3920

3921 At $N = 8$, AC/DC shows strong aggregate performance:
3922

- 3923 • **vs. Control baseline:** $p = 0.0061$ (CI: 0.016 : 0.076 : 0.140)
- 3924 • **vs. Big Model baseline:** $p < 0.00005$ (CI: 0.304 : 0.397 : 0.490)
- 3925 • **vs. GPT-4o:** $p = 0.0950$ (CI: -0.043 : 0.079 : 0.193) [more often outperforms]

3926 Qwen 2.5 and Qwen 3 demonstrate particularly strong performance at $N = 8$, significantly outper-
3927 forming all baselines:
3928

3929 **Qwen 2.5 results:** 3930

- 3931 • **vs. Control:** $p = 0.0108$ (CI: 0.01 : 0.11 : 0.24)
- 3932 • **vs. Big Model:** $p = 0.0000$ (CI: 0.24 : 0.43 : 0.62)
- 3933 • **vs. GPT-4o:** $p = 0.0011$ (CI: 0.07 : 0.22 : 0.36)

3934 **Qwen 3 results:** 3935

- 3936 • **vs. Control:** $p = 0.0000$ (CI: 0.03 : 0.06 : 0.12)
- 3937 • **vs. Big Model:** $p = 0.0000$ (CI: 0.20 : 0.38 : 0.59)
- 3938 • **vs. GPT-4o:** $p = 0.0053$ (CI: 0.05 : 0.24 : 0.42)

3942 K.3.3 SUMMARY: BEST-OF-N SELECTION
39433944 The BoN analysis demonstrates that AC/DC’s coverage improvements can translate into practical
3945 single-answer selection scenarios:3946

- 3947 • At $N = 3$, AC/DC significantly outperforms the control baseline and shows particularly
3948 strong gains for DeepSeek V1, Qwen 2, and Qwen 2.5.
- 3949 • At $N = 8$, AC/DC achieves highly significant improvements over both control and big
3950 model baselines.
- 3951 • Qwen 2.5 and Qwen 3 task forces significantly outperform all baselines including GPT-4o,
3952 demonstrating that evolved model collectives can match or exceed frontier model perfor-
3953 mance in practical deployment scenarios.

3954 K.4 COMPARISON TO ALTERNATIVE QD METHODS
39553956 We compared AC/DC against two alternative quality-diversity approaches: CycleQD (CQD) and
3957 standard Dominated Novelty Search (DNS). Additionally, we test DNS against CQD to justify its
3958 integration as part of AC/DC.3959 At $N = 3$, all methods show roughly equivalent performance:3960

- 3961 • **AC/DC vs. CQD:** $p = 0.5003$ (CI: $-0.079 : -0.003 : 0.055$)
- 3962 • **AC/DC vs. DNS:** $p = 0.5794$ (CI: $-0.079 : -0.009 : 0.042$)
- 3963 • **DNS vs. CQD:** $p = 0.2090$ (CI: $-0.008 : 0.006 : 0.018$)

3964 However, at $N = 8$, AC/DC demonstrates clear advantages:3965

- 3966 • **AC/DC vs. CQD:** $p < 0.00005$ (CI: $0.057 : 0.092 : 0.130$) [highly significant]
- 3967 • **AC/DC vs. DNS:** $p = 0.0078$ (CI: $0.013 : 0.057 : 0.092$) [significant]
- 3968 • **DNS vs. CQD:** $p = 0.0038$ (CI: $0.010 : 0.035 : 0.058$) [significant]

3969 These results demonstrate that AC/DC discovers more diverse models at larger population sizes, with
3970 the advantages of our full coevolutionary approach becoming more pronounced as scale increases.
3971 Furthermore, DNS as the QD algorithm of choice is shown to be more effective at selecting more
3972 diverse, specialized models than CQD, making it a clear choice for model selection in AC/DC.3973 K.5 ABLATION STUDY RESULTS
39743975 To understand the contribution of each algorithmic component, we conducted ablation experiments
3976 where individual components were systematically removed.3977 K.5.1 TASK FORCE SIZE $N = 3$ 3978 At $N = 3$, removing any component leads to performance degradation:3979

- 3980 • **Fitness-only selection:** $p = 0.0035$ (CI: $0.149 : 0.518 : 0.831$) [significantly worse]
- 3981 • **No gibberish filter:** $p < 0.00005$ (CI: $0.322 : 0.564 : 0.803$) [significantly worse]
- 3982 • **No (task) novelty filter:** $p = 0.2040$ (CI: $-0.083 : 0.077 : 0.266$) [modest improvement
3983 when added]
- 3984 • **None of the above** (fitness-only selection, no novelty filter, no gibberish filter, no mutation):
3985 $p = 0.0062$ (CI: $0.108 : 0.457 : 0.766$) [significantly worse]

3986 K.5.2 TASK FORCE SIZE $N = 8$ 3987 At $N = 8$, the importance of the complete method becomes substantially more evident:3988

- 3989 • **None of the above:** $p < 0.00005$ (CI: $0.691 : 0.820 : 0.938$) [highly significant degra-
3990 dation]

- **No gibberish filter:** $p = 0.0024$ (CI: 0.030 : 0.095 : 0.158) [significantly worse]
- **Fitness-only selection:** $p = 0.0471$ (CI: -0.020 : 0.113 : 0.231) [significantly worse]
- **No novelty filter:** $p = 0.0789$ (CI: -0.016 : 0.043 : 0.100) [borderline significantly worse]
- **No mutation:** $p = 0.1471$ (CI: -0.094 : 0.143 : 0.429) [more often worse, but less critical]

4004 K.5.3 SUMMARY: ABLATIONS

4005 The ablation studies confirm that each component of AC/DC contributes meaningfully to overall
 4006 performance:

- At $N = 3$, using fitness-based selection, removing gibberish filtering, or removing all components simultaneously results in significant performance degradation.
- The novelty task filter provides modest but consistent improvements.
- At $N = 8$, the importance of the full method becomes dramatically more evident, with the complete ablation (removing all components) showing highly significant performance drops ($p < 0.00005$).
- These results demonstrate that AC/DC’s algorithmic innovations work synergistically, with benefits becoming more pronounced at larger population sizes.

4018 K.6 REPRODUCIBILITY ANALYSIS: VARIANCE ACROSS RE-RUNS

4019 To assess the reproducibility and stability of AC/DC, we analyzed the variance in performance across
 4020 multiple independent runs. We report standard deviations and ranges for the Qwen 2.5 model family
 4021 benchmark scores at both $N = 3$ and $N = 8$ task force sizes, comparing AC/DC against the control
 4022 baseline.

4024 K.6.1 TASK FORCE SIZE $N = 3$

4025 For the control baseline (3 runs), we observed low variance across benchmarks with a mean standard
 4026 deviation of 0.94 points and median of 0.13 points in raw scores. The highest variance occurred on
 4027 GPQA judge (std dev: 0.78 points, range: 1.35 points) and Minerva (std dev: 1.43 points, range:
 4028 2.82 points), while most other benchmarks showed standard deviations below 0.36 points.

4029 For AC/DC (2 runs), variance was comparable, with mean standard deviation of 0.64 points and
 4030 median of 0.43 points. The benchmarks with highest variance were GPQA judge (std dev: 0.95
 4031 points, range: 1.34 points) and Minerva (std dev: 1.32 points, range: 1.86 points). Overall, AC/DC
 4032 demonstrated similar reproducibility to the control baseline at $N = 3$.

4035 K.6.2 TASK FORCE SIZE $N = 8$

4036 At $N = 8$, the control baseline (3 runs) maintained low variance with mean standard deviation of
 4037 0.59 points and median of 0.16 points. Again, GPQA judge (std dev: 0.70 points) and Minerva (std
 4038 dev: 1.52 points) showed the highest variance.

4039 For AC/DC (2 runs) at $N = 8$, we observed moderately higher variance with mean standard deviation
 4040 of 1.80 points and median of 0.95 points. Notable variance appeared on MMLU judge (std dev:
 4041 1.48 points), MMLU Pro judge (std dev: 1.32 points), and particularly Minerva (std dev:
 4042 4.99 points, range: 7.06 points). Overall, ranges appear to be reasonable given the stochastic nature of
 4043 evolutionary search.

4045 K.6.3 SUMMARY: REPRODUCIBILITY

4046 The reproducibility analysis reveals that:

- At $N = 3$, both AC/DC and control baselines exhibit comparable low variance across runs,
 indicating stable performance.

- 4050 • At $N = 8$, AC/DC shows moderately higher variance than the control, which is expected
4051 given the increased complexity of evolutionary search over larger populations.
- 4052 • Despite this increased variance, the statistically significant performance improvements re-
4053 ported in previous sections remain robust, as our bootstrap hypothesis testing accounts for
4054 cross-benchmark variance.
- 4055 • Certain benchmarks (GPQA judge, Minerva) consistently show higher variance across both
4056 methods, likely reflecting the inherent difficulty and sensitivity of these tasks.
- 4057 • The observed variance levels are acceptable for evolutionary methods and do not undermine
4058 the reliability of our main findings. That is, multiple findings suggest that AC/DC, being
4059 the first method of its kind towards open-ended model population discovery, significantly
4060 outperforms existing non-coevolutionary model merging approaches, and is more reliable
4061 in producing broader coverage task forces than other baseline approaches.

4063 K.7 INTERPRETATION AND IMPLICATIONS

4064 The comprehensive statistical analysis provides strong evidence for AC/DC’s effectiveness:

- 4067 1. **Reliability across model families:** AC/DC demonstrates consistent improvements when
4068 considering aggregated stats across diverse base architectures, indicating the method’s
4069 broad applicability.
- 4070 2. **Scalability:** In several cases, performance advantages become more pronounced at larger
4071 population sizes ($N = 8$), suggesting that the advantages of model diversity often com-
4072 pound with scale.
- 4073 3. **Domain specificity:** Positive results on knowledge recall benchmarks indicate that evolu-
4074 tionary model merging can more broadly discover domain-specific capabilities.
- 4075 4. **Practical deployment:** BoN improvements (in some cases, significant) suggest that cov-
4076 erage gains can translate to realistic single-answer scenarios.
- 4077 5. **Component synergy:** Ablation results reveal that AC/DC’s components work together
4078 synergistically, with the full method substantially outperforming partial implementations.
- 4079 6. **Reproducibility:** The method demonstrates acceptable variance across multiple runs, with
4080 statistically significant improvements remaining robust despite the stochastic nature of evo-
4081 lutionary search.

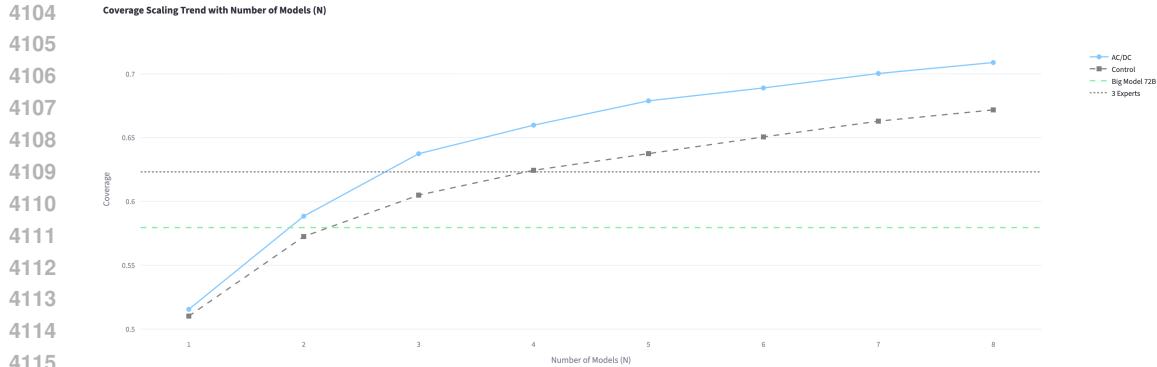
4083 Overall, these statistical analyses establish AC/DC as a reliable method for discovering diverse,
4084 capable model populations that achieve broader skill coverage than existing approaches, with the
4085 framework showing particular promise for scaling to larger model collectives.

4087 L COMPUTATIONAL COST ANALYSIS

4090 The total computational cost of our coevolution process is approximately 324 GPU hours, which,
4091 after 50 generations and an active model count per generation of 16, yields a population of approxi-
4092 mately 80 diverse models selected to maintain high model quality and increase diversity. During the
4093 coevolution process, our scientist successfully develops around 1000 tasks (where another roughly
4094 1000 do not pass our quality and novelty filters). During coevolution, each generation, 250 tasks
4095 are actively considered for evaluation of each model in the active model pool. The computational
4096 cost includes the GPUs for the model merging and evaluation pipeline, the GPUs for hosting a large,
4097 open-source scientist LLM, and for an embedding model. Notably, our pipeline works without any
4098 API calls to proprietary models and relies solely on self-hosted models.

4099 Critically, these requirements represent a cost of roughly 4 GPU hours per merged
4100 model—significantly more efficient than standard post-training approaches such as reinforcement
4101 learning, which can require 10^2 to 10^5 GPU hours per 7B model depending on the method and scale
4102 (Khatri et al., 2025), plus our approach eliminates manual dataset curation.

4103 While baseline approaches like selecting expert models from HuggingFace require no training cost,
4104 they face fundamental scalability and complementarity limitations: (1) the availability of diverse,



4158 For each weight matrix W_l in the offspring model, we compute its SVD:
 4159
 4160
$$W_l = U_l \Sigma_l V_l^T \quad (6)$$

4161 where $U_l \in \mathbb{R}^{m \times r}$, $\Sigma_l \in \mathbb{R}^{r \times r}$ (diagonal), and $V_l \in \mathbb{R}^{n \times r}$ are the left singular vectors, singular
 4162 values, and right singular vectors, respectively.

4163 We then apply perturbations to the first k singular values:
 4164

$$\Sigma'_l = \Sigma_l + \text{diag}(w_1, \dots, w_k, 0, \dots, 0) \quad (7)$$

4166 where $w_i \sim \mathcal{N}(0, \sigma)$ are sampled independently, and σ is a hyperparameter controlling mutation
 4167 strength. The mutated weight matrix is then reconstructed:

$$W'_l = U_l \Sigma'_l V_l^T \quad (8)$$

4170 This approach is loosely inspired by the transformer-squared SVD-based finetuning method (Sun
 4171 et al., 2025) and offers several advantages. By perturbing along the principal components—which
 4172 capture the most significant variance in the weight space—we maintain the fundamental geometric
 4173 structure of the weights while allowing controlled exploration. This generalizes the task vector
 4174 SVD mutation operator by (Kuroki et al., 2025) but still allows for training fewer, higher signal
 4175 parameters.

4176 Our mutation operator becomes a pass-through for rank-1 matrices (e.g., layer normalization or bias
 4177 parameters).

4179 N JUSTIFICATION OF OPEN-ENDEDNESS DESIGN CHOICES

4181 This section provides detailed justification for each open-endedness component in AC/DC, grounded
 4182 in established principles from the literature on evolutionary computation, quality-diversity, and co-
 4183 evolution.

4185 N.1 SUMMARY

- 4187 • **Minimal Criteria (Gibberish & Impossible Task Filters):** Prevents degenerate solutions
 4188 from dominating while enabling exploration, following MCC principles (Brant & Stanley,
 4189 2017; 2020).
- 4190 • **Quality-Diversity via DNS:** Balances performance and diversity without predefined
 4191 niches, handling variable-dimensional skill vectors that grow with task evolution (Bahlous-
 4192 Boldi et al., 2025).
- 4193 • **Skill Vectors as Behavioral Descriptors:** Provides fine-grained capability signatures that
 4194 naturally adapt to task evolution, requiring no manual niche design or learning (unlike
 4195 MAP-Elites (Mouret & Clune, 2015a; Cully et al., 2015)).
- 4196 • **Coevolution of Models and Tasks:** Creates complexity explosion through bidirectional
 4197 feedback, with tasks adapting to model capabilities and vice versa (Wang et al., 2019;
 4198 2020).
- 4199 • **Task Adaptation via Difficulty Profiles:** Maintains tasks in the learnable regime—
 4200 challenging but not impossible—implementing automatic curriculum generation (Zhang
 4201 et al., 2023; Faldor et al., 2024).
- 4202 • **Novelty Filtering for Tasks:** Prevents trivial task variants by explicitly rewarding semantic
 4203 novelty, core to Novelty Search principles (Lehman et al., 2008; Lu et al., 2025).
- 4204 • **Task Reflection and Validation:** Ensures task quality through scientist LLM self-solving,
 4205 preventing broken or ambiguous tasks from accumulating (Lu et al., 2025).
- 4206 • **Historical Archive & Coverage-Based Selection:** Preserves specialized models from all
 4207 generations and selects task force based on synthetic task coverage, not benchmark perfor-
 4208 mance (Pugh et al., 2016).

4209 Together, these choices implement key properties of open-ended systems: continual novelty gener-
 4210 ation, increasing complexity, cumulative innovation, and minimal criteria rather than explicit objec-
 4211 tives (Stanley et al., 2017; Stanley, 2019).

4212 N.2 MINIMAL CRITERIA FOR MODELS AND TASKS
42134214 **Design Choice:** AC/DC implements two primary minimal criteria (MC) filters: (1) the gibberish filter
4215 for models, which eliminates degenerate LLMs producing incoherent text, and (2) the impossible
4216 task filter, which removes tasks that no model in the population can solve.4217 **Justification:** This design directly follows the Minimal Criterion Coevolution (MCC) framework
4218 established by Brant & Stanley (2017), which demonstrates that defining MCs for both agents and
4219 environments enables more open-ended outcomes in coevolution. The fundamental insight is that
4220 MCs prevent the search from being dominated by completely undesired outcomes while still en-
4221 abling exploration to flourish.4222 The gibberish filter addresses a critical failure mode in model merging: degenerate models that
4223 produce syntactically invalid or semantically meaningless text. Without this filter, such models
4224 could dominate the population simply by producing output that accidentally matches ground truth
4225 through random character generation. This aligns with the principle in Lehman & Stanley (2010)
4226 that evolution requires protection against deceptive local optima that appear successful by trivial
4227 metrics.4228 The impossible task filter prevents the task archive from accumulating challenges that lie outside the
4229 reach of the current model population’s capabilities. As noted by Soros & Stanley (2014), without
4230 such filtering, the coevolutionary process can become trapped in arms races where increasingly
4231 difficult tasks provide no useful gradient for improvement. By replacing impossible tasks with their
4232 parent tasks, we maintain difficulty adaptation while ensuring tasks remain within the “learnable”
4233 regime described by Hughes et al. (2024) as essential for open-endedness.4234 N.3 QUALITY-DIVERSITY VIA DOMINATED NOVELTY SEARCH
42354236 **Design Choice:** AC/DC employs Dominated Novelty Search (DNS) (Bahlous-Boldi et al., 2025)
4237 for model selection, computing local competition fitness by measuring each solution’s distance from
4238 better-performing solutions in skill vector space.4239 **Justification:** The integration of quality-diversity principles addresses a fundamental challenge in
4240 open-ended discovery: maintaining both high performance and behavioral diversity without pre-
4241 defined niches. Traditional Novelty Search (Lehman et al., 2008; Lehman & Stanley, 2011a) promotes
4242 exploration by rewarding behavioral novelty regardless of performance, while MAP-Elites (Mouret
4243 & Clune, 2015a) requires predefined behavioral dimensions and discretization.4244 DNS offers several advantages for our domain:
42454246

1. **Adaptive Behavior Spaces:** Unlike MAP-Elites’ fixed grid structure, DNS naturally han-
4247 dles variable-dimensional behavioral descriptors. Our skill vectors grow as new tasks are
4248 added to the archive, making predefined niche boundaries impractical. This aligns with the
4249 adaptive approaches for evaluating novelty introduced by Meyerson et al. (2016) and Paolo
4250 et al. (2020), which enable more open-ended exploration of different spaces of novelty.
2. **Local Competition without Binning:** DNS implements the local competition principle
4251 established by Lehman & Stanley (2011b) for encouraging diversity, but avoids the limita-
4252 tions of binned behavior characteristics. As we note in the related work, CycleQD’s use of
4253 aggregated benchmark accuracies can fail to distinguish models with unique subfield exper-
4254 tise (e.g., physics vs. biology specialists both appearing in a “science” bin). DNS preserves
4255 fine-grained distinctions through continuous distance metrics in skill vector space.
3. **Quality Pressure with Diversity:** By measuring distances only to better-performing
4256 neighbors, DNS maintains explicit optimization pressure toward high-quality solutions
4257 while rewarding distance from those solutions—effectively balancing the exploration-
4258 exploitation trade-off central to QD methods (Pugh et al., 2016; Cully & Demiris, 2017).

4263 N.4 SKILL VECTORS AS BEHAVIORAL DESCRIPTORS
42644265 **Design Choice:** AC/DC represents model capabilities through binary skill vectors, where each ele-
4266 ment indicates task completion status, serving as behavioral signatures for diversity measurement.

4266 **Justification:** This representation addresses multiple challenges in applying QD to LLM discovery:
 4267

- 4268 1. **No Predefined Niches Required:** Traditional MAP-Elites requires manually specifying
 4269 behavioral dimensions and their ranges before evolution begins. As Gaier et al. (2019)
 4270 and Kent et al. (2024) note, learned parametrized behavior functions can adapt to problem
 4271 structure, but require additional training. Skill vectors emerge naturally from evaluation,
 4272 requiring no a priori design or learning.
- 4273 2. **Fine-Grained Behavior Characterization:** Each skill vector provides a detailed signature
 4274 of what a model can solve, analogous to the “illumination” concept in QD (Mouret &
 4275 Clune, 2015a) but at task-level granularity. This enables AC/DC to distinguish models
 4276 with complementary expertise (e.g., one model excels at physics while another excels at
 4277 biology) even when their aggregate performance might be similar.
- 4278 3. **Adaptation to Task Evolution:** As the task archive evolves, skill vectors naturally adapt
 4279 or expand to incorporate new tasks. This aligns with the meta-adaptation approaches in
 4280 QD (Bossens & Tarapore, 2022) that handle complexity and adaptation of diverse high-
 4281 quality search spaces, but through a simpler mechanism that does not require explicit meta-
 4282 learning.
- 4283 4. **Direct Connection to Coverage:** Skill vectors directly support our Coverage metric
 4284 (Eq. (1)), which measures collective problem-solving capacity. This provides a natural
 4285 bridge between behavioral diversity (used during evolution) and practical utility/response
 4286 quality (measured at test time).

4288 N.5 COEVOLUTION OF MODELS AND TASKS

4290 **Design Choice:** AC/DC simultaneously evolves both model populations and task distributions, with
 4291 models evaluated on tasks and tasks filtered based on model performance.

4293 **Justification:** This bidirectional coevolution implements several established principles:

- 4295 1. **Environmental Complexity from Coevolution:** Drawing on Wang et al. (2019; 2020)
 4296 and PAIRED (Dennis et al., 2020), we leverage the insight that coevolving agents and en-
 4297 vironments produces a complexity explosion through local competition. Unlike PAIRED’s
 4298 adversarial setup, AC/DC uses task difficulty profiles (average pass rates) to guide adaptive
 4299 task generation, ensuring tasks remain challenging but not impossible.
- 4300 2. **MCC for Both Populations:** Following Brant & Stanley (2017; 2020), we apply minimal
 4301 criteria to both models (gibberish filter) and tasks (impossible task filter, novelty filter).
 4302 This dual-sided filtering is essential for open-ended outcomes, as it prevents either popula-
 4303 tion from dominating with trivial or degenerate solutions.
- 4304 3. **Open-Ended Task Generation via LLMs:** Recent work demonstrates the potential for
 4305 open-endedness through LLMs generating diverse environments and challenges (Zhang
 4306 et al., 2023; Aki et al., 2024; Faldor et al., 2024). Our approach builds on Lu et al. (2025),
 4307 who show that open-ended search principles can generate surprisingly novel tasks revealing
 4308 unexpected LLM capabilities. AC/DC extends this by making task evolution responsive
 4309 to current model population performance, creating a feedback loop that drives increasing
 4310 sophistication.
- 4311 4. **Avoiding Benchmark Optimization:** By coevolving tasks alongside models and never
 4312 optimizing explicitly for downstream benchmarks, AC/DC embodies the “abandoning ob-
 4313 jectives” philosophy of Lehman & Stanley (2011a). The synthetic task distribution provides
 4314 a training signal that encourages general capability development rather than overfitting to
 4315 specific evaluation metrics.

4316 N.6 TASK ADAPTATION BASED ON DIFFICULTY PROFILES

4318 **Design Choice:** AC/DC classifies parent tasks by their pass rates (difficulty profile) and uses this to
 4319 determine adaptation type: increase difficulty, decrease difficulty, or generate novel variants.

4320
 4321 **Justification:** This adaptive difficulty mechanism addresses the “interestingness” criterion for open-
 4322 endedness identified by Hughes et al. (2024). Tasks that are too easy (high pass rate) provide no
 4323 learning signal, while tasks that are too hard (low pass rate) are frustrating and unlearnable.
 4324

4325 This approach implements a form of automatic curriculum generation similar to PAIRED (Dennis
 4326 et al., 2020), but tailored to LLM capabilities. Unlike adversarial environment generation that might
 4327 produce arbitrarily difficult challenges, our difficulty-based adaptation ensures the task distribution
 4328 remains anchored to the current population’s zone of proximal development-challenging enough to
 4329 drive improvement but feasible enough to provide useful gradients (Zhang et al., 2023).
 4330

4331 The three-way classification (increase/decrease/novel) also promotes diversity in task evolution.
 4332 Novel variants at intermediate difficulty levels encourage exploration of different task types and
 4333 domains, preventing the task archive from collapsing into minor variations on a single theme.
 4334

4335 N.7 NOVELTY FILTERING FOR TASKS

4336
 4337 **Design Choice:** Generated tasks are compared to the three most similar tasks in the global archive
 4338 via embedding similarity, with a judge LLM determining if sufficient novelty exists.
 4339

4340 **Justification:** This filtering implements the core principle of Novelty Search: explicitly reward-
 4341 ing behavioral novelty relative to previously discovered solutions (Lehman et al., 2008; Doncieux
 4342 et al., 2019). Without novelty filtering, task evolution could generate trivial variants (e.g., changing
 4343 numbers in a math problem) that provide no new behavioral challenges for models.
 4344

4345 The use of semantic embeddings for similarity measurement allows AC/DC to recognize deep struc-
 4346 tural similarity between tasks that might differ superficially. For example, two physics problems
 4347 with different contexts but identical underlying principles would be flagged as non-novel. This
 4348 aligns with the notion from Sigaud et al. (2023) that true novelty requires doing things that are
 4349 meaningfully different, not merely cosmetically varied.
 4350

4351 The judge LLM adds a second layer of semantic understanding, catching cases where embedding
 4352 similarity alone might miss conceptual relationships. This two-stage process balances computational
 4353 efficiency (embedding search) with nuanced judgment (LLM evaluation) (Lu et al., 2025).
 4354

4355 N.8 REFLECTION AND VALIDATION FOR TASK QUALITY

4356
 4357 **Design Choice:** The scientist LLM attempts to solve its own generated tasks, with automatic cor-
 4358 rection for compilation errors and refinement prompts for logic errors.
 4359

4360 **Justification:** This self-evaluation cycle addresses a critical challenge in synthetic data generation:
 4361 ensuring that automatically generated tasks are well-formed and solvable. Recent work on synthetic
 4362 data quality (Havrilla et al., 2024) emphasizes that data quality matters as much as quantity.
 4363

4364 The iterative refinement process implements a form of minimal criterion for task quality, preventing
 4365 the task archive from accumulating broken or ambiguous tasks. By having the scientist LLM solve
 4366 its own tasks before accepting them, we create selection pressure toward tasks with clear problem
 4367 statements and unambiguous scoring functions.
 4368

4369 This also relates to the principle of “autotelic learning” in open-ended systems (Etcheverry et al.,
 4370 2021), where the system must develop its own evaluation criteria. The scientist LLM effectively
 4371 learns to generate tasks that meet implicit quality standards through its own solution attempts.
 4372

4373 N.9 HISTORICAL MODEL ARCHIVE AND TASK FORCE SELECTION

4374
 4375 **Design Choice:** AC/DC maintains a historical archive of all models every G_{task} generations and
 4376 selects the final task force by maximizing coverage over the global task archive.
 4377

4378 **Justification:** The historical archive implements a key principle from QD: maintaining a collection
 4379 of diverse high-quality solutions rather than just the current population (Pugh et al., 2016). This is
 4380 crucial because behavioral diversity at intermediate generations might not be preserved if we only
 4381 keep the final population. Some specialized models might be replaced during evolution, but could
 4382 still contribute unique capabilities to the final ensemble.
 4383

4374
 4375 The coverage-based selection for the task force directly optimizes for our practical goal: collective
 4376 problem-solving capacity across diverse tasks. This selection is independent of downstream bench-
 4377 marks, implementing the “novelty search without objectives” approach while still providing a clear
 utility-based criterion for ensemble construction.

4378 Importantly, this selection happens after coevolution completes, avoiding optimization pressure dur-
 4379 ing evolution. This prevents the kind of overfitting to specific benchmarks that could be expected in
 4380 recent evolutionary model merging work (Akiba et al., 2025). Our models are selected based on syn-
 4381 thetic task coverage, then evaluated on held-out benchmarks, ensuring genuine out-of-distribution
 4382 generalization.
 4383

4384 N.10 SYNTHESIS: AC/DC AS AN OPEN-ENDED SYSTEM

4385
 4386 Taken together, these design choices implement the key properties of open-ended systems identified
 4387 by Stanley et al. (2017) and Stanley & Lehman (2015):
 4388

- 4389 1. **Continual Generation of Novelty:** The coevolution of models and tasks produces an on-
 4390 going stream of new capabilities and challenges, with no predetermined endpoint.
- 4391 2. **Increasing Complexity:** Task difficulty adaptation and model selection pressure drive both
 4392 populations toward increasing sophistication over time (demonstrated in Fig. 1’s improve-
 4393 ment trajectory).
- 4394 3. **Cumulative Innovation:** New models build on previous models through crossover, and
 4395 new tasks build on previous tasks through adaptation, creating a stepping-stone effect where
 4396 discoveries enable further discoveries.
- 4397 4. **No Explicit Fitness Function:** While we use quality (task pass rates) and diversity (skill
 4398 vector distances) for selection, we never optimize explicitly for downstream benchmarks,
 4399 allowing unexpected capabilities to emerge.
- 4400 5. **Minimal Criteria Rather Than Objectives:** Our gibberish and impossible task filters pre-
 4401 vent completely undesired outcomes without constraining the search to predefined goals.

4402
 4403 This combination of principles, grounded in established open-endedness literature, enables AC/DC
 4404 to discover model collectives with broader and more diverse capabilities than methods that optimize
 4405 directly for benchmark performance, while maintaining lower computational costs than training
 4406 large monolithic models.

4408 O LLM USAGE DURING PAPER WRITING

4409
 4410 We leverage LLMs to assist in polishing the paper’s text and generating tables.
 4411
 4412
 4413
 4414
 4415
 4416
 4417
 4418
 4419
 4420
 4421
 4422
 4423
 4424
 4425
 4426
 4427