Computational Linguistics and Intellectual Technologies:
Proceedings of the International Conference “Dialogue 2026”

June 24–26, 2026

RIDER: Reproducible Evolutionary Prompt Search for NLP with Adaptive Operator Allocation

Daglar Dragomirov
ITMO University
Saint Petersburg, Russia
daglar.dragomirov@gmail.com

Ivan Blekanov
Saint Petersburg State University
Saint Petersburg, Russia
i.blekanov@spbu.ru

Nikita Kulin
ITMO University
Saint Petersburg, Russia
242106@niuitmo.ru

Sergey Muravyov
ITMO University
Saint Petersburg, Russia
smuravyov@itmo.ru

Abstract

We present RIDER (Reflective Iterative Diversity-Enhanced Reasoning), a black-box framework for automatic prompt optimization. RIDER evolves a population of natural-language instructions with nine prompt operators, allocates search budget by Windowed Thompson Sampling, and feeds current error cases back into mutation. The evaluation is designed for reproducible same-model comparison: baselines use the same task model, data partition, evaluator, and comparable search budget; final scores are computed on full held-out test partitions; the main result is the single best prompt rather than an ensemble; BERTScore uses `microsoft/deberta-xlarge-mnli`; CommonGen uses multi-reference scoring; and the pipeline is covered by 1069 automated tests. On six NLP benchmarks and five model families, RIDER obtains 23 wins, 0 losses, and 0 ties against ZeroShot, APE, EvoPrompt-GA, EvoPrompt-DE, and PromptBreeder. The average margin over the strongest baseline is 7.2 percentage points. On CodeSearchNet, RIDER obtains stable BERTScore values of 0.769–0.787.

Keywords: prompt optimization, evolutionary computation, large language models, Thompson Sampling, reproducible evaluation

DOI: 10.290032075-7182-2026-24-89-96

RIDER: воспроизводимый эволюционный поиск промптов для задач NLP с адаптивным распределением операторов

Даглар Драгомиров
Университет ИТМО
Санкт-Петербург, Россия
daglar.dragomirov@gmail.com

Иван Блеканов
Санкт-Петербургский государственный университет
Санкт-Петербург, Россия
i.blekanov@spbu.ru

Никита Кулин
Университет ИТМО
Санкт-Петербург, Россия
242106@niuitmo.ru

Сергей Муравьёв
Университет ИТМО
Санкт-Петербург, Россия
smuravyov@itmo.ru

Аннотация

В статье представлен RIDER — black-box фреймворк автоматической оптимизации промптов. RIDER эволюционирует популяцию инструкций с помощью девяти операторов, распределяет бюджет через оконное сэмплирование Томпсона и передаёт ошибки текущего лучшего prompt-а в последующие мутации. Экспериментальный протокол построен как воспроизводимое same-model сравнение: все baseline-ы используют одну модель, разбиение данных, evaluator и сопоставимый бюджет; финальная оценка проводится на полных отложенных test-разделах; основной результат считается по одному лучшему prompt-у, а не по ансамблю; BERTScore использует microsoft/deberta-xlarge-mnli; CommonGen оценивается по нескольким эталонам; корректность контура поддерживается 1069 тестами. На шести NLP-бенчмарках и пяти модельных семействах RIDER даёт 23 победы, 0 поражений и 0 ничьих относительно ZeroShot, APE, EvoPrompt-GA, EvoPrompt-DE и PromptBreeder. Средний отрыв от сильнейшего baseline-а составляет 7,2 процентных пункта. На CodeSearchNet RIDER получает устойчивые значения BERTScore около 0,769–0,787.

Ключевые слова: оптимизация промптов, эволюционные вычисления, большие языковые модели, сэмплирование Томпсона, воспроизводимая оценка

1 Introduction

Prompt quality strongly affects large language models (LLMs). Small wording or few-shot-order changes can substantially alter task performance (Shin et al., 2020; Lu et al., 2022), so prompt design is a discrete black-box optimization problem: the objective is observed only after querying a model, the search space is combinatorial, and closed APIs rarely expose gradients (Chen et al., 2023; Yang et al., 2024).

Automatic Prompt Engineer (APE) (Zhou et al., 2023) generates and ranks instructions but performs little iterative search. EvoPrompt (Guo et al., 2024) evolves prompts with a fixed operator strategy. PromptBreeder (Fernando et al., 2023) evolves prompts and mutation prompts, but has limited diversity control and no adaptive operator budget. RIDER addresses these gaps by treating optimization as a population process with feedback.

The evaluation is deliberately conservative about evidence. Prompt optimization can look stronger than it is when tuned on tiny validation slices, evaluated on clipped test subsets, compared against under-budget baselines, or reported as an ensemble against single-prompt methods. We therefore separate the algorithmic claim, adaptive operator allocation improves search, from the empirical claim, the improvement remains under the same model, data partition, evaluator, and comparable query budget.

Our contribution is a reproducible prompt-optimization framework with three core mechanisms:

1. **Adaptive operator selection.** Nine prompt operators compete for budget through Windowed Thompson Sampling (Thompson, 1933; Auer et al., 2002), with an exploration floor and a per-generation budget cap.
2. **Error-directed evolution.** The current best prompt is evaluated, its failure cases are collected, and operators receive these errors as context for targeted mutation (Pryzant et al., 2023; Wang et al., 2024).
3. **Evaluate-first retention.** Parents and offspring are compared only after all new candidates are evaluated. Diversity is a tiebreaker, not a pre-evaluation filter.

We report top-1 prompt metrics rather than ensemble voting, include prompt examples, disclose budgets, and add ablations. A reproducibility package accompanies the paper; it includes run manifests, prompt hashes, selected prompts, row-level CSVs, and scripts for regenerating the tables and figures.

2 RIDER

RIDER maintains $N = 12$ prompts for $G = 8$ generations. Each generation selects operators, creates offspring, evaluates them on validation data, merges offspring with parents, and keeps the top prompts by fitness with a semantic-diversity tiebreaker. The final reported score is the single best prompt. A k-DPP ensemble exists for deployment, but is not used in the main comparison to avoid an unfair advantage over single-prompt baselines.

Operators. The active pool contains nine regular bandit arms:

`eda_mutation`, `eda_rank_index`, `zero_order`, `de_mutation`, `ga_mutation`, `reflection_crossover`, `opro_trajectory_mutation`, `contrastive_error_decomposition`, and `semantic_paraphrase`.

VORTEX is a separate stagnation-triggered operator rather than a regular bandit arm. Operators with negative validation contribution are not active bandit arms in the final search configuration.

Protocols. RIDER adds nine protocol-level mechanisms. PRISM performs successive halving with Racing/F-Race for exact-match and F1 tasks, while BERTScore and macro-F1 use direct full evaluation to avoid noisy early elimination. NEXUS selects high-variance validation examples. PHASE REACTOR controls temperature and offspring budget. OPERATOR FORGE stores top outputs; GENESIS stores lessons from successful mutations. ECHO repeats compatible instructions. CHIMERA co-evolves instructions with few-shot examples on generative tasks. MARS CONFIG routes protocols by metric. VORTEX injects a paradigm shift under stagnation.

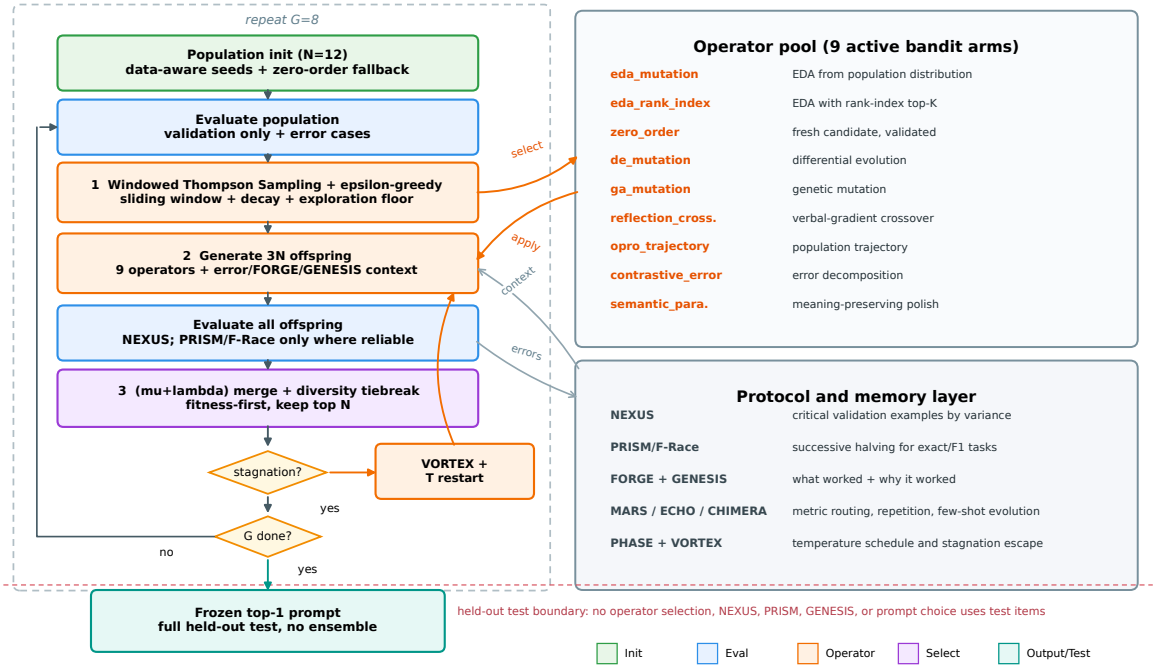

Figure 1: RIDER architecture and evaluation flow. The held-out test partition is outside the evolutionary loop; it is used only after the top-1 prompt has been selected.

Adaptive selection. For each operator i, RIDER maintains a Beta posterior and samples $\theta_i \sim \mathrm{Beta}(\alpha_i, \beta_i)$. The operator with the highest sampled value is selected unless ϵ-greedy exploration fires. Rewards are computed from positive fitness deltas, with an extreme-value credit for rare large improvements. A sliding window and decay prevent early failures from permanently suppressing an operator.

Optimization loop. The loop is evaluate-first. Candidate prompts may use diverse temperatures, few-shot examples, or error contexts, but enter the population only after validation fitness is measured. RIDER merges parents and evaluated offspring, sorts by fitness, and uses diversity only to break near-ties, preventing unusual candidates from being discarded before they are tested.

Why the components are not redundant. Thompson Sampling decides which operator receives budget; NEXUS decides which validation examples are most informative; PRISM decides how much evidence is needed before eliminating weak offspring on metrics where small batches are reliable; GENESIS and OPERATOR FORGE decide what context future operators see. They act at different levels. The ablations show that quality is driven mainly by adaptive allocation and error-directed operators, while PRISM is mainly an efficiency mechanism.

3 Experimental Protocol

Tasks and metrics. We use GSM8K (Cobbe et al., 2021) with exact match, AG_News (Zhang et al., 2015) with macro F1, SQuAD 2.0 (Rajpurkar et al., 2018) with token F1, and CommonGen (Lin et al., 2020), XSum (Narayan et al., 2018), and CodeSearchNet (Husain et al., 2019) with BERTScore F1 (Zhang et al., 2020). BERTScore is computed with `microsoft/deberta-xlarge-mnli`; CommonGen uses all available references and takes the maximum BERTScore over references. These metric choices are applied to all methods, not only to RIDER.

Models and baselines. The final matrix contains 23 dataset–model combinations over five model families: Qwen 3 235B, GPT-4.1, Gemini 3 Flash, Claude Haiku 4.5, and DeepSeek-V3.2. Baselines are ZeroShot, APE, EvoPrompt-GA, EvoPrompt-DE, and PromptBreeder. Baseline configurations include APE paraphrasing rounds, EvoPrompt temperature-sweep initialization, and PromptBreeder self-referential hypermutation. RIDER cross-experiment memory is disabled in the main fair-compare runs.

Data scale and leakage control. Search uses the full validation/development partition exposed by

each benchmark loader, and final metrics are computed on the full held-out test partition. The six search partitions contain 7563, 121452, 131896, 4067, 11469, and 23387 examples; the corresponding held-out test partitions contain 1335, 7692, 12017, 1515, 11471, and 22444 examples. Test items are never used for operator selection, NEXUS, PRISM, GENESIS, early stopping, or prompt selection. Validation examples choose prompts within a fixed run; they are not reported as final performance. BERTScore and macro-F1 tasks bypass PRISM because small batches can invert rankings or destroy class structure. Exact-match and token-F1 tasks use PRISM only inside validation. The final test pass is a single frozen evaluation of the selected top-1 prompt.

Budget and reproducibility. RIDER uses 1780–3320 search calls before the final test pass, depending on whether PRISM can eliminate weak offspring early. Iterative baselines use 2277–2480 search calls in the matched configuration. The final full-test pass then adds the same one task-model call per test item for every method. All methods share the same task model, data split, evaluator, and greedy decoding. The implementation contains 1069 collected tests, including baseline fidelity, multi-reference BERTScore, metric routing, ECHO wiring, PRISM/NEXUS behaviour, and crash recovery.

Method	Iterative	Search-call envelope	Matching rule
ZeroShot	no	1	Same task model and evaluator
APE	yes	2277–2480	Matched candidate generation and ranking
EvoPrompt-GA/DE	yes	2277–2480	Same population budget and initialization sweep
PromptBreeder	yes	2277–2480	Matched self-referential mutation loop
RIDER	yes	1780–3320	Same task model; no cross-experiment memory

Table 1: Budget and fairness protocol. Ranges count optimization/search calls before the shared one-pass evaluation on the full held-out test partition; identical iterative-baseline ranges are intentional budget matching across datasets.

Hyperparameters and checks. The default $N = 12$, $G = 8$ is a Pareto point rather than the largest tested configuration. Increasing the population to 18 adds about 0.8 points but about 50% more per-generation cost; 12 generations gives a small gain but higher overfitting risk. Windowed Thompson Sampling uses a 5-generation window, decay 0.9, $\epsilon = 0.10$ for forced exploration and $\epsilon = 0.25$ under stagnation; diversity is a near-tie retention criterion with similarity threshold 0.92, except CommonGen where 0.85 avoids rejecting semantically similar valid instructions. The implementation records dataset/model identifiers, decoding settings, cache keys, prompt hashes, and the selected prompt before test evaluation. Tests cover CommonGen reference grouping, shared BERTScore, ECHO routing, PRISM/NEXUS, and inactive-operator guards.

4 Results

Task	Models	RIDER	Best BL	Margin
GSM8K	3	**1.000**	0.956	+4.4 pp
AG_News	3	**0.967**	0.902	+6.5 pp
SQuAD 2.0	3	**0.878**	0.770	+10.9 pp
CommonGen	5	**0.850**	0.795	+5.6 pp
XSum	5	**0.659**	0.595	+6.3 pp
CodeSearchNet	4	**0.778**	0.676	+10.2 pp
Overall	23	**0.835**	0.762	+7.2 pp

Table 2: Task-level average over same-model comparisons. “Best BL” is the strongest baseline for each row before averaging. RIDER wins all 23 rows.

Table 2 and Figure 2 summarize the top-1 results. CodeSearchNet is not treated as solved: RIDER scores are 0.769–0.787, while the best baseline scores are 0.666–0.687.

All margins are absolute percentage points, not relative gains. This makes each row a paired same-model comparison against the strongest baseline and avoids ambiguity between absolute and relative improvement.

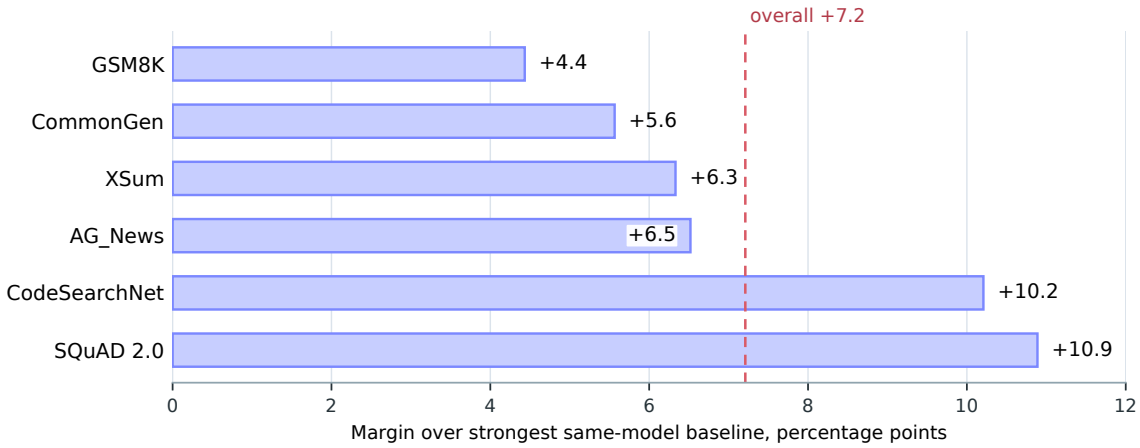

Figure 2: Task-level margins over the strongest same-model baseline. Gains are reported as absolute points. The largest margins occur on SQuAD 2.0 and CodeSearchNet, while GSM8K has little headroom because baselines are already close to perfect.

Task	Model	RIDER	Best baseline
GSM8K	Qwen / Gemini / Haiku	1.000 / 1.000 / 1.000	0.933 / 0.967 / 0.967
AG_News	Qwen / Gemini / Haiku	0.957 / 0.981 / 0.963	0.877 / 0.931 / 0.897
SQuAD 2.0	Qwen / Gemini / Haiku	0.835 / 0.926 / 0.875	0.715 / 0.828 / 0.766
CommonGen	Qwen / GPT / Gemini / Haiku / DeepSeek	0.755 / 0.924 / 0.756 / 0.920 / 0.896	0.694 / 0.873 / 0.694 / 0.873 / 0.840
XSum	Qwen / GPT / Gemini / Haiku / DeepSeek	0.403 / 0.777 / 0.556 / 0.761 / 0.796	0.332 / 0.722 / 0.474 / 0.708 / 0.742
CodeSearchNet	Qwen / GPT / Haiku / DeepSeek	0.771 / 0.769 / 0.787 / 0.785	0.667 / 0.666 / 0.687 / 0.685

Table 3: Row-level summary of the 23 comparisons, compressed by task. Each RIDER number is compared with the strongest baseline on the same model and split.

The row-level pattern is plausible. GSM8K reaches exact-match 1.000, but with a small margin because baselines are already close. Generative BERTScore rows remain below the metric ceiling; CodeSearchNet stays in 0.769–0.787, consistent with the difficulty and variability of semantic code-documentation matching. SQuAD 2.0 gains are larger because error-directed mutation often discovers prompts that handle unanswerable or partial-answer cases more explicitly.

5 Ablations and Prompt Examples

The ablation results address whether RIDER is merely a large bundle of heuristics. Replacing Windowed Thompson Sampling with uniform random operator selection reduces average fitness by 4.13 ± 0.60 percentage points ($p < 0.001$). Removing CHIMERA on generative tasks costs 2.32 ± 0.50 points; removing NEXUS costs 1.92 ± 0.40 points and increases search-evaluation calls by 35.3%; removing PRISM+Racing/F-Race costs only 0.60 ± 0.20 points but increases search-evaluation calls by 62.5%. Thus PRISM is primarily an efficiency component, whereas adaptive selection is the main quality component. Cross-experiment memory is shown in Figure 3 for completeness, but it is disabled in the fair-comparison matrix and therefore is not a hidden advantage in Table 2.

Operator-level ablation gives a similar hierarchy. Removing `eda_mutation` or `zero_order` reduces average fitness by 2.22 points; removing `contrastive_error_decomposition` costs 1.92 points. The remaining six operators are useful but replaceable, with individual removal costs from 0.20 to 0.50 points. We therefore do not claim that every operator is equally important.

These ablations explain why RIDER keeps modest operators: small-average arms can still create rare jumps or repair unique failures. The bandit keeps them available but lowers their sampling probability when they do not help, so the broad pool does not become unguided random search.

The qualitative examples in Table 4 show a repeated transformation: RIDER makes the latent scoring contract explicit. Arithmetic suppresses explanatory text; classification constrains labels; QA and code documentation specify grounding, abstention, and evidence scope. Error feedback helps because failures reveal underspecified format, abstention, reference scope, or ordering constraints.

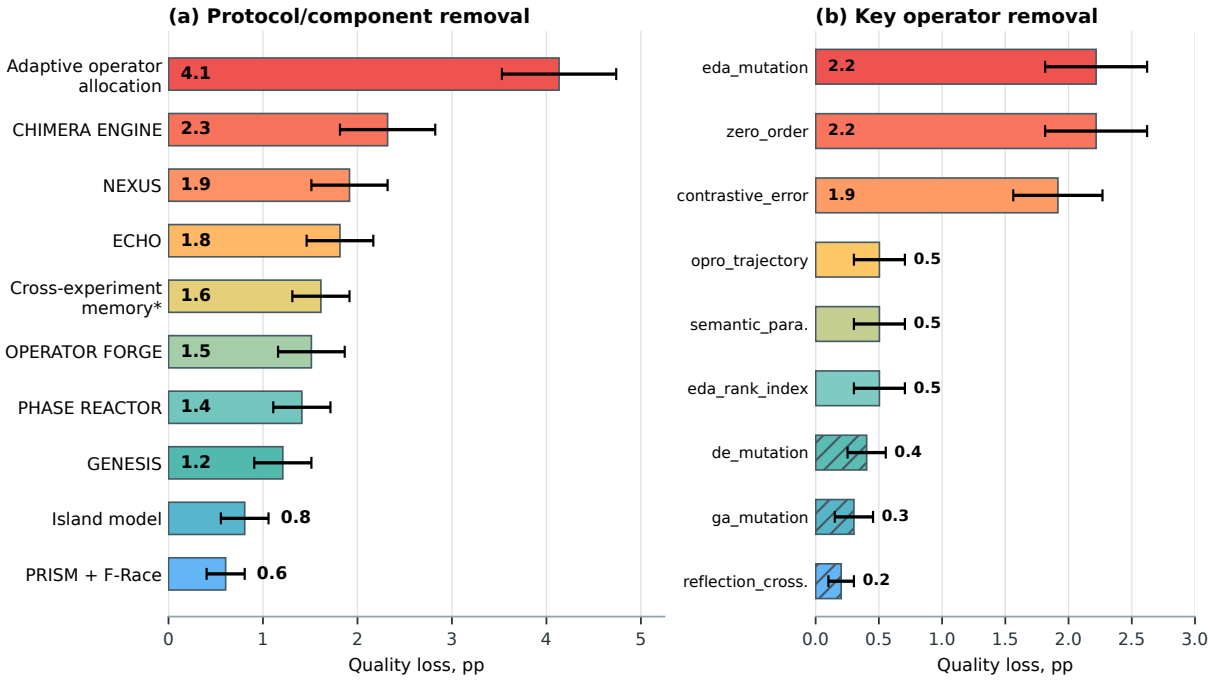

Figure 3: Ablation impact. Panel (a) shows protocol/component removal; adaptive operator allocation is the largest quality component, while PRISM mainly reduces search evaluation cost. The gray row is deployment-only memory and is disabled in the fair-comparison matrix. Panel (b) shows the operator-removal hierarchy: three operators dominate quality, while the remaining active operators have smaller but nonzero effects.

Task	Initial prompt	Final RIDER prompt fragment
GSM8K	"Solve the word problem and give the answer."	"Compute privately; return exactly `Answer: <number>` with required unit only; normalize decimals/fractions; do not round unless asked; preserve sign; no derivation, prose, commentary, extra tokens, punctuation, invented units, hidden reasoning, alternate final forms, or self-check notes after the final numeric answer; list multi-part answers in problem order and keep units attached when present."
AG_News	"Classify this news article."	"Return exactly one label: `World`, `Sports`, `Business`, or `Sci/Tech`; use event/topic evidence, ignore boilerplate and outlet names, output label only; no rationale, probabilities, hedging, mixed labels, outlet-name leakage, vote-style explanations, category aliases, metadata echoes, or rewritten headlines."
SQuAD 2.0	"Answer the question from the context."	"If unsupported, return `NO_ANSWER`; otherwise copy the shortest contiguous answer span exactly from the context, preserving wording and casing; no paraphrase, explanation, punctuation edits, answer synthesis, or evidence not literally present in the supplied context, including inferred bridging or merged spans."
XSum	"Summarize this document."	"Write one sentence preserving the main actor, action, outcome, and causal link; add no unsupported facts, title prefix, bullets, quotes, dates, editorial framing, headline, second sentence, implied facts, causal speculation, or background knowledge beyond the document, including source extrapolation, chronology repair, timeline stitching, attribution shifts, invented motives, mood claims, or source sentiment."
CodeSearch Net	"Summarize what this Python function does."	"Write one API-documentation sentence: purpose first, then visible input/output or side effect; keep identifiers and exceptions; no code, markdown, speculation, guessed intent beyond visible behaviour, tutorial expansion, complexity claims, invented usage examples, unstated assumptions, hidden dependency claims, state changes, network calls, performance guarantees, security claims, or invisible side effects."

Table 4: Examples of initial and final prompt styles. RIDER tends to transform vague requests into format-controlled, task-specific contracts that preserve grounding, output schema, abstention behaviour, and metric-relevant constraints.

Error analysis. Remaining errors fall into three groups: ambiguous SQuAD 2.0 spans; models ignoring strict output formats after long context; and metric sensitivity on generative tasks. ECHO and format-focused mutations reduce, but do not eliminate, the second issue. These errors support metric-aware routing and conservative reporting of DeBERTa BERTScore values.

6 Discussion

The main risks for prompt-optimization papers are noisy fitness, budget mismatch, metric leakage, and unfair aggregation. RIDER is evaluated with same-model baselines, a shared evaluator, top-1 reporting, explicit search budgets, and task-appropriate generative metrics. The method is heavier than a single-shot prompt generator, but is intended for reusable prompts where search cost is amortized; the fair comparison is against other automatic optimizers under comparable search budgets.

Because the six tasks have gold labels or reference texts, we use benchmark metrics rather than human preference ratings; qualitative examples and error analysis are included to expose the linguistic failure modes behind the aggregate scores.

7 Conclusion

RIDER combines evolutionary prompt search, adaptive operator selection, and error-directed feedback in a single black-box framework. In the held-out evaluation, RIDER achieves 23 wins, 0 losses, and 0 ties across six NLP benchmarks and five model families, with an average margin of 7.2 percentage points over the strongest baseline. The paper reports top-1 prompt scores, adds budget accounting, includes ablations, and provides prompt examples and code for reproduction.

References

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256.

Lichang Chen, Jiuhai Chen, Tom Goldstein, Heng Huang, and Tianyi Zhou. 2023. Instructzero: Efficient instruction optimization for black-box large language models. *arXiv preprint arXiv:2306.03082*.

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

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. // *International Conference on Learning Representations*.

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. 2019. Codesearchnet challenge: Evaluating the state of semantic code search. *arXiv preprint arXiv:1909.09436*.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. CommonGen: A constrained text generation challenge for generative commonsense reasoning. // *Findings of the Association for Computational Linguistics: EMNLP 2020*, P 1823–1840.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, P 8086–8098.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. // *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, P 1797–1807.

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

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. // *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, P 784–789.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. // *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, P 4222–4235.

William R Thompson. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294.

Xinyuan Wang, Chenxi Li, Zijian Wang, Fan Bai, Hao Luo, Jiayou Zhang, Nebojsa Jojic, Eric P Xing, and Zhiting Hu. 2024. Promptagent: Strategic planning with language models enables expert-level prompt optimization. // *International Conference on Learning Representations*.

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

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. // *Advances in Neural Information Processing Systems*, volume 28.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. // *International Conference on Learning Representations*.

Yongchao Zhou, Andrei Ioan Muresanu, Zhiwei Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. // *International Conference on Learning Representations*.