

It Knows More Than It Can Say: Self-Evolving Prompts for Extremely Low-Resource Languages

Gaurav Sarkar

IIIT Hyderabad

gaurav.sarkar@research.iiit.ac.in

Jay Gala

Microsoft Research

t-jaygala@microsoft.com

Abstract

Structured prompting can make large language models (LLMs) speak languages that are virtually absent from their training data: for Tulu, a Dravidian language with over two million speakers, a hand-crafted five-layer prompt reduces vocabulary contamination from the dominant neighbour Kannada from 80% to 5% with zero parameter updates. But that prompt took native speakers and targeted elicitation sessions to build: an expert bottleneck that cannot scale to the thousands of languages in the same situation. We introduce CIPHER (Contrastive Iterative Prompt Heuristic Evolution for Resource-scarce languages), which removes the expert from the loop by making the model be its own linguist. A single LLM plays every role: it translates, contrastively diagnoses its own errors against references, abstracts recurring defects into canonical linguistic rules, routes each rule to the prompt layer that repairs it, and generates judge-filtered self-play data, while an outer evolutionary loop mutates the prompt *architecture* itself under Pareto selection, including discovering new layer types. From 250 translation pairs and no human curation, CIPHER mines 3,471 typed error records, eliminates contamination at every scale we test (47%→0% for a 3B model), raises grammar accuracy from 28% to 44% with GPT-5, and autonomously invents a contamination-filter layer that yields the best architecture found. The evolved prompt is a compact, human-auditable linguistic artifact, produced for API-hours instead of expert-weeks.

1 Introduction

The top ten languages account for roughly 90% of LLM training data, while the remaining 7,000+ languages share the rest (Joshi et al., 2020). For most of these, LLM-based translation lags far behind high-resource quality (Robinson et al., 2023). Languages that share a script and lexical ancestry with a dominant relative fail in a particularly decep-

tive way: the model does not fall silent, it answers fluently *in the wrong language*. Asked for Tulu, a Dravidian language of coastal Karnataka with over two million speakers but minimal digital presence, models produce Kannada-inflected pastiche, a failure mode Devadiga and Chopra (2026) term *vocabulary contamination*.

Devadiga and Chopra (2026) showed this is fixable without touching model weights: a five-layer structured prompt (identity, negative constraints, grammar rules, few-shot examples, self-verification; ~2,800 tokens) cuts contamination from 80% to 5% and lifts grammatical accuracy from 18% to 85%. The catch is how the prompt was built: three native speakers documented complete paradigms for 15 verbs (48 forms each), an eight-case system, and a 50-word negative-constraint list through targeted elicitation sessions, followed by four rounds of manual prompt iteration. The artifact works; the process is the bottleneck. FLORES+ contains fewer than 1,000 parallel Tulu sentences (NLLB Team et al., 2022), and thousands of languages sit in the same position with no linguist team on call.

Our starting observation is an asymmetry: a model that cannot *produce* correct Tulu can often still *recognize and articulate* how its output differs from a reference. The knowledge needed to write the structured prompt is partially latent in the model and partially recoverable from a small set of translation pairs, provided there is a procedure that extracts it, verifies it, and stores it somewhere durable. CIPHER is that procedure. One LLM serves as generator, contrastive error analyst, rule abstractor, synthetic-data author, quality judge, and prompt architect. An **inner loop** evolves the *content* of a typed, layered prompt: translate a development batch, diagnose each failure with Contrastive Linguistic Error Mining (CLEM), cluster defects into canonical rules, and route every rule to the layer that repairs its error type. An **outer loop**

evolves the prompt’s *structure*: it mutates layer order, token budgets, and formats, and, when defects fit no existing layer, proposes entirely new layer types, selecting survivors by Pareto rank over accuracy, contamination, and token cost.

Our contributions:

- **CLEM**: automated contrastive error diagnosis that converts each failed translation into a typed defect record with a deterministic mapping from error type to prompt layer (§3.3).
- **Architecture evolution with layer discovery**: to our knowledge the first system that learns not only the content but the *structure* of a linguistic prompt, including inventing new layer types from unroutable errors (§3.4).
- **Evidence across three backbones** that self-evolution eliminates contamination at every scale (GPT-5, Qwen2.5-72B, Llama-3.2-3B), while grammar gains track analyst capacity; the single best architecture contains a layer no human designed (§5).

2 Related Work

Prompting LLMs into unseen languages. MTOB translates Kalamang from a grammar book in context (Tanzer et al., 2024); LingoLLM pipelines dictionaries and morphological analyzers into prompts (Zhang et al., 2024b); DiPMT++ adapts to an unseen language from a dictionary and a small parallel set (Zhang et al., 2024a); Devadiga and Chopra (2026) hand-engineer the layered Tulu prompt we take as our target artifact. All of these consume linguistic resources that experts already wrote. CIPHER instead *synthesizes* the resource: raw translation pairs in, structured description out.

Multilingual reasoning. Ranaldi and Pucci (2025) combine structured reasoning passages with self-refinement to improve multilingual reasoning and cross-lingual consistency. CrossPAL (Ranaldi et al., 2024) aligns program-based intermediate reasoning across languages through two-stage prompting, while its self-consistent variant ensembles reasoning paths across languages. Both studies highlight the role of explicit intermediate structure in supporting multilingual reasoning. CIPHER targets low-resource language production instead: its intermediate representations are reference-grounded linguistic defects and canonical rules, stored in a persistent grammar-and-lexicon prompt. It evolves that prompt’s content and architecture to address

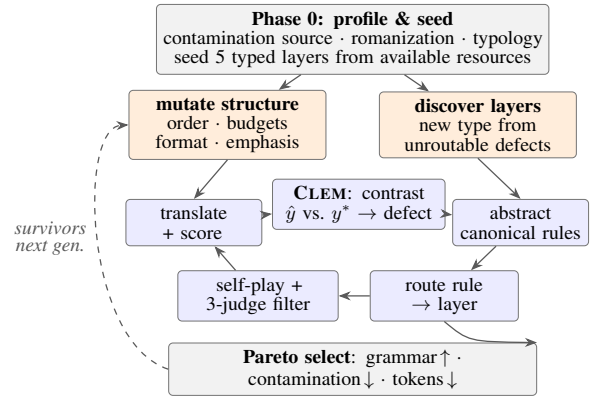


Figure 1: CIPHER. The inner loop (blue) evolves layer *content*: translate, contrastively diagnose defects (CLEM), abstract canonical rules, route each rule to the layer that repairs it, refresh few-shot data by judged self-play. The outer loop (orange) evolves *structure* by mutation and layer discovery; candidates compete under Pareto selection.

contamination and grammatical errors, rather than optimizing multi-step reasoning performance.

Automatic prompt optimization. APE (Zhou et al., 2023), OPRO (Yang et al., 2024), EvoPrompt (Guo et al., 2024), Promptbreeder (Fernando et al., 2023), and DSPy (Khatab et al., 2024) search over instructions or exemplars against a scalar metric. These methods treat the prompt as an opaque string or fixed pipeline; none produce typed linguistic knowledge or change the prompt’s architecture. CIPHER’s search space is a structured memory (typed layers with budgets, formats, and positions) and its update signal is a linguistic diagnosis, not just a score.

Self-improvement and evolving memory. Self-Refine (Madaan et al., 2023) and Reflexion (Shinn et al., 2023) generate transient, episode-level feedback. Recent work on agent memory meta-evolves the memory architecture itself alongside the experience it stores (OPPO AI Agent Team and LV-NUS Lab, 2025). CIPHER brings that content/structure separation to language acquisition, where the “memory” is a persistent, human-readable grammar-and-lexicon prompt distilled from the model’s own mistakes. Intrinsic self-correction is known to be unreliable (Huang et al., 2024); CIPHER instead grounds every diagnosis in a reference translation: the signal is contrast, not introspection.

3 CIPHER

3.1 Prompts as typed linguistic memory

A prompt architecture P is an ordered list of layers $\ell_i = (\text{type}, \text{content}, \text{budget}, \text{format}, \text{emphasis})$, initialized with the five types of Devadiga and Chopra (2026) under a 2,800-token budget. Given development pairs $\mathcal{D}$, we score a prompt with a single scalar, its *linguistic fitness*

$$F(P) = 0.3(1-c) + 0.3g + 0.15f + 0.25\text{chrF},$$

where c is the contamination rate (fraction of outputs containing any watchlisted Kannada form; a string match, no judge), g is grammar accuracy (LLM-judged score ≥ 0.7), f is judged fluency, and chrF is character n -gram F_2 (Popović, 2015).

3.2 Phase 0: profiling and cold start

The system first probes the backbone with target-language phrases to identify the contamination source (Kannada), pick a romanization scheme, and elicit typological priors (SOV, agglutinative, postpositions). Layers are then seeded from whatever resources exist: translation pairs are the only requirement; a Swadesh list, a bilingual dictionary, curated constraint pairs, or a grammar sketch are each optional accelerants. This graceful degradation matters: most languages will have some, not all, of these.

3.3 Inner loop: content evolution

Each of $T=7$ iterations: **(1) Evaluate** P on $\mathcal{D}$. **(2) Diagnose**: for every failure (contaminated or judged < 0.7), CLEM presents the analyst with the triple (source, prediction $\hat{y}$, reference y^*) and asks not *whether* but *why* they differ, emitting a typed defect record: error type $\in \{\text{contamination, morphological, syntactic, register, unknown-vocab, semantic, custom}\}$, the specific wrong form, the correct form, and the violated rule. Confirmed contaminations are appended to the watchlist, so the contamination metric sharpens as mining proceeds. **(3) Abstract**: defects are clustered into canonical rules (fifteen instances of the same Kannada intrusion become one constraint entry), prioritized by support. **(4) Route**: a fixed map sends each rule to the layer that repairs it (contamination $\rightarrow$ negative constraints; morphological/syntactic $\rightarrow$ grammar; unknown-vocab/semantic $\rightarrow$ few-shot; register $\rightarrow$ self-verification), with per-layer budgets enforced by compression. **(5) Self-play**: the model generates

pairs that specifically exercise the most frequent error patterns; each pair must pass a watchlist check and a three-judge quality vote before entering the few-shot pool. The best content (smoothed F) is kept.

3.4 Outer loop: architecture evolution and layer discovery

Every generation, survivors spawn candidates by structural mutation: swap layer positions (negative constraints are kept early, consistent with the salience finding of Devadiga and Chopra, 2026), transfer 50–200 budget tokens between layers, or flip a layer’s format (prose/table/examples-only) and emphasis. Each candidate is optimized by the inner loop, then ranked by Pareto dominance over (grammar $\uparrow$, contamination $\downarrow$, prompt tokens $\downarrow$), tie-broken by F ; the top-1 survives.

The novel step is **layer discovery**. Defects tagged *custom* plus persistent low-confidence diagnoses accumulate as evidence that the current taxonomy is missing a slot. The architect then proposes a new layer (name, purpose, content template, position, budget), and the augmented architecture joins the generation as one more candidate, with no special treatment: it survives only if it wins on the same Pareto front. The five-layer taxonomy is thus a prior, not a constraint.

4 Experimental Setup

Language and data. Tulu, with Kannada as the interfering neighbour. We use 250 community-verified English–Tulu pairs from the public release of Devadiga and Chopra (2026), plus a 100-item Swadesh list, 53 curated Kannada $\rightarrow$ Tulu constraint pairs, a 586-entry dictionary, and a 1.5 KB grammar sketch. No native speaker participates at any point.

Protocol. Following the iterative-refinement protocol of Devadiga and Chopra (2026), optimization is driven by a fixed development set sampled once (25 pairs for GPT-5, 15 for the smaller backbones); its references are the only supervision CLEM ever sees, and all metrics are reported on it (see Limitations). The remaining pairs seed few-shot content and are extended by self-play. GPT-5 runs the full search (3 generations $\times$ 3–4 candidates $\times$ 7 inner iterations, 7.1 h wall-clock); Llama-3.2-3B (Dubey et al., 2024) runs a reduced budget (2 generations, ≤ 4 candidates, 3 inner iterations; ≈ 13 min); Qwen2.5-72B (Qwen Team, 2024) pro-

	Cont.↓	Gram.↑	chrF↑	F↑
<i>GPT-5 (full evolution, 25 dev pairs)</i>				
seeded init (5 layers)	8.0	28.0	.409	.574
CIPHER (6 layers)	0.0	44.0	.520	.651
<i>Llama-3.2-3B (reduced budget, 15 dev pairs)</i>				
seeded init	46.7	26.7		.324
CIPHER	0.0	6.7 [†]		.565
<i>Qwen2.5-72B (paired, same 15 pairs & instance)</i>				
minimal instruction [‡]	40.0	6.7	.155	
CIPHER	20.0	6.7	.247	

Table 1: Main results (contamination and grammar in %). Contamination and chrF are judge-free; grammar and fluency are LLM-judged. [†]Judged by the 3B model itself and unreliable at that scale (its selection-time evaluations reached 40–47%); see Limitations. [‡]An independent generation-and-judging run of this baseline reproduced contamination exactly (40.0) while judged grammar varied (26.7; chrF .118, F .388).

Generation best	F↑	Cont.↓	Gram.↑
gen. 1	.627	0.0	36.0
gen. 2	.621	16.0	40.0
gen. 3, twin w/o discovery	.483	24.0	16.0
gen. 3, + discovered layer	.651	0.0	44.0

Table 2: Architecture search (GPT-5). In generation 3 the discovery-augmented candidate and its unaugmented twin share the same parent and inner-loop protocol; the invented *Lexical Distinction & Contamination Filter* layer separates them (single paired run; see text).

vides a paired comparison in which the *same model instance* answers the *same 15 pairs* under a minimal instruction (“Respond in Tulu.”) and under its CIPHER-evolved prompt. In every run a single model fills all roles: generator, analyst, judges, and architect.

5 Results and Analysis

Contamination is solved by self-evolution at every scale. Across backbones, CIPHER drives watchlist contamination to or toward zero (Table 1): 8→0% for GPT-5, 46.7→0% for the 3B model, and 40→20% in the paired 72B comparison, where chrF also rises 59% relative (.155→.247); the 40% baseline replicates exactly across two independent runs of the same setup. This mirrors the expert study’s finding that negative constraints transfer consistently across architectures, except that here the constraint list was mined, not curated: of 3,471 defect records extracted by CLEM, 36% are contaminations, and each confirmed one both patches the prompt and sharpens the metric’s watchlist. Since the watchlist only grows during mining, the final

0% is scored by a stricter detector than the initial 8%, i.e., measured conservatively.

Grammar tracks analyst capacity. GPT-5 lifts grammar accuracy from 28% to 44% while shedding contamination entirely; the 3B model, whose diagnoses and judgments are themselves noisy, fixes contamination (a lexical, checkable property) but cannot reliably bootstrap morphosyntax. The remaining defect mass is 19% morphological, 15% syntactic, 17% unknown-vocabulary, and 13% semantic. The mined rules are recognizably the ones a field linguist would write: “*for impersonal present states use existential undu, not ul-lae*”; “*purpose is the dative -gu on the verb root, not the verbal noun -vudu*”; “*vanas is ‘lunch’; madhyāna is a Kannada loan*”. None was written by one.

Layer discovery wins the search. The best architecture is not a re-weighted copy of the human design (Table 2). In generation 3 the architect, prompted only by defects that fit no existing layer, invented a *Lexical Distinction & Contamination Filter*: an 800-token layer of confusable-form mappings and decision rules for existential copulas, placed beside the grammar rules. Against its unaugmented twin, the augmented candidate moves dev-set fitness from .483 to .651, contamination from 24% to 0%, and grammar from 16% to 44%. This is a single paired run with divergent inner-loop paths (promising evidence, not a controlled ablation); still, the search’s winner is an architecture no human proposed. Evolution also quietly re-allocated budget, shrinking identity (200→137 tokens) and growing self-verification (200→263), and the final prompt spends 2,345 of its 2,800-token budget.

Qualitatively, the failure mode changes in kind, not degree. Under a minimal instruction, the 72B baseline answers “*When is your birthday?*” with looping pseudo-Tulu followed by an English apology. The evolved GPT-5 system answers “*I want to go to the beach*” with *yenku beach-gu poregu undu* (reference: *yenku beachgu poregu undu*), and “*This quality is not good*” with *undetha quality aath edde iiji*, matching the reference exactly. The artifact behind this is a readable 2.3k-token document a native speaker could audit in an afternoon: CIPHER turns the expert from author into reviewer, and the per-language cost from weeks of elicitation into roughly seven API-hours.

6 Conclusion

CIPHER shows that the structured prompts previously hand-built for extremely low-resource languages can instead be *grown*: the same LLM that fails to speak a language can mine its own failures into rules, route those rules into a typed prompt memory, and evolve the memory’s very architecture; our best run discovered a layer type no human proposed. Contamination is eliminated without any expert labor; grammar improves with analyst strength and remains the open frontier. We release code, mined rules, and evolved prompts; native-speaker auditing of machine-grown grammars is the natural next step.

Limitations

Development-set evaluation. Following the iterative protocol of the expert study we replicate, all metrics are computed on the fixed pair set that also drives error mining; the references are the loop’s supervision, so results measure closed-loop knowledge extraction rather than fully held-out generalization. Two mitigating observations: the two judge-free metrics (watchlist contamination, chrF) improve alongside judged ones, and the mined artifacts are canonical rules rather than memorized sentences. A held-out test set and native-speaker evaluation are the immediate next step. **Judge circularity.** Grammar and fluency are scored by the same model being improved; at 3B scale this is visibly unreliable (Table 1), and even at larger scales LLM judges have known biases (Zheng et al., 2023). Concretely, two independent judging runs of the identical Qwen baseline agreed on contamination (40.0% both) but scored grammar at 26.7% vs. 6.7%; our conclusions therefore rest on the judge-free metrics wherever possible. **Scale of evidence.** One language, 15–25 evaluation pairs per run, single seeds, no significance testing; API cost bounded the search budget. **Baselines.** We compare against minimal instructions and our own seeded initialization; re-running the expert prompt under our metrics, many-shot in-context baselines, and generic prompt optimizers (e.g., Yang et al., 2024; Khattab et al., 2024) are the right next comparisons. **Rule quality and the expert gap.** Some mined constraints are noisy or wrong, and self-play can amplify errors the judges miss. The 85% grammar accuracy of the fully expert-curated prompt is well above our 44%, but the numbers are not directly comparable (different models, evaluation

sets, and human vs. LLM judging), and 44% is a lower bound under a deliberately small search budget: with 25 evaluation pairs each example is worth 4 points, intermediate iterations reached 56%, and the budget–performance curve remains unexplored. Scaling that budget naively would, however, partly buy overfitting to the development set rather than real gains. We view the expert and the loop as complementary: CIPHER drafts, experts verify.

Ethics Statement

CIPHER aims to widen language technology access for communities whose languages lack digital resources. Machine-grown linguistic descriptions can nonetheless encode errors; outputs should be labelled as machine-generated and validated with native speakers before any use that represents the language to learners or the public. All Tulu data used here comes from the publicly released, community-verified resources of Devadiga and Chopra (2026).

References

- Prathamesh Devadiga and Paras Chopra. 2026. Making large language models speak Tulu: Structured prompting for an extremely low-resource language. *arXiv preprint arXiv:2602.15378*.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2023. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*.
- Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2024. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*.
- Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*.
- Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293.

- Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. DSPy: Compiling declarative language model calls into self-improving pipelines. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*.
- NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, et al. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.
- OPPO AI Agent Team and LV-NUS Lab. 2025. MemEvolve: Meta-evolution of agent memory systems. *arXiv preprint arXiv:2512.18746*.
- Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395.
- Qwen Team. 2024. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*.
- Leonardo Ranaldi and Giulia Pucci. 2025. [Multilingual reasoning via self-training](#). 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 11566–11582, Albuquerque, New Mexico. Association for Computational Linguistics.
- Leonardo Ranaldi, Giulia Pucci, Barry Haddow, and Alexandra Birch. 2024. [Empowering multi-step reasoning across languages via program-aided language models](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12171–12187, Miami, Florida, USA. Association for Computational Linguistics.
- Nathaniel R. Robinson, Perez Ogayo, David R. Mortensen, and Graham Neubig. 2023. ChatGPT MT: Competitive for high- (but not low-) resource languages. In *Proceedings of the Eighth Conference on Machine Translation (WMT)*.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*.
- Garrett Tanzer, Mirac Suzgun, Eline Visser, Dan Jurafsky, and Luke Melas-Kyriazi. 2024. A benchmark for learning to translate a new language from one grammar book. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*.
- Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*.
- Chen Zhang, Xiao Liu, Jiuheng Lin, and Yansong Feng. 2024a. Teaching large language models an unseen language on the fly. In *Findings of the Association for Computational Linguistics: ACL 2024*.
- Kexun Zhang, Yee Man Choi, Zhenqiao Song, Taiqi He, William Yang Wang, and Lei Li. 2024b. Hire a linguist!: Learning endangered languages in LLMs with in-context linguistic descriptions. In *Findings of the Association for Computational Linguistics: ACL 2024*.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems 36 (NeurIPS), Datasets and Benchmarks Track*.
- Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*.