Evolutionary Search for Automated Design of Uncertainty Quantification Methods

Mikhail Seleznyov[1,2], Daniil Korbut[3], Viktor Moskvoretskii[4], Oleg Somov[1,5], Alexander Panchenko[2,1], Elena Tutubalina[1,6]

[1]AIRI, [2]Skoltech, [3]Amazon,
[4]EPFL, [5]MIPT, [6]ISP RAS Research Center for Trusted AI

Correspondence: seleznev@airi.net

Abstract

Uncertainty quantification (UQ) methods for large language models are predominantly designed by hand based on domain knowledge and heuristics, limiting their scalability and generality. We apply LLM-powered evolutionary search to automatically discover unsupervised UQ methods represented as Python programs. On the task of atomic claim verification, our evolved methods outperform strong manually-designed baselines, achieving up to 6.7% relative ROC-AUC improvement across 9 datasets while generalizing robustly out-of-distribution. Qualitative analysis reveals that different LLMs employ qualitatively distinct evolutionary strategies: Claude models consistently design high-feature-count linear estimators, while Gpt-oss-120B gravitates toward simpler and more interpretable positional weighting schemes. Surprisingly, only Sonnet 4.5 and Opus 4.5 reliably leverage increased method complexity to improve performance – Opus 4.6 shows an unexpected regression relative to its predecessor. Overall, our results indicate that LLM-powered evolutionary search is a promising paradigm for automated, interpretable hallucination detector design.

1 Introduction

Large Language Models (LLMs) are widely deployed as chatbots and coding assistants, however their propensity to hallucinate is a significant blocker on the path toward reliable and trustworthy general-purpose systems (Huang et al., 2025).

An important step in mitigating factual hallucinations is developing robust detectors which work across multiple domains and don't require access to ground truth facts (Vazhentsev et al., 2026). A common approach to designing such detectors is based on the uncertainty quantification (UQ) task. The core assumption is that there is a way to quantify an intrinsic quantity – LLM uncertainty – such that the estimated uncertainty value is highly correlated with hallucination occurrences.

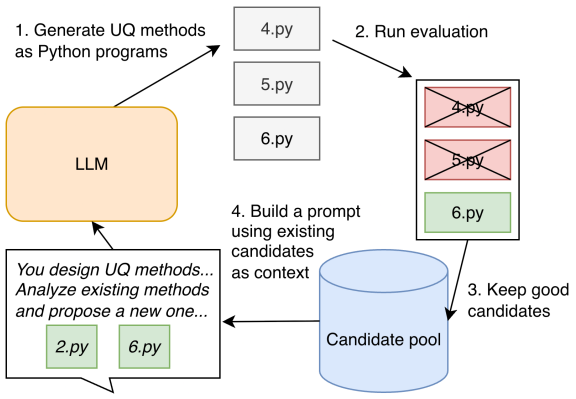

Figure 1: A visualization of LLM-powered evolutionary search pipeline, used to autonomously design uncertainty quantification (UQ) methods for hallucination detection tasks. At the start the candidate pool is initialized with a single selected baseline method.

UQ methods for LLMs typically rely on output distributions or internal representations, and achieve strong empirical performance (Fadeeva et al., 2023). However, most of these methods are manually designed by researchers, who use domain knowledge and intuition to craft often complex scoring rules, algorithms, or heuristics. Unfortunately, such hand-engineered approaches may be brittle and limited in scalability, and are often viewed as less promising than more automated alternatives (Sutton, 2019).

Meanwhile, recent progress in LLMs has enabled a new direction for scalable automation of complex tasks. AlphaEvolve (Novikov et al., 2025), in particular, highlights the potential of LLM-powered evolutionary optimization as a general framework for automated discovery. Such methods have already demonstrated progress on challenging problems, including data-center scheduling, efficient matrix multiplication subroutines, hardware circuit design, and other domains (Sharma, 2025;

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop), pages 1280–1290

Lange et al., 2025; Lee et al., 2025).

Unlike manual design or end-to-end deep learning, LLM-powered evolutionary search manages to combine two goals that have traditionally been hard to achieve at the same time. First, it offers interpretability, because one can carefully design the search space in advance. Second, it offers scalability, by taking advantage of the ever-improving capabilities of language models.

Our contributions are along two axes: we show that evolutionary search works for automated design of UQ methods, and we use it as a lens to reveal qualitative differences in how frontier LLMs approach open-ended optimization.

1. **We propose applying LLM-based evolutionary search to automatically design UQ methods for hallucination detection**. This yields explicit, human-readable scoring functions that combine the scalability of evolutionary optimization with the interpretability of hand-crafted methods.

2. **We demonstrate that evolved methods surpass strong manually-crafted baselines on atomic claim verification**, achieving statistically significant gains on 6 out of 9 datasets while generalizing robustly out-of-distribution from a single training dataset. Cross-task transfer to selective classification is limited, but in-domain evolution matches existing alternatives without requiring attention maps.

3. **We show that different frontier LLMs exhibit qualitatively distinct search strategies**, and only certain models reliably leverage increased method complexity to improve performance.

Overall, these results establish LLM-powered evolutionary search as a viable paradigm for scalable, interpretable hallucination detector design, and surface some behavioral differences between frontier models that warrant further investigation.

2 Related work

Uncertainty Quantification. Uncertainty of LLM predictions is generally a strong signal of answer correctness. Most uncertainty estimators operate on the model output probability distribution, including methods such as Sequence Probability, Mean Token Entropy, and Perplexity (Fadeeva et al., 2023). Despite their simplicity, these approaches are well aligned with the probabilistic nature of language models and remain difficult to outperform, particularly in cross-domain and cross-task transfer settings (Moskvoretskii et al., 2025).

More advanced methods leverage internal model signals, such as attention maps (e.g., RAUQ (Vazhentsev et al., 2025), AttentionScore (Sriramanan et al., 2024)) or hidden representations (e.g., SATMD and SATRMD (vaz, 2025)). Alternatively, uncertainty can be estimated through *verbalized confidence*, where the LLM explicitly reports its own confidence via prompting (Kadavath et al., 2022). All the above methods are manually designed. Some of them generalize well, but there is no silver bullet, and development of task-specific uncertainty quantification approaches still relies heavily on researchers' manual effort.

LLM-powered evolutionary search. Recent work explores using LLMs as mutation and refinement operators inside evolutionary search loops. AlphaEvolve (Novikov et al., 2025) is a prominent demonstration of this paradigm for scientific and algorithmic discovery, combining LLM-proposed code edits with automated evaluators in an iterative evolutionary pipeline. Follow-up work has focused on improving openness, efficiency, and adaptability of this framework. ShinkaEvolve (Lange et al., 2025) improves sample efficiency through better parent selection, novelty-based rejection, and adaptive model selection, while CodeEvolve (Assumpção et al., 2026) and GigaEvo (Khrulkov et al., 2025) provide open-source evolutionary coding frameworks aimed at reproducibility and broader adoption. Beyond static search, Surina et al. (2025) propose combining evolutionary search with reinforcement learning so that the LLM itself improves during search, and ThetaEvolve (Wang et al., 2025) studies test-time learning within evolving open-problem solvers. Related ideas also appear beyond code optimization: Mind Evolution (Lee et al., 2025) applies evolutionary generation and recombination to inference-time reasoning, showing that evolutionary search can be a more general mechanism for improving LLM outputs.

Program synthesis. Several papers explore application of evolution-like pipelines for generation of programs. Liu et al. (2025) search for novel LLM architectures using a multi-agent approach. Wei et al. (2025) optimize CUDA kernels for common SGLang (Zheng et al., 2024) routines, and Press et al. (2025) try to speed up general computational routines (e.g. `np.qr` or `python.gzip`).

However, to the best of our knowledge, there are no works which considered autonomous design of hallucination detectors.

3 Method

We focus on developing unsupervised and computationally efficient uncertainty quantification (UQ) methods, which take in the internal states of a small pretrained transformer model, e.g. Llama-3.1-8B-Instruct (Team, 2024), and produce a single real-valued sequence-level uncertainty estimate.

We use EvoTune framework introduced by Surina et al. (2025) since its Docker-based implementation ensures good portability. The process of evolutionary search is schematically described in Figure 1. The framework accepts four inputs: a baseline UQ method as a starting point, a dataset D for candidate evaluation, the evolution prompt, and the evolution-driving LLM. A dataset must contain (x, y) pairs, where x is a set of precomputed internal states for a given model generation (logits, residual hidden states, attention maps, or a subset of these), and $y \in \mathbb{R}$ is ground truth quality score of this generation (e.g. True/False for binary classification, ROUGE-L for summarization, etc).

Evolution prompt. The evolution prompt describes the task, inputs available for the UQ methods, provides design guidelines and includes example candidates from the candidate pool along with their performance on the dataset D. At initialization, the candidate pool only contains the baseline method. Then, the sampling procedure depends on the round index. First, the top p_i percent of candidates are preselected. Second, a probability distribution p over candidates is computed, with

$$p(\texttt{candidate}) \propto \exp\left(\frac{\texttt{candidate\_score}}{T_{\texttt{cand\_sampling}}}\right),$$

where `candidate_score` is the target metric of this candidate on the dataset D and $T_{\texttt{cand\_sampling}}$ controls how concentrated the distribution is. Note this is distinct from temperature T used for the evolution LLM.

Main evolution loop. At each round, several **candidate** methods are generated by the evolution LLM in form of Python programs, using the same prompt and non-zero temperature T. Second, target metrics are computed on the dataset D. Third, candidates are stored in the candidate pool. Finally, the **evolution prompt** is built, based on task-dependent instructions and 1-4 randomly sampled candidates. For most **evolution runs**, we set the number of evolution rounds to 500. We only sample 2 candidates per round, since we found that models frequently generate conceptually identical candidates even at temperature 1.0, which led to inefficient compute usage. Unless otherwise specified, we use Sequence Probability (Fomicheva et al., 2020) as a starting point. For the LLM model governing the generation of novel candidates, we consider Gpt-oss-120B (OpenAI et al., 2025) as a strong open-source model, as well as a number of closed-source models from Claude family: Sonnet 4.0, Haiku 4.5, Sonnet 4.5, Opus 4.5, Opus 4.6 (Anthropic, 2025a,b,d,c, 2026).

4 Benchmarks

We apply LLM-powered evolutionary search to two tasks, related to hallucination mitigation: atomic claim verification and selective prediction.

Atomic claim verification. We evaluate atomic claim verification as a binary classification task, where each claim x is assigned a label $y \in \{0, 1\}$ indicating its correctness, following Vazhentsev et al. (2026). Uncertainty scores serve as a proxy for correctness, with higher uncertainty interpreted as a higher probability of an incorrect claim. The benchmark is comprised of 9 datasets that cover diverse claim sources: human-written, rule-generated, short-form LLM generations, and excerpts parsed from long-form LLM outputs. On average, claims are 10–25 tokens long; exact per-dataset average lengths are shown in Figure 3. During evolutionary search, we use training subset of PopQA as dataset D for candidate scoring. The performance is measured by AUC-ROC.

Selective prediction. We evaluate selective prediction by assigning each input x a confidence score $s(x) \in \mathbb{R}$, which is used to rank predictions and compute performance under varying rejection thresholds. Evaluation is conducted on 8 datasets from Fadeeva et al. (2023), covering tasks such as math reasoning, machine translation, and summarization. Importantly, this benchmark includes datasets with both short- and long-form answers.

For evaluation, we first obtain model generations for each sample in each dataset, again using Llama-3.1-8B-Instruct. For comparison, average length of generations reaches 140 tokens for GSM8K, 68 tokens for SamSum, and 51 tokens

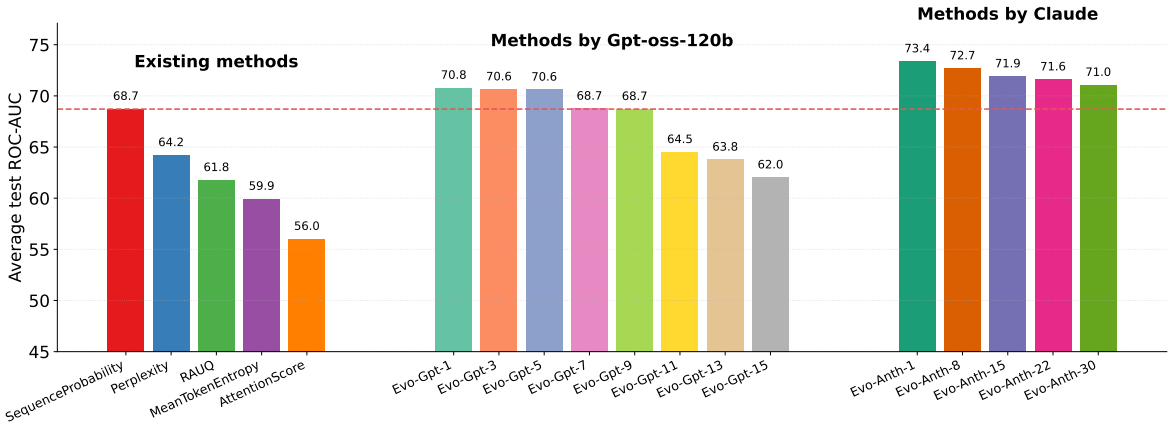

Figure 2: Average ROC-AUC across 9 hallucination detection datasets (atomic factual claims). For Claude-generated candidates, top-30 methods were selected by validation performance on PopQA dataset. We report 5 of them with test-performance ranks 1, 8, 15, 22, and 30 respectively. For Gpt-oss-generated methods, we show every second method among top-16 by validation performance on PopQA.

for TruthfulQA dataset. We then compute ground truth quality scores for each generation: ROUGE-L (Lin, 2004) for SamSum, and TruthfulQA; exact match for BabiQA, MMLU and GSM8K; BLEU (Papineni et al., 2002) for WMT14 and WMT19; and F1 score for CoQA.

Performance is measured using the Prediction-Rejection Ratio (PRR). Compared to ROC-AUC, PRR naturally accommodates both discrete and continuous quality metrics, making it suitable for diverse generation tasks. To compute PRR, first the rejection curves are built based on uncertainty estimates and ground truth quality of model predictions. Rejection curves represent average response quality as we abstain from the most uncertain predictions. PRR then is the ratio of the area between the UQ method and random baseline rejection curves to the area between the ideal and random baseline curves. We compute PRR over the first 50% of the curve, as higher rejection rates are typically impractical. The metric is normalized: PRR = 0 indicates random chance performance, while values near 1 indicate optimal performance.

5 Results

5.1 Claim Verification

We apply evolutionary search to design UQ methods for atomic claim verification. We conduct 20 runs using Gpt-oss-120B and 10 runs using Claude models, varying hyperparameters such as evolution prompt, LLM temperature, and number of candidates shown at each iteration to broadly explore the design space. For Claude-generated candidates, we select the top-3 candidates per run by validation performance on PopQA, yielding a shortlist of 30 methods. For Gpt-oss, this strategy produced many near-duplicate methods differing only in minor implementation details; we therefore manually curated a diverse subset of 16 methods among top scorers on the PopQA validation set. We then evaluate generalization on the test sets of PopQA and 8 additional atomic claim verification datasets from Vazhentsev et al. (2026). The resulting average performance is shown in Figure 2.

Autonomously designed UQ methods surpass strong baselines in claim verification. Both Gpt-oss-120B- and Claude-evolved methods outperform existing baselines by average ROC-AUC across all 9 datasets. Notably, only the PopQA training set is used for fitness evaluation; the remaining 8 datasets serve as out-of-distribution tests. Nevertheless, the evolved methods exhibit strong generalization. To assess statistical significance, we run a pairwise bootstrap difference test between all evolved methods and the strongest baseline, Sequence Probability. The results are given in Figure 7: the best evolved methods win on six and tie on one dataset.

5.2 Selective Prediction

Selective prediction presents a more challenging test of generalization for two reasons. First, methods must capture fine-grained quality distinctions (e.g., between summaries with different ROUGE-L scores) rather than a binary correct/incorrect signal. Second, the substantially longer inputs may induce

	Avg PRR	BaBI	CoQA	MMLU	Truth fulQA	GSM8K	WMT 14 de	WMT 14 fr	WMT 19 de	WMT 19 ru	SAM Sum
RAUQ	**54.6**	**79.2**	80.2	**75.9**	37.2	88.5	34.7	38.8	45.8	33.4	32.4
SP×ExpectedRank	**54.6**	78.7	**80.6**	**75.9**	39.1	90.2	34.0	37.6	44.7	32.7	**32.8**
SP×MeanEntropy	**54.6**	78.0	79.8	75.8	39.0	90.2	34.2	38.0	45.4	33.0	**32.9**
Evo-Anth-30	54.5	78.1	79.0	75.8	39.3	90.1	34.0	37.8	45.1	32.7	**32.8**
Evo-Gpt-9	54.4	**79.2**	80.5	75.8	**39.4**	89.9	33.3	37.0	44.0	32.2	32.7
Evo-Gpt-7	54.4	79.1	80.5	75.8	**39.4**	89.9	33.3	37.0	44.0	32.2	32.7
SequenceProbability	54.4	79.1	80.5	75.8	**39.4**	89.9	33.3	37.0	44.0	32.2	32.7
Evo-Anth-1	54.3	78.2	77.5	75.8	39.1	**90.4**	34.0	37.9	44.9	32.7	32.7
Evo-Anth-8	54.0	75.0	76.9	75.8	39.1	**90.4**	34.2	38.1	45.0	32.8	32.7
Evo-Anth-22	54.0	76.0	79.4	75.8	39.3	**90.3**	33.2	37.1	44.0	32.2	32.7
Perplexity	54.0	77.0	76.3	**75.9**	36.2	88.4	34.7	39.0	46.1	33.6	32.4
Evo-Anth-15	53.7	74.6	79.1	75.8	39.3	90.1	33.1	36.9	43.8	32.1	32.7
MeanTokenEntropy	53.7	73.7	76.7	75.1	36.1	88.4	**34.8**	**39.1**	**46.5**	**33.8**	32.6
Evo-Gpt-5	52.5	67.7	73.3	75.8	35.8	89.1	34.0	38.4	45.4	33.2	32.2
Evo-Gpt-1	52.5	67.7	73.1	75.8	35.8	89.0	34.1	38.5	45.4	33.3	32.2
Evo-Gpt-13	52.1	69.2	75.8	72.7	35.4	89.8	33.1	37.2	43.7	32.4	31.8
Evo-Gpt-11	51.0	72.0	72.2	70.1	34.1	86.7	31.9	36.7	42.9	31.4	31.6
AttentionScore	50.8	72.8	72.8	66.4	**39.4**	87.5	30.9	35.1	40.7	30.5	32.0
Evo-Gpt-15	50.6	72.0	71.6	70.2	33.7	85.8	31.6	36.4	42.6	30.8	31.6
Evo-Gpt-3	50.4	70.9	68.5	73.3	33.5	86.0	31.3	35.7	41.9	30.8	31.5

Table 1: PRR scores across different methods and datasets (sorted by average rank). Blue and teal: Gpt-generated methods, orange: Claude-generated methods. Teal methods are evolved specifically for selective prediction, using training subsets of CoQA and SAMSum.

distribution shift relative to the short atomic claims used during evolution. We evaluate the evolved claim-verification methods in a zero-shot transfer setting on the selective prediction benchmark and, as a complementary experiment, launch additional evolution runs using Gpt-oss-120B with training splits of CoQA and SAMSum for candidate scoring. Table 1 reports PRR values for all estimators, ranked by average performance.

Cross-task transfer is limited, but in-domain evolution remains competitive. Many methods evolved for claim verification underperform established baselines such as Sequence Probability and MeanTokenEntropy. This suggests that the binary verification objective does not fully capture the richer quality signal required for selective prediction. Interestingly, evolved methods are slightly ahead specifically on GSM8K, which has the longest average generations — despite being optimized for short atomic claims. In-domain evolution (teal) yields methods that slightly exceed or match existing alternatives on average. Notably, these methods don't require access to attention maps, which is a limitation of RAUQ approach.

6 Analysis

We structure our analysis around several concrete research questions:

RQ1: How do search strategies differ across LLMs used in the evolutionary pipeline?

RQ2: What recurring patterns emerge in autonomously designed methods?

RQ3: Which input features are most useful for the evolutionary search?

RQ4: How does method complexity relate to generalization performance across ?

RQ5: How similar are evolved methods to the strongest baseline?

6.1 RQ1: Search Strategy Differences

To impose a simplicity prior, we include a free-form constraint in the evolution prompt: *"Use up to 3 features."* Gpt-oss-120B adheres to this instruction and consistently produces solutions that rely on at most three conceptually distinct features. Claude models, by contrast, systematically violate the constraint, generating significantly more complex methods that incorporate 10–30 features. Despite this, Claude-evolved methods ultimately achieve stronger generalization.

This finding is noteworthy: classical statistical learning theory (Vapnik, 1998) predicts overfitting under high feature counts, particularly in the absence of explicit regularization. Moreover, the mechanisms commonly used to explain generalization in overparameterized deep networks such as implicit bias of SGD (Liu et al., 2020; Galanti and Poggio, 2022) and loss landscape geometry (Chiang et al., 2023) do not apply here, since evolu-

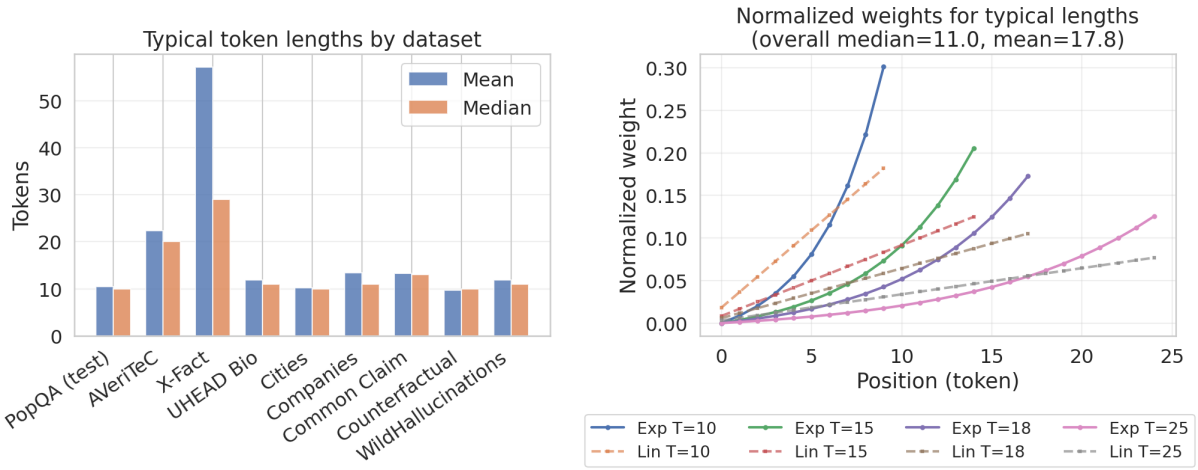

Figure 3: Left: average and median token lengths of atomic claims. Right: visualization of exponential and linear weighting for several typical claim lengths. Exponential weighting puts more emphasis on last tokens than linear weighting, especially for shorter sequences.

tionary search constructs explicit programs without gradient-based optimization. We conjecture that the combination of the LLM's prior knowledge and the discrete nature of evolutionary selection acts as an implicit regularizer. This may represent a qualitatively distinct generalization mechanism specific to LLM-driven program synthesis.

6.2 RQ2: Patterns in Evolved Methods

We observe that positional information is a recurring motif in methods optimized for atomic claim verification. Among the estimators discovered by Gpt-oss-120B, position is leveraged in several distinct forms. Namely, Evo-Gpt-1 and Evo-Gpt-2 apply exponential weighting to token-wise log-probabilities, assigning greater importance to later positions in the sequence. The resulting weighting schemes are illustrated in Figure 3. Meanwhile, Evo-Gpt-3 computes the negated correlation between log-probabilities and token positions: sequences in which log-probabilities decline over the course of generation are scored as more uncertain.

Claude finds weights distinct from gradient descent. The best-performing method developed by Claude is essentially a linear model with 36 custom features, without a single dominant signal. Thus, evolutionary search simultaneously performed feature engineering and weight tuning. A natural question is whether these weights are similar to what is found by training a logistic regression, and whether the found method can be further improved by choosing optimal weights. To answer this, we trained two logistic regressions, based on feature sets of Evo-Anth-1 and Evo-Anth-15 candidates.

	AUC	95% CI
Sequence Probability	74.1	(72.7, 75.4)
Evolution (attention maps)	75.1	(73.8, 76.7)
Evolution (logits)	78.3	(76.9, 79.6)
Evolution (hidden states)	74.1	(72.7, 75.4)
Evolution, T = 0.7	78.3	(76.9, 79.6)
Evolution, T = 1.0	78.7	(77.4, 80.0)
Evolution, T = 1.0 + domain knowledge	76.8	(75.5, 78.0)

Table 2: Ablation study, PopQA test split (candidates chosen by validation performance).

Since we initially selected top-30 candidates by validation performance, Evo-Anth-15 should be representative of an honest median method with good validation but unknown test performance.

Notably, logistic regression finds very different sets of weights compared to evolution. Correlation between vectors of coefficients found by evolution and logistic regression equals -0.23 for Evo-Anth-1, and 0.06 for Evo-Anth-15, which suggests that the two methods occupy qualitatively different basins in weight space. Nonetheless, the performance is comparable, as shown in Table 3.

6.3 RQ3: Utility of Input Features

We consider three categories of input features available from the target LLM: logits, attention maps, and hidden states from the residual stream. To determine which feature types are most beneficial in this setting, we conduct a controlled ablation study in which only a designated subset of features is

	PopQA (train)	PopQA (test)	AVeriTeC	X-Fact	UHead	Cities	Companies	Common Claim	Counter fact.	Wild Hall.
Evo-1	0.84	0.83	0.68	0.53	0.68	0.99	0.89	0.69	0.78	0.65
LogReg-1	0.86 +0.02	0.84 +0.01	0.65 −0.04	0.57 +0.04	0.69 +0.01	0.98 −0.01	0.87 −0.02	0.67 −0.03	0.77 −0.01	0.65 0.00
Evo-15	0.82	0.81	0.68	0.51	0.67	0.99	0.84	0.67	0.76	0.66
LogReg-15	0.85 +0.03	0.83 +0.02	0.65 −0.03	0.51 0.00	0.68 +0.02	0.99 −0.01	0.90 +0.06	0.68 +0.01	0.80 +0.03	0.63 −0.02

Table 3: Results of training logistic regressions on feature sets of Evo-Anth-1 and Evo-Anth-15 candidates. LogReg-X corresponds to a logistic regression employing Evo-X features. Notable, weights found via evolutionary search generalize slightly better to OOD datasets, despite both approaches only used PopQA train set. Displayed deltas sometimes disagree with AUC values due to rounding; in these cases deltas are more precise.

made available to evolved candidates. All configurations retain access to basic logit-derived statistics (per-token log-probabilities and entropies), as methods relying solely on attention or hidden-state features yielded near-random performance. In the `logits`, `attention`, and `hidden states` regimes, we additionally provide the top-512 logits per token, full attention maps, or embeddings from all tokens and layers, respectively. All runs use Gpt-oss-120B as the evolution LLM and report test ROC-AUC on PopQA.

As shown in Table 2, access to attention maps and hidden-state features does not improve performance beyond the logits-only condition. Manual inspection of generated candidates reveals that evolution frequently converges on the same rudimentary derived features — such as the entropy of attention distributions or the average Euclidean distance between consecutive token embeddings — none of which provide meaningful uplift.

We additionally investigate the effects of sampling temperature and domain knowledge injection. Higher temperature yields modestly higher top scores, likely due to increased candidate diversity. Incorporating domain knowledge into the evolution prompt, however, does not improve results (Table 2, lower half).

6.4 RQ4: Method Complexity and Generalization

A natural question is whether LLMs can design complex UQ methods without overfitting, and whether this capability scales with model size. To investigate, we measure the Spearman correlation between method complexity (operationalized as line count) and test ROC-AUC on PopQA for each evolution LLM. As shown in Figure 4, only Sonnet 4.5 and Opus 4.5 exhibit a consistent positive relationship between complexity and performance. Surprisingly, Opus 4.6 shows a regression relative

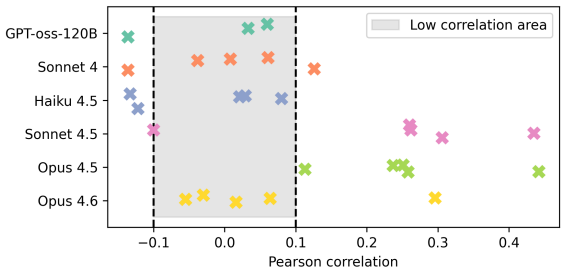

Figure 4: Correlation between method complexity and performance for different LLMs. Each dot corresponds to one evolution run.

to Opus 4.5, suggesting that the relationship between model capability and effective use of complexity is non-monotonic. We conjecture this may reflect reduced candidate diversity in Opus 4.6's outputs, though this remains untested.

To verify that this finding is not driven by outliers, we visualize the full evolution trajectory — 500 rounds of candidates plotted in (complexity, performance) space — for one representative run per model (Figure 5). Opus 4.6 indeed shows no upward trend: candidates cluster around two performance levels ($\sim$70 and $\sim$50 ROC-AUC) irrespective of complexity. Haiku 4.5 and Sonnet 4.0 exhibit a slight upward drift that is masked by outliers in the aggregate correlation statistic. Gpt-oss-120B produces a notably more dispersed candidate distribution, reflecting its broader but less targeted exploration of the design space.

6.5 RQ5: Similarity to Existing Baselines

Sequence Probability is the strongest baseline for atomic claim verification, and many evolved methods build upon it as a base signal. This raises a natural question: are the evolved methods fundamentally different from Sequence Probability, or do they constitute minor variations of the same underlying approach?

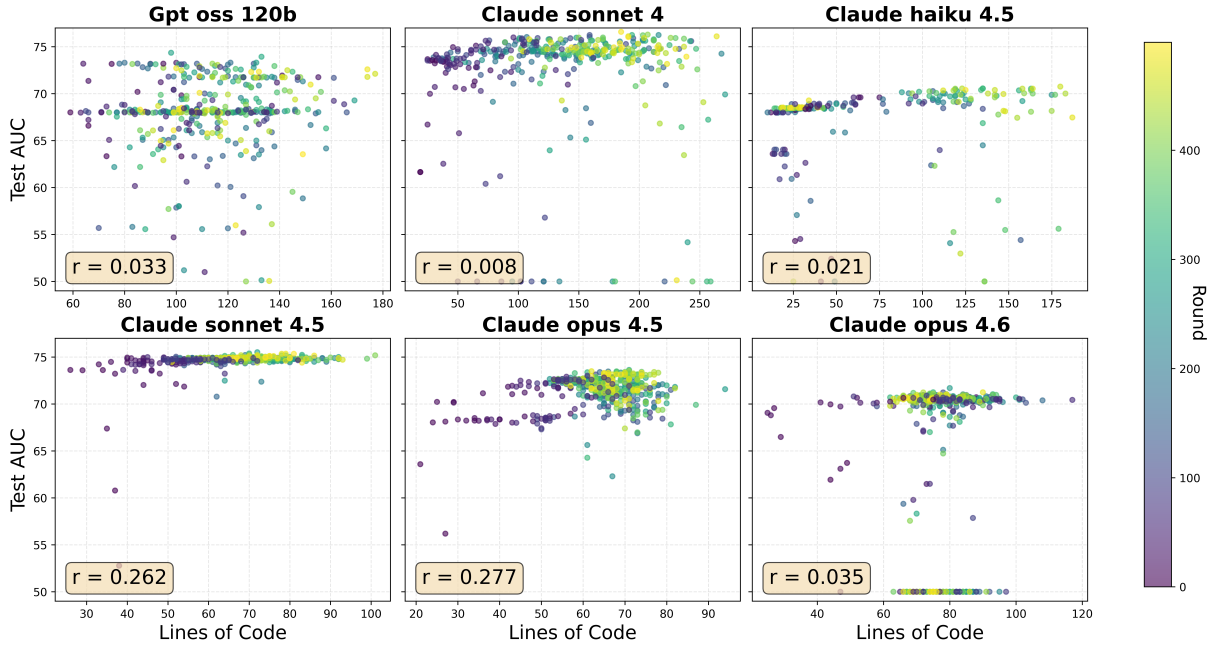

Figure 5: Evolution dynamics in (complexity, performance) coordinates for 6 different models. We use line count as the simplest proxy for method complexity; other proxies such as amount of AST nodes, number of binary and unary operators, or Halstead volume follow the same pattern.

To answer this, we introduce a data-driven notion of UQ method similarity. For each method, we compute its predictions on a dataset of N atomic claims, yielding a fixed-length score vector. We then measure Spearman correlation between these vectors for any two methods. This approach captures the degree to which two methods produce similar rankings over inputs, regardless of differences in scale or functional form.

To construct a representative evaluation set, we sample examples from PopQA, X-Fact, and UHead training subsets in equal proportions, yielding a mixture of 5,536 claims. We then compute the similarity between Sequence Probability and all candidate methods from a single Gpt-oss-120B evolution run, alongside their test performance (Figure 6). Notably, several evolved methods exhibit near-zero Spearman correlation with Sequence Probability, indicating that they produce qualitatively different uncertainty rankings while achieving comparable performance. This confirms that evolutionary search can discover diverse UQ approaches.

7 Conclusion

We applied LLM-powered evolutionary search to the automated design of UQ methods and evaluated the resulting methods across 17 datasets spanning two tasks. On atomic claim verification, evolved methods surpass strong existing baselines, achieving up to 6.7% relative ROC-AUC improvement across 9 datasets and generalizing robustly out-of-distribution despite using only a single training dataset for candidate scoring. On selective classification, direct transfer of claim-verification methods is limited; however, in-domain evolution recovers competitive performance, suggesting the framework is adaptable rather than task-specific.

Qualitative analysis reveals two distinct search strategies. Gpt-oss-120B produces interpretable, low-complexity estimators that rely heavily on token positions. Claude models instead construct high-dimensional linear estimators with 10–30 features that, unexpectedly, generalize well out-of-distribution — and whose weights are largely uncorrelated with those found by logistic regression, pointing to a qualitatively different solution regime than gradient descent. Analysis of evolution dynamics shows that the ability to leverage complexity effectively is model-dependent: Sonnet 4.5 and Opus 4.5 exhibit a consistent positive complexity-performance relationship, while Gpt-oss-120B, Sonnet 4, Haiku 4.5, and — surprisingly — Opus 4.6 do not. We conjecture that the latter may reflect reduced output diversity in Opus 4.6, though this remains to be verified.

Beyond hallucination detection, these results suggest that frontier LLMs differ not just in ca-

pability but in how they explore open-ended search spaces. This deserves further attention since LLM-powered optimization is applied to increasingly complex scientific and engineering problems.

Limitations

While the results are encouraging, several limitations should be kept in mind. Our evolutionary search is performed using feedback from a single training dataset, PopQA. This setup still leaves open the possibility that some useful design choices are tied to properties of that dataset. However, the resulting evolved methods generalize well across 8 other atomic claim verification benchmarks.

The study also focuses on a deliberately constrained search space: we optimize unsupervised, lightweight, and interpretable Python programs built over precomputed model statistics. This makes the discovered methods easy to analyze and inexpensive at inference time, but it also means that more expressive detector families, including supervised or neural approaches, are outside the scope of the current work.

In addition, our experiments rely on internal signals extracted from a single generator model, Llama-3.1-8B-Instruct. As a result, the extent to which the evolved estimators transfer across model families, scales, or decoding regimes remains an open question. A related point is that our evaluation covers two settings, atomic claim verification and selective prediction, which already provide a meaningful range of tasks, but do not capture the full diversity of hallucination phenomena encountered in long-form generation, interactive dialogue, or tool-augmented systems.

Finally, several of our observations are primarily empirical. In particular, the strong generalization of some higher-dimensional evolved estimators, as well as the differences in search behavior across LLMs, are interesting findings, but their underlying mechanisms are not yet fully understood.

Acknowledgments

The work was supported by a grant, provided by the Ministry of Economic Development of the Russian Federation in accordance with the subsidy agreement (agreement identifier 000000C313925P4G0002) and the agreement with the Ivannikov Institute for System Programming of the Russian Academy of Sciences dated June 20, 2025 No. 139-15-2025-011.

References

2025. Token-Level Density-Based Uncertainty Quantification Methods for Eliciting Truthfulness of Large Language Models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2246–2262, Albuquerque, New Mexico. Association for Computational Linguistics.

Anthropic. 2025a. Introducing claude 4.0.

Anthropic. 2025b. Introducing claude haiku 4.5.

Anthropic. 2025c. Introducing claude opus 4.5.

Anthropic. 2025d. Introducing claude sonnet 4.5.

Anthropic. 2026. Introducing claude opus 4.6.

Henrique Assumpção, Diego Ferreira, Leandro Campos, and Fabricio Murai. 2026. Codeevolve: an open source evolutionary coding agent for algorithmic discovery and optimization. *Preprint*, arXiv:2510.14150.

Ping-yeh Chiang, Renkun Ni, David Yu Miller, Arpit Bansal, Jonas Geiping, Micah Goldblum, and Tom Goldstein. 2023. Loss landscapes are all you need: Neural network generalization can be explained without the implicit bias of gradient descent. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Ekaterina Fadeeva, Roman Vashurin, Akim Tsvigun, Artem Vazhentsev, Sergey Petrakov, Kirill Fedyanin, Daniil Vasilev, Elizaveta Goncharova, Alexander Panchenko, Maxim Panov, Timothy Baldwin, and Artem Shelmanov. 2023. Lm-polygraph: Uncertainty estimation for language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023 - System Demonstrations, Singapore, December 6-10, 2023*, pages 446–461. Association for Computational Linguistics.

Marina Fomicheva, Shuo Sun, Lisa Yankovskaya, Frédéric Blain, Francisco Guzmán, Mark Fishel, Nikolaos Aletras, Vishrav Chaudhary, and Lucia Specia. 2020. Unsupervised quality estimation for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:539–555.

Tomer Galanti and Tomaso A. Poggio. 2022. SGD noise and implicit low-rank bias in deep neural networks. *CoRR*, abs/2206.05794.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 43(2):42:1–42:55.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, and 17 others. 2022. Language models (mostly) know what they know. *CoRR*, abs/2207.05221.

Valentin Khrulkov, Andrey Galichin, Denis Bashkirov, Dmitry Vinichenko, Oleg Travkin, Roman Alferov, Andrey Kuznetsov, and Ivan Oseledets. 2025. Gigaevo: An open source optimization framework powered by llms and evolution algorithms. *Preprint*, arXiv:2511.17592.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. 2025. Shinkaevolve: Towards open-ended and sample-efficient program evolution. *CoRR*, abs/2509.19349.

Kuang-Huei Lee, Ian Fischer, Yueh-Hua Wu, Dave Marwood, Shumeet Baluja, Dale Schuurmans, and Xinyun Chen. 2025. Evolving deeper llm thinking. *Preprint*, arXiv:2501.09891.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Jinlong Liu, Yunzhi Bai, Guoqing Jiang, Ting Chen, and Huayan Wang. 2020. Understanding why neural networks generalize well through GSNR of parameters. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yixiu Liu, Yang Nan, Weixian Xu, Xiangkun Hu, Lyumanshan Ye, Zhen Qin, and Pengfei Liu. 2025. Alphago moment for model architecture discovery. *CoRR*, abs/2507.18074.

Viktor Moskvoretskii, Maria Marina, Mikhail Salnikov, Nikolay Ivanov, Sergey Pletenev, Daria Galimzianova, Nikita Krayko, Vasily Konovalov, Irina Nikishina, and Alexander Panchenko. 2025. Adaptive Retrieval Without Self-Knowledge? Bringing Uncertainty Back Home. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 6355–6384, Vienna, Austria. Association for Computational Linguistics.

Alexander Novikov, Ngân Vu, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. 2025. Alphaevolve: A coding agent for scientific and algorithmic discovery. *CoRR*, abs/2506.13131.

OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, and 108 others. 2025. gpt-oss-120b and gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ori Press, Brandon Amos, Haoyu Zhao, Yikai Wu, Samuel K. Ainsworth, Dominik Krupke, Patrick Kidger, Touqir Sajed, Bartolomeo Stellato, Jisun Park, Nathanael Bosch, Eli Meril, Albert Steppi, Arman Zharmagambetov, Fangzhao Zhang, David Perez-Pineiro, Alberto Mercurio, Ni Zhan, Talor Abramovich, and 5 others. 2025. Algotune: Can language models speed up general-purpose numerical programs? *CoRR*, abs/2507.15887.

Asankhaya Sharma. 2025. Openevolve: an open-source evolutionary coding agent.

Gaurang Sriramanan, Siddhant Bharti, Vinu Sankar Sadasivan, Shoumik Saha, Priyatham Kattakinda, and Soheil Feizi. 2024. LLM-Check: Investigating Detection of Hallucinations in Large Language Models. In *Advances in Neural Information Processing Systems*, volume 37, pages 34188–34216. Curran Associates, Inc.

Anja Surina, Amin Mansouri, Lars Quaedvlieg, Amal Seddas, Maryna Viazovska, Emmanuel Abbe, and Caglar Gulcehre. 2025. Algorithm discovery with llms: Evolutionary search meets reinforcement learning. *CoRR*, abs/2504.05108.

Richard S. Sutton. 2019. The bitter lesson. `http://www.incompleteideas.net/IncIdeas/BitterLesson.html`. Accessed: 2026-03-15.

Llama Team. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley-Interscience.

Artem Vazhentsev, Maria Marina, Daniil Moskovskiy, Sergey Pletenev, Mikhail Seleznyov, Mikhail Salnikov, Elena Tutubalina, Vasily Konovalov, Irina Nikishina, Alexander Panchenko, and Viktor Moskvoretskii. 2026. Leveraging llm parametric knowledge for fact checking without retrieval. *Preprint*, arXiv:2603.05471.

Artem Vazhentsev, Lyudmila Rvanova, Gleb Kuzmin, Ekaterina Fadeeva, Ivan Lazichny, Alexander Panchenko, Maxim Panov, Timothy Baldwin, Mrinmaya Sachan, Preslav Nakov, and Artem Shelmanov. 2025. Uncertainty-Aware Attention Heads: Efficient

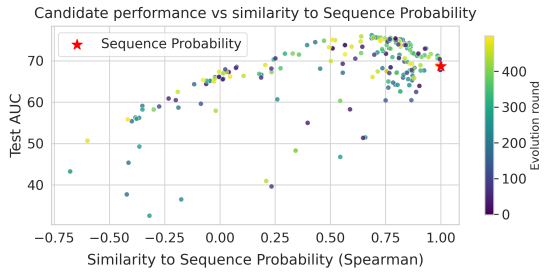

Figure 6: Methods' similarity to SequenceProbability (measured as Spearman correlation between predictions of method and SequenceProbability) versus test performance. Notably, there exist methods which have accuracy comparable to SequenceProbability, but producing significantly different sample rankings (with Spearman correlation even less than 0).

Unsupervised Uncertainty Quantification for LLMs. *CoRR*, abs/2505.20045.

Yiping Wang, Shao-Rong Su, Zhiyuan Zeng, Eva Xu, Liliang Ren, Xinyu Yang, Zeyi Huang, Xuehai He, Luyao Ma, Baolin Peng, Hao Cheng, Pengcheng He, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. 2025. Thetaevolve: Test-time learning on open problems. *Preprint*, arXiv:2511.23473.

Anjiang Wei, Tianran Sun, Yogesh Seenichamy, Hang Song, Anne Ouyang, Azalia Mirhoseini, Ke Wang, and Alex Aiken. 2025. Astra: A multi-agent system for GPU kernel performance optimization. *CoRR*, abs/2509.07506.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark W. Barrett, and Ying Sheng. 2024. Sglang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

A Additional plots for RQ5: Similarity to Existing Baselines

Sequence Probability is used as a starting point for most of the experiments throughout the paper. How similar are the evolved candidates to their starting point, and how this similarity changes over the course of evolution? To answer this question, we measure similarity between Sequence Probability and all candidates from a selected evolution run, and plot it on Figure 6. The methodology for similarity estimation is detailed in Section 6.5.

B Statistical significance testing for atomic claim verification results

To test whether the improvements of evolved uncertainty quantification methods are in ROC-AUC on atomic claim verification datasets are statistically significant, we run paired bootstrap difference test between Sequence Probability (the strongest baseline) and all other methods. We run comparisons on each dataset separately. Figure 7 shows aggregated results. Best evolved methods provide significant improvements on 6 datasets, and tie on 1.

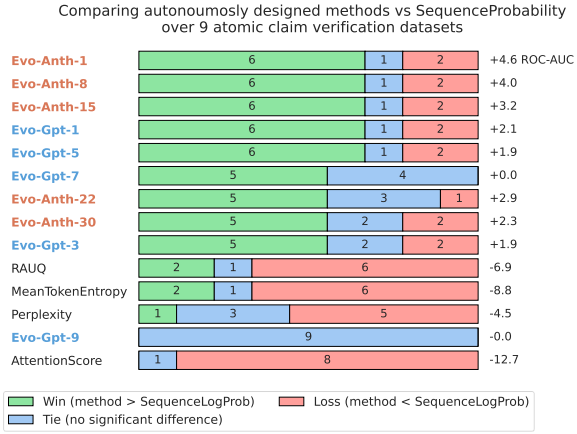

Figure 7: Best autonomously designed methods deliver statistically significant improvement over the strongest baseline (SequenceLogProb) on 6/9 datasets, and tie on one. We use paired bootstrap difference test with Bonferroni correction. Orange methods are designed using Claude models, and blue methods are designed using Gpt-oss-120B.

C Computational costs

We use 2 A100 GPUs to evaluate the candidates and run a Gpt-oss-120B instance via vLLM. A single evolution run with 500 rounds and 2 candidates per round takes about 24 hours. We set maximum amount of generated tokens to 4096, so it requires at most ∼4M output tokens per run, and typically x2-x3 times less. Including input tokens, one evolution run with Sonnet 4.5 costs about 50 USD.