

Same Gains, Half the Cost: Selector Choice in Automatic Prompt Optimization for Financial Question Answering

Cyprian Bohojlo
Mercer
cyprianbohøjlo@gmail.com

Christopher M Stewart
University of Memphis -
Institute for Intelligent Systems
cmstwr1@memphis.edu

Abstract

Agentic pipelines assign a separately optimized system prompt to each step, so the cost of automatic prompt optimization (APO) multiplies across the pipeline. In this setting, the strategy for selecting candidate prompts cannot ignore budgetary constraints. We study the selection step of a textual gradient APO framework (Pryzant et al., 2023) in financial question answering with retrieval-augmented generation, a domain that stresses both the prompt and the reward signal. We compare the framework’s stateless Upper Confidence Bound (UCB) selector against two learned alternatives, PPO and DPO, with BEM as the shared reward. Across 135 experiments spanning three benchmarks, three budgets, and five random seeds, the selectors differ little in average accuracy gain (UCB 13.5% vs. DPO 13.4% vs. PPO 11.2%). Averaging over seeds erases these gaps and cost becomes the differentiating factor. PPO runs in under half the time of UCB (76 vs. 182 minutes per experiment) and its cost stays nearly flat as budget grows. For cost-sensitive deployments, selection has to consider efficiency in addition to accuracy, and PPO emerges as a promising low-cost selector within this APO setup. This design was shaped by an in-progress deployment at a consulting firm that assigns one system prompt to each step of an agentic data-cleaning pipeline.

1 Introduction

Production large language model (LLM) systems increasingly take the form of agentic pipelines in which each step is instructed by its own system prompt. Prompt quality drives how reliably the agent acts at every step. Optimizing those prompts automatically is attractive, but the cost of automatic prompt optimization (APO) is paid once per step and multiplies across the pipeline. Under these conditions, the strategy that an APO framework uses to select candidate prompts is less of an implemen-

tation detail and more of a budget decision. This paper studies that selection step.

Our testbed is financial question answering (QA) over corporate filings with retrieval-augmented generation (RAG). This domain also hosts the early-stage deployment that motivated the work. Answering questions using financial documents often requires extracting a figure from a table, locating a qualifying statement in the surrounding text, and performing multi-step arithmetic. Even semantically equivalent rephrasings of prompts can produce markedly different answers (Sclar et al., 2024; Zhao et al., 2024). RAG is widely adopted for factual grounding in this domain (Lithgow-Serrano et al., 2025). However, it compounds the problem by requiring the prompt to guide both passage retrieval and answer generation. Together these properties make the space of effective prompts virtually impossible to navigate manually (Pryzant et al., 2023).

APO frameworks (Zhou et al., 2023; Yang et al., 2024; Khattab et al., 2024; Pryzant et al., 2023) address the challenge of prompt sensitivity by systematically generating, evaluating, and refining candidate prompts. By design, these frameworks evaluate each candidate independently and discard information from previous evaluations. In production settings where every LLM call carries a cost, this stateless approach wastes budget re-exploring regions of the prompt space that prior evaluations have already informed. Moreover, these frameworks are built for tasks where the question arrives with its context attached. In RAG, the prompt itself determines what context is retrieved. We ask whether the selection step can be improved by learning from previous evaluations. We build on the textual gradient framework of Pryzant et al. (2023) and introduce PPO (Schulman et al., 2017) and DPO (Rafailov et al., 2023) as alternatives to its stateless Upper Confidence Bound (UCB) selector.

This paper makes three contributions. First, we

show that for agentic pipelines the choice among APO selectors is driven by cost rather than accuracy. The selectors we compare show no statistically detectable accuracy difference across seeds, so once their per-prompt cost multiplies across pipeline steps, cost decides which selector a pipeline can afford. We give the first study of learned selection policies under this framing. Second, we report a five-seed, cost-accounted comparison of stateless and learned selectors. The accuracy gaps between selectors wash out across seeds while the cost gaps persist. Third, we derive the study’s design requirements from an early-stage deployment that assigns a separate system prompt to each step of an agentic pipeline. In that setting, selection cost is the deciding criterion. Together, these give practitioners a tested basis for choosing a selector before optimization costs multiply.

2 Background and related work

APO frameworks such as APE (Zhou et al., 2023), OPRO (Yang et al., 2024), DSPy (Khatab et al., 2024), and ProTeGi (Pryzant et al., 2023) evaluate candidate prompts independently on a held-out sample with no information carried forward from previous evaluations. Among these, ProTeGi is unique in using beam search with an explicit selection mechanism (UCB), making it the most natural framework in which to study whether a learned selection policy improves performance. Reinforcement Learning (RL) has been applied to other parts of the prompt optimization pipeline, including token-level prompt generation (RLPrompt (Deng et al., 2022)), per-input prompt matching from a fixed set (DP₂O (Li et al., 2024)), and joint weight-and-prompt tuning (P²O (Lu et al., 2026)), but not to the selection step itself (Li et al., 2025).

Several of these methods target candidate generation rather than selection. DSPy (Khatab et al., 2024) compiles and tunes multi-stage programs. LLM-as-optimizer approaches such as OPRO (Yang et al., 2024) propose new prompts from optimization trajectories. Reflective or evolutionary methods search the prompt space by mutation and recombination. Whereas these vary how candidates are produced, our approach holds generation fixed and studies how candidates are selected. The two axes are complementary, so we treat these methods as orthogonal to our question rather than as baselines, and adopt ProTeGi because it exposes an explicit, swappable selection step.

A further consideration is the choice of reward signal. Most APO work relies on metrics such as accuracy or F1, which assume a single correct surface form. In generative QA, where answers are free-form text, these metrics penalize valid answers that differ only in wording (e.g., “\$2.3 billion” vs. “\$2.3B”). We use BERT matching (BEM) (Bulian et al., 2022), which provides a continuous, semantically grounded score that distinguishes genuinely wrong answers from correct answers expressed differently.

3 Methodology

In our adaptation of the textual gradient approach of Pryzant et al. (2023) to RAG-based financial QA, for each source document every page is sliced into overlapping 1,024-character chunks, then the embedding model *intfloat/e5-small-v2* by Wang et al. (2022) projects each chunk into an embedding space. The embeddings for a given filing are stored in a local *Chroma* vector store, so that retrieval cost per question is a single nearest-neighbor query. At inference time, the input to the LLM is composed of a prompt, a question, and the retrieved context, with the seed prompt given in Appendix B.

For a fixed minibatch $\mathcal{B} \subset \text{train}$, the optimizer iteratively refines a seed prompt p_0 via a three-phase loop: gradient generation, prompt editing, and local exploration. At each iteration, surviving prompt candidates are scored and the top- B are retained in a beam that serves as the memory of the optimization process. We make the selection step interchangeable, retaining the original UCB selector as the baseline and adding PPO and DPO. These three selectors are presented in more detail in Section 3.2.

3.1 Optimization Loop

The optimization loop repeats three phases per iteration:

1. **Gradient generation:** Each answer generated by the LLM is evaluated with the BEM scoring model described in Section 3.2. Using a critic prompt ∇ (see Appendix F), the critic produces statements $\{g_m\}_{m=1}^M$ that describe the failure modes of p over $\mathcal{B}$.
2. **Prompt editing:** Each g_m is a *textual gradient*, a natural-language critique of the current prompt’s failures that indicates a direction of improvement by analogy to a numerical gradient. It is injected into an editing prompt

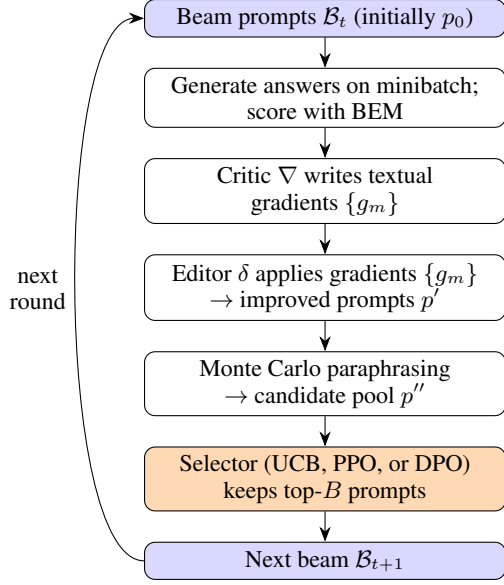


Figure 1: One iteration of the prompt optimization loop. The selector is the interchangeable component studied in this paper.

δ , asking the LLM to produce S improved prompts $\{p'_{i,s}\}_{s=1}^S$ by moving semantically in the direction opposite to the weaknesses outlined in g_m .

3. **Local exploration:** Every $p'_{i,s}$ is paraphrased M times via Monte Carlo sampling, yielding an expanded candidate pool p'' .

Throughout this process, the optimizer rewrites only the Task section of the prompt. The output-format and prediction sections of the seed prompt are held fixed, as the templates in Appendix F show. The set of best prompts from the pool p'' is then selected using one of the three available selectors. The chosen selector scores each prompt on a random subset of questions, estimates the expected reward $\hat{R}(p'')$, and keeps the top- B prompts to form the next beam $\mathcal{B}_{t+1}$.

3.2 Selection Strategies

All three selectors share a common reward signal. BEM (Bulian et al., 2022) produces a score between 0 and 1 by computing token-level similarity between generated and reference answers. Unlike exact match or F1, BEM provides a smooth gradient of quality rather than a binary correct-or-incorrect signal, which makes it more informative as a reward for iterative prompt optimization. We adopt it for its suitability as the optimization reward, but make no specific claims that it outperforms exact match or F1 for evaluation. Surface-form met-

rics assign zero reward to semantically correct answers phrased differently from the reference, which would misdirect the textual-gradient loop. The three selectors differ only in how they use this signal to decide which candidates advance. UCB uses the observed BEM scores directly through a bandit objective. PPO and DPO learn a reusable policy that predicts which candidates to retain from previous optimization experience. Appendix E gives the three selector objectives in full.

UCB (baseline): The original selector from the ProTeGi paper frames prompt selection as a multi-armed bandit problem. Each candidate prompt is an arm, and the Upper Confidence Bound formula balances exploitation of high-performing prompts with exploration of under-evaluated ones.

PPO: Proximal Policy Optimization (Schulman et al., 2017) replaces the stateless bandit with a learned policy π_θ that is updated from previous evaluations. In our setting, each action corresponds to selecting a candidate prompt. During optimization, only candidates sampled by the policy are evaluated with BEM, and the resulting score is the reward signal used to learn the policy. The learned policy assigns each candidate a probability and ranks them in preference order, retaining the top-ranked prompts for the next beam. The objective uses a clipped surrogate to maximize the expected reward while constraining policy updates.

DPO: Direct Preference Optimization (Rafailov et al., 2023) sidesteps explicit reward modeling by learning directly from pairwise preferences. During training, candidate prompts are converted into preference pairs. When two prompts are evaluated on the same minibatch, the one with the higher BEM score is the preferred prompt and the other is the dispreferred prompt. The learned policy is optimized to assign a higher probability to preferred prompts. Unlike PPO, which maximizes expected reward, DPO’s objective minimizes the preference loss, encouraging the policy to increase the likelihood of preferred prompts relative to dispreferred ones. In APO’s optimization loop, this policy assigns a preference score to a sampled subset of candidate prompts. The highest-ranked prompts are carried forward to the next beam. DPO has a simpler training objective than PPO as it requires no value function or clipping; however, this comes at the cost of relying on well-separated preference pairs.

4 Experiments

4.1 Benchmarks

We evaluate these three selectors on three financial QA benchmarks that span a range of document types, question complexities, and reasoning demands. **FinQA** (Chen et al., 2021) contains questions requiring numerical reasoning over SEC filings. Here, we use 6,251 QA pairs across 2,110 unique documents after removing exact-duplicate question-answer pairs that recur across different document contexts, keeping one instance of each. **FinDoc-RAG** (Lithgow-Serrano et al., 2025) draws over 600 QA pairs from 46 banking documents stratified across nine complexity levels, from single-document factual extraction to multi-document synthesis. **FinanceBench** (Islam et al., 2023) provides 150 publicly available question-answer-document triplets grounded in real public filings, with questions requiring domain knowledge and multi-step numerical reasoning. Together, the three benchmarks cover short- and long-context reasoning, public filings and private banking documents, and a spectrum from factual lookup to multi-step analysis.

4.2 Setup

Following standard practice, each dataset is split 80/20 into training and held-out test sets. The optimization process runs on the training set. All reported accuracy figures are evaluated on a fixed random subsample of 20% of the held-out test set, drawn once per run and held fixed for all of that run’s evaluations. We subsample rather than evaluate on the full split to bound evaluation cost across the 135-experiment grid. The Limitations section notes the variance this introduces. A round is one iteration of the optimization loop of Section 3.1, and T denotes the final round. At each round $t = 0, \dots, T$, the chosen selector scores the candidate pool using BEM on a random minibatch. Except for a model ablation analysis (see below), we use OpenAI GPT-4o-mini as the underlying LLM, chosen for its low inference cost relative to larger frontier models, with retrieval handled by ChromaDB vector stores using *intfloat/e5-small-v2* embeddings. At inference time the prompt template is instantiated with the retrieved context and the question, and the resulting string is sent to the model as a single user message.

Following Pryzant et al. (2023), we vary the four most impactful hyperparameters: minibatch

size per evaluation, number of evaluation rounds, number of candidate prompts admitted per round, and top- K prompts evaluated per round. These are grouped into three budget levels, low, medium, high, with the number of optimization rounds fixed at 6 and beam size S at 4 across all conditions. This yields 3 benchmarks $\times$ 3 budgets $\times$ 3 selectors $\times$ 5 random seeds, or 135 experiments. We repeat each of the 27 configurations across five random seeds and report the mean and standard deviation across seeds. All experiments use the same initial seed prompt p_0 to ensure comparability. Our code and experiment outputs are included as supplementary material. Our primary metric is BEM accuracy: the fraction of generated answers whose BEM score exceeds an equivalence threshold, counting them as semantically correct. Following Bulian et al. (2022), we set the threshold to 0.56, the optimal value they report. Accuracy gain is the difference in BEM accuracy between the seed prompt and the best prompt in the final round’s beam, in percentage points. The accuracy is computed on a 20% subsample of each test set.

5 Results

Across all 135 experiments, every selector improves over the seed prompt on average. UCB and DPO achieve the highest average accuracy gains (13.5% and 13.4%), with PPO trailing at 11.2%. Given per-run standard deviations of roughly 10–13 percentage points, these differences are small and do not establish a clear winner at this sample size (paired t-tests and Wilcoxon tests across seeds find no significant selector difference; 95% CIs overlap; see Appendix D). As Figure 2 indicates, performance varies more by benchmark than by selector, with FinQA seeing the largest gains (17.6%), followed by FinanceBench (12.6%) and FinDoc-RAG (7.9%).

Budget effect. Average gain does not increase monotonically with budget. It rises from 13.5% at low budget to 14.7% at medium, then falls to 9.9% at high, as shown in Figure 3. The drop at high budget is largely a ceiling effect. Higher budgets raise the seed-prompt baseline itself, so the optimizer has less headroom to improve. FinDoc-RAG’s baseline, for instance, rises from 38% at low budget to 66% at high. The improvement is also uneven across runs. Some individual experiments show zero or negligible gains even at high budget, so larger budgets raise expected performance only

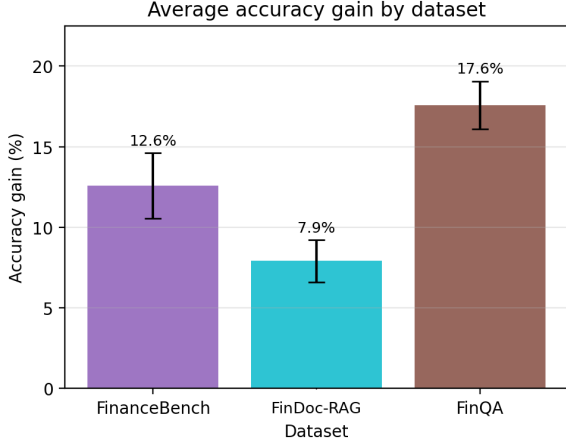


Figure 2: Average accuracy gain across all datasets

up to a point and do not guarantee improvement in every case.

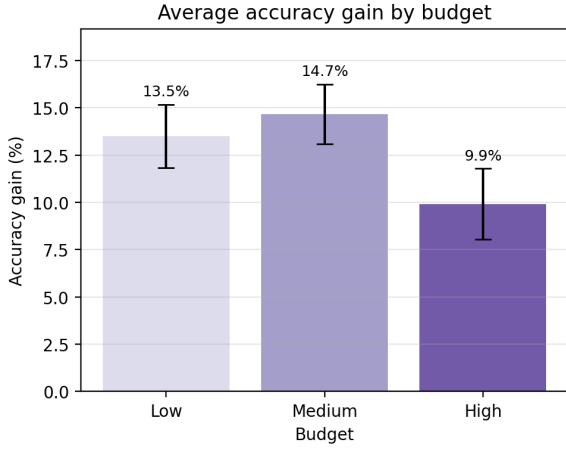


Figure 3: Average accuracy gain by budget

Efficiency. The accuracy gains above must be weighed against computational cost. As Figure 4 shows, PPO averages 76 minutes per experiment, against 182 for UCB and 150 for DPO. The gap widens sharply with budget. PPO’s runtime stays nearly flat, at 69, 76, and 82 minutes for low, medium, and high budgets, whereas UCB’s rises from 71 to 364 minutes and DPO’s from 63 to 284. UCB’s exploration-exploitation mechanism requires evaluating under-sampled candidates at every round, which becomes increasingly expensive as the candidate pool grows, and DPO’s preference comparisons scale similarly. PPO’s learned policy avoids this by concentrating evaluations on promising candidates early, giving it both the lowest and the most predictable cost.

5.1 Cross-Model Ablation

To test whether these findings are specific to GPT-4o-mini, we replicated the FinDoc-RAG experiments with Claude Haiku 4.5, running all three selectors across three seeds at the low and medium budgets. Optimization improved over the seed prompt for both models, with average gains of 14% (std 6.0) for Haiku and 10.9% (std 10.6) for GPT-4o-mini on the matched budget and seed configurations. Optimization significantly improved for both models over the seed prompt ($p < .001$ for each), and the gap between their gains was not significant ($p = .38$), so the improvement holds across both models without favoring either. Haiku optimized from lower baselines than GPT-4o-mini, yet reached comparable final accuracy. As with GPT-4o-mini, the three selectors differed little in gain under Haiku (UCB 14.7%, PPO 14.0%, DPO 13.3%). The cost pattern also held. Under Haiku at medium budget, PPO required roughly half as many API calls as UCB or DPO (2,463 vs 4,957 on average) while achieving equivalent accuracy, replicating the cost-efficiency pattern observed with GPT-4o-mini.

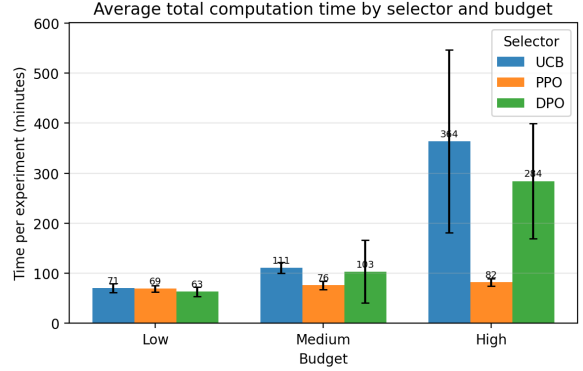


Figure 4: Average total computation time per experiment with GPT-4o-mini by selector and budget. Error bars show standard deviation across five seeds.

6 Discussion

For teams deploying prompt optimization in production, the choice of selector is driven by both accuracy and cost. UCB delivers the highest average accuracy gain at 13.5% against 11.2% for PPO, but the difference sits well within the seed-to-seed variance and is not statistically detectable. The cost difference is much clearer. UCB averages 182 minutes per experiment versus 76 for PPO, and because each optimization round scores candidate prompts via LLM inference, UCB’s exhaustive exploration translates directly into higher API spend.

Using average runtime as a proxy for that spend, PPO requires roughly 42% of UCB’s compute. A token-level cost estimate points the same way. Pricing each model call at GPT-4o-mini’s published rates, an average run costs about \$1.1 under PPO against \$2.8 under UCB, roughly 40%.¹ For a team running prompt optimization across N tasks, defaulting to PPO saves about 58% of wall-clock time at no statistically distinguishable cost in accuracy, though this calculus depends on budget, as we show next.

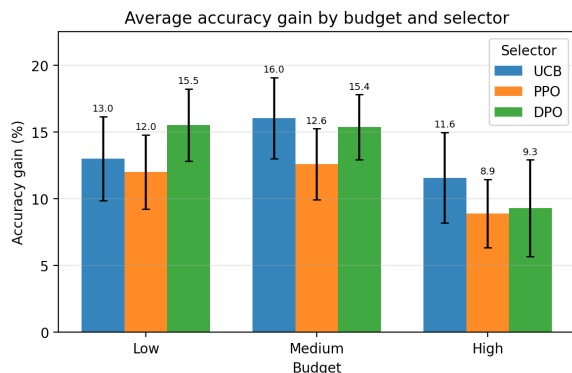


Figure 5: Average accuracy gain by budget and selector

As Figure 5 shows, the budget effect is uneven across selectors and does not simply favor more budget. Average gain peaks at medium budget for UCB and PPO and declines at high budget for all three, reflecting the ceiling effect noted above. UCB edges out PPO at every budget level, but the margins are small relative to variance, and DPO is competitive at low and medium budgets.

More broadly, these results suggest that the selection step in APO matters more for efficiency than for accuracy. With five seeds, the accuracy gaps between selectors shrink to within seed-to-seed variance. What does separate the selectors robustly is cost. PPO is roughly half as expensive as UCB and potentially more stable across budgets.

These findings suggest two further directions. The RL-based selectors currently treat each opti-

¹These figures are estimated rather than metered. We count the chat-completion calls issued by the optimizer loop, namely gradient, edit, paraphrase, candidate-scoring, and per-round test-evaluation calls, and price them at the GPT-4o-mini rates of \$0.15 and \$0.60 per million input and output tokens. BEM scoring runs locally and adds no API cost. We assume retrieved chunks of about 250 tokens, with one, three, and six chunks at low, medium, and high budget, answers of about 30 tokens, and about 800 input and 200 output tokens for the critic, editor, and paraphrase calls. The PPO-to-UCB ratio follows from the scoring call counts and is largely insensitive to these token assumptions, whereas the absolute dollar figures are approximate.

mization round independently, but incorporating cross-round state, for example by encoding the trajectory of prompt quality over time, could let the policy adapt its exploration as optimization progresses. And while this work focuses on financial QA, the selection problem is domain-agnostic, so applying these methods to other RAG-intensive verticals such as legal or medical document analysis would test whether the efficiency gains transfer to settings with different document structures and reasoning demands.

7 Design Requirements from a Deployment

The design of this study was shaped by an early-stage deployment at Mercer, a US-based consulting firm that is part of Marsh Group. Mercer is building an agentic system that autonomously cleans and pre-processes data, replacing a multi-step workflow that consultants previously carried out manually. Each step of the workflow is instructed by its own system prompt, and historical client data yields QA-style examples that serve as the training signal for optimizing those prompts. We used the deployment’s requirements to fix the design choices in our benchmark study.

Three requirements distinguish this setting from single-task APO research. First, each pipeline step needs its own optimized system prompt, so per-prompt optimization cost multiplies across the pipeline. Second, the QA-style training examples have free-form answers, so reward signals that assume a single correct surface form penalize valid outputs. This requirement motivated BEM as the reward. Third, retrieval is needed at some steps but not others. Some steps depend only on their immediate input, while a planned extension adds retrieval over publicly available financial data so the agent can fill in missing fields rather than leaving them blank. This aspect motivated studying selection inside a RAG configuration.

These requirements mean that cost predictability and accuracy are weighted equally in this study. The per-run cost estimates of Section 6 scale directly to the pipeline level. A pipeline with 20 independently optimized steps would cost roughly \$56 to optimize under UCB and roughly \$22 under PPO.

The content in this paper is provided for general information only and does not amount to professional advice.

8 Conclusion

This paper asked whether the candidate selection step of APO benefits from a learned policy. Across 135 experiments over three benchmarks, three budgets, and five seeds, no selector difference in accuracy is detectable at our sample size, and a cross-model ablation with Claude Haiku 4.5 reproduces this pattern. The “same gains” of our title should be read as this non-detection result: no selector difference is detectable at our sample size, and equivalence can be bounded only at margins of roughly three to five percentage points. What separates the selectors is cost. PPO runs at under half the price of UCB and stays nearly flat as budget grows. Within this ProTeGi-style setup, that makes PPO a promising low-cost selector for cost-sensitive pipelines where per-prompt optimization cost multiplies across steps.

Limitations

This work has several limitations. Our main study uses a single LLM: GPT-4o-mini. The cross-model ablation in Section 5.1 adds Claude Haiku 4.5, but only on FinDoc-RAG and only at the low and medium budgets, so the two-model agreement we report does not establish generalization across the full grid of experiments. Both models are closed API systems, and we hold the embedding model (intfloat/e5-small-v2) fixed throughout, so retrieval-side generalization is untested. Second, the evaluation relies on BEM as the sole reward signal, so alternative metrics could yield different rankings among the selection strategies. Our benchmarks, while diverse in structure and complexity, are all English-language and drawn from the financial domain. Additionally, while we report each configuration across five random seeds, five is a modest sample for distinguishing closely spaced selectors, and the variance is large on small test sets. FinanceBench in particular has only six held-out examples, so its per-seed standard deviations reach 20% and its results should be read as indicative rather than conclusive. Transient API failures are also scored as errors, i.e. a call that fails after retries returns an empty answer, which BEM counts as incorrect, and a failure during prompt editing yields no new candidates in that round, adding noise indistinguishable from optimization variance. Finally, per-run accuracy is computed on a 20% subsample of each test set, so seed-to-seed variance reflects both optimization stochasticity and test-subsample

composition.

References

- Jannis Bulian, Christian Buck, Wojciech Gajewski, Benjamin Börschinger, and Tal Schuster. 2022. [Tomayto, tomahto. Beyond token-level answer equivalence for question answering evaluation](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 291–305, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Zhiyu Chen, Wenhui Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P. Xing, and Zhiting Hu. 2022. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3369–3391.
- Pranab Islam, Anand Kannappan, Douwe Kiela, Rebecca Qian, Nino Scherrer, and Bertie Vidgen. 2023. FinanceBench: A new benchmark for financial question answering. *arXiv preprint arXiv:2311.11944*.
- 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)*.
- Chengzhengxu Li, Xiaoming Liu, Yichen Wang, Duyi Li, Yu Lan, and Chao Shen. 2024. Dialogue for prompting: A policy-gradient-based discrete prompt generation for few-shot learning. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI)*, pages 18481–18489.
- Wenwu Li, Xiangfeng Wang, Wenhao Li, and Bo Jin. 2025. A survey of automatic prompt engineering: An optimization perspective. *arXiv preprint arXiv:2502.11560*.
- Oscar Lithgow-Serrano, David Kletz, Vani Kanjirangat, David Adametz, Marzio Lunghi, Claudio Bonesana, Matilde Tristany Farinha, Yuntao Li, Detlef Replinger, Marco Pierbattista, Stefania Stan, and Oleg Szehr. 2025. Assessing RAG system capabilities on financial documents. In *Proceedings of the 10th Workshop on Financial Technology and Natural Language Processing (FinNLP), EMNLP 2025*, Suzhou, China.

Xinyu Lu, Kaiqi Zhang, Jinglin Yang, Boxi Cao, Yaojie Lu, Hongyu Lin, Min He, Xianpei Han, and Le Sun. 2026. P²O: Joint policy and prompt optimization. *arXiv preprint arXiv:2603.21877*.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chengguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with “gradient descent” and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7957–7968, Singapore. Association for Computational Linguistics.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models’ sensitivity to spurious features in prompt design or: How I learned to start worrying about prompt formatting. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

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)*.

Shengming Zhao, Yuheng Huang, Jiayang Song, Zhijie Wang, Chengcheng Wan, and Lei Ma. 2024. Towards understanding retrieval accuracy and prompt quality in RAG systems. *arXiv preprint arXiv:2411.19463*.

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)*.

A Hyperparameter Configuration

All experiments fix the number of optimization rounds at 6 and the beam size $\mathcal{S}$ at 4. The three budget levels vary the remaining four hyperparameters as follows:

Table 1: Hyperparameter settings by budget level.

Hyperparameter	Low	Medium	High
Minibatch size per evaluation	24	36	48
Evaluation rounds	2	4	8
Candidate prompts per round	2	4	8
Top- K prompts per round	1	3	6
Total evaluation budget	96	576	3,072

B Seed Prompt

The following prompt was used as the initial seed p_0 across all 135 experiments:

Task
You are a financial QA assistant.
Answer the question using only the information in the context.

Output format
Return your answer as free text. If it’s a number, just output the number (don’t wrap in words). Otherwise, output a concise sentence.

Prediction
Context: {context}
Question: {question}
Answer:

C Full Experimental Results

Table 2: Experimental results by dataset, budget, and selector. Values are mean $\pm$ standard deviation across five random seeds, in percentage points. Final Acc. is the BEM accuracy of the best beam prompt at the last optimization round; Gain is Final Acc. minus Base Acc.

Dataset	Budget	Selector	Base Acc.	Final Acc.	Gain
FinanceBench	low	UCB	10.0 $\pm$ 8.2	30.0 $\pm$ 6.7	20.0 $\pm$ 12.5
FinanceBench	low	PPO	10.0 $\pm$ 8.2	30.0 $\pm$ 12.5	20.0 $\pm$ 6.7
FinanceBench	low	DPO	6.7 $\pm$ 8.2	30.0 $\pm$ 12.5	23.3 $\pm$ 8.2
FinanceBench	medium	UCB	23.3 $\pm$ 8.2	33.3 $\pm$ 21.1	10.0 $\pm$ 13.3
FinanceBench	medium	PPO	23.3 $\pm$ 8.2	33.3 $\pm$ 21.1	10.0 $\pm$ 13.3
FinanceBench	medium	DPO	20.0 $\pm$ 12.5	33.3 $\pm$ 21.1	13.3 $\pm$ 12.5
FinanceBench	high	UCB	36.7 $\pm$ 19.4	40.0 $\pm$ 22.6	3.3 $\pm$ 12.5
FinanceBench	high	PPO	30.0 $\pm$ 6.7	40.0 $\pm$ 13.3	10.0 $\pm$ 8.2
FinanceBench	high	DPO	36.7 $\pm$ 12.5	40.0 $\pm$ 8.2	3.3 $\pm$ 16.3
FinDoc-RAG	low	UCB	37.6 $\pm$ 12.5	49.6 $\pm$ 5.4	12.0 $\pm$ 12.6
FinDoc-RAG	low	PPO	38.4 $\pm$ 11.8	41.6 $\pm$ 8.2	3.2 $\pm$ 10.6
FinDoc-RAG	low	DPO	37.6 $\pm$ 10.9	42.4 $\pm$ 7.0	4.8 $\pm$ 8.2
FinDoc-RAG	medium	UCB	51.2 $\pm$ 4.7	61.6 $\pm$ 4.1	10.4 $\pm$ 5.4
FinDoc-RAG	medium	PPO	48.0 $\pm$ 3.6	60.8 $\pm$ 3.9	12.8 $\pm$ 5.3
FinDoc-RAG	medium	DPO	47.2 $\pm$ 5.9	61.6 $\pm$ 8.2	14.4 $\pm$ 6.5
FinDoc-RAG	high	UCB	67.2 $\pm$ 4.7	72.0 $\pm$ 3.6	4.8 $\pm$ 4.7
FinDoc-RAG	high	PPO	65.6 $\pm$ 7.0	71.2 $\pm$ 4.7	5.6 $\pm$ 4.8
FinDoc-RAG	high	DPO	64.8 $\pm$ 7.8	68.0 $\pm$ 5.1	3.2 $\pm$ 6.4
FinQA	low	UCB	16.5 $\pm$ 1.2	23.5 $\pm$ 6.6	7.0 $\pm$ 6.4
FinQA	low	PPO	16.4 $\pm$ 1.1	29.2 $\pm$ 6.9	12.8 $\pm$ 6.8
FinQA	low	DPO	16.7 $\pm$ 0.7	35.1 $\pm$ 2.9	18.4 $\pm$ 3.1
FinQA	medium	UCB	22.1 $\pm$ 1.3	49.8 $\pm$ 1.8	27.7 $\pm$ 1.8
FinQA	medium	PPO	22.1 $\pm$ 1.4	37.0 $\pm$ 10.5	15.0 $\pm$ 9.9
FinQA	medium	DPO	22.5 $\pm$ 1.7	40.9 $\pm$ 6.2	18.4 $\pm$ 7.3
FinQA	high	UCB	23.3 $\pm$ 1.5	49.8 $\pm$ 2.4	26.6 $\pm$ 1.7
FinQA	high	PPO	22.8 $\pm$ 1.3	33.9 $\pm$ 12.7	11.1 $\pm$ 13.7
FinQA	high	DPO	22.6 $\pm$ 1.9	43.9 $\pm$ 8.4	21.3 $\pm$ 8.0

D Statistical Tests

This appendix reports the significance tests summarized in Section 5. All tests use the per-seed accuracy gains from Table 2, pairing selectors by dataset, budget, and seed ($n = 45$ paired triples).

Table 3 gives the three pairwise selector comparisons. No pairwise difference is significant at $\alpha = .05$ under either a paired t -test or a Wilcoxon signed-rank test.

Comparison	Mean diff	95% CI	t -test	Wilcoxon
UCB vs PPO	+2.4	[−1.2, 6.0]	.19	.30
UCB vs DPO	+0.1	[−3.2, 3.5]	.93	1.00
PPO vs DPO	−2.2	[−5.8, 1.3]	.22	.16

Table 3: Pairwise selector comparisons across all 45 dataset-budget-seed triples. Mean differences and CIs in percentage points; last two columns are p -values. CIs use the t critical value ($df = 44$).

The 95% confidence intervals on each selector’s overall mean gain, computed across all 45 per-seed observations, overlap substantially: UCB 13.5 [9.7, 17.3], PPO 11.2 [8.0, 14.3], DPO 13.4 [9.8, 17.0] percentage points.

We also state an equivalence bound via two one-sided tests (TOST) at $\alpha = .05$, computed from the 90% confidence intervals on the paired mean differences (UCB vs PPO [−0.6, 5.4], UCB vs DPO [−2.6, 2.9], PPO vs DPO [−5.2, 0.7]). The data establish equivalence for UCB vs DPO at margins of ± 2.9 pp or wider, but for UCB vs PPO and PPO vs DPO only at margins of roughly ± 5.4 pp and ± 5.2 pp. Differences smaller than these margins can be neither detected nor ruled out, which is why we state our finding as non-detection at this sample size rather than as demonstrated equivalence.

Individual cells do show non-trivial differences. Four of the nine dataset-budget cells have a gap above 8 percentage points between the best and worst selector. Per-cell comparisons use a paired t -test across seeds. One FinQA cell reaches significance (low budget, $p = .008$), and no other cell does ($p \geq .063$). The winning selector across the large-gap cells is not consistent, however (DPO at low budget, UCB at medium and high), which is the pattern expected from seed variance on a hard task rather than a stable selector advantage.

These tests are limited by sample size. With $n = 5$ seeds and a typical within-cell standard deviation of roughly 9 percentage points, the design has 80% power only for per-cell differences of about 19 percentage points. Smaller differences may exist but cannot be separated from seed variance at this sample size. Accordingly, we read the aggregate result as no detectable selector difference on accuracy rather than as proven equivalence.

E Selector Objectives

This appendix gives the objective for each of the three selectors compared in Section 3.2. All three score candidates with the same BEM reward and differ only in how that reward drives selection.

UCB. Each candidate prompt is treated as an arm of a multi-armed bandit. At round t the selector picks

$$p_i = \arg \max_p \left\{ Q_t(p) + c \sqrt{\frac{\log t}{N_t(p)}} \right\}, \quad (1)$$

where $Q_t(p)$ is the empirical mean BEM reward of prompt p , $N_t(p)$ is the number of times p has been evaluated, and c controls the exploration-exploitation trade-off.

PPO. The policy π_θ is trained on the clipped surrogate objective of Schulman et al. (2017),

$$L_t(\theta) = \mathbb{E}_t [L_t^{\text{CLIP}}(\theta) - c_1 L_t^{\text{VF}}(\theta) + c_2 H[\pi_\theta(\cdot | s_t)]] , \quad (2)$$

where L_t^{CLIP} is the clipped surrogate objective and c_1 and c_2 are weighting coefficients. The value-function and entropy terms are:

$$L_t^{\text{VF}}(\theta) = (V_\theta(s_t) - \hat{R}_t)^2, \\ H[\pi_\theta(\cdot | s_t)] = - \sum_a \pi_\theta(a | s_t) \log \pi_\theta(a | s_t),$$

where V_θ is the learned value function, $\hat{R}_t$ is a return estimate, and the entropy term balances the exploration-exploitation trade-off. Our policy carries no state variable across rounds, so $s_t \equiv 0$ and the learned value baseline $V_\theta(0)$ reduces to a scalar. That scalar reduces the variance of the policy-gradient estimator without biasing it. We clip the probability ratio at $\varepsilon = 0.2$ and the gradient norm at 1.0.

DPO. Candidate prompts evaluated on the same minibatch are converted into preference pairs. Write p_w for the prompt with the higher BEM score, p_l for the prompt with the lower score, and x for the minibatch on which the pair was scored. The policy minimizes

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = - \mathbb{E}_{(x, p_w, p_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(p_w | x)}{\pi_{\text{ref}}(p_w | x)} - \beta \log \frac{\pi_\theta(p_l | x)}{\pi_{\text{ref}}(p_l | x)} \right) \right] \quad (3)$$

where π_{ref} is a reference policy and β controls how far π_θ may deviate from it.

F Optimizer Prompts

The optimizer uses three fixed prompt templates, reproduced verbatim below. Braces mark runtime substitutions. In all three, the substituted prompt text is the # Task section of the candidate rather than the full prompt.

Critic ∇ . Produces the textual gradients $\{g_m\}_{m=1}^M$ of Section 3.1.

You are designing a retrieval-augmented question-answering prompt.

My current prompt is:

"{prompt}"

But this prompt gets the following examples wrong:

{error_string}

give {num_feedbacks} reasons why the prompt could have gotten these examples wrong.

Wrap each reason with <START> and <END>

Errors are sampled from the minibatch positions where BEM judges the prediction non-equivalent to the gold answer, and formatted as

Example 1

Question: "..."

Gold answer: "..."

Prediction: "..."

Editor δ . Applies one gradient to produce improved candidates.

You are designing a retrieval-augmented question-answering prompt.

My current prompt is:

"{prompt}"

But it gets the following examples wrong:

{error_str}

Based on these examples the problem with this prompt is that {feedback_str}

Based on the above information, I wrote

{steps_per_gradient} different improved prompts.

Each prompt is wrapped with <START> and <END>.

Paraphraser. Generates the Monte Carlo variants that expand the candidate pool.

Generate a variation of the following instruction while keeping the semantic meaning.

Input: {prompt_section}

Output: