Prompt Optimization Improves Robustness of Language Model Benchmarks for Medical Tasks

Asad Aali, Muhammad Ahmed Mohsin, Vasiliki Bikia, Arnav Singhvi, Suhana Bedi, Hejie Cui, Miguel Fuentes, Alyssa Unell, Yifan Mai, Michael Adam Pfeffer, Roxana Daneshjou, Sanmi Koyejo, Emily Alsentzer, Christopher Potts, Nigam Shah, Akshay S. Chaudhari

Stanford University

Abstract

While language model (LM) benchmarking frameworks such as MedHELM enable holistic evaluation across medical tasks, their leaderboards often rely on a fixed prompt per benchmark, without invoking step-by-step reasoning. Furthermore, it is known that fixed prompts may not generalize well across LMs, potentially yielding unrepresentative estimates. Unless we estimate each LM's performance ceiling, we risk underestimating performance. Prompting frameworks, such as DSPy, offer a scalable alternative to prompt engineering for alleviating this challenge. We present a framework that integrates DSPy with MedHELM, introducing structured prompting that elicits reasoning. Using four prompting methods, we evaluate four LMs across four benchmarks against MedHELM's leaderboard. We find that structured prompting: (i) outperforms the MedHELM baseline (+3.2% absolute across-benchmark average), (ii) reduces variance associated with prompt design (−2.9% absolute across-benchmark standard deviation), (iii) alters performance gaps (flips LM rankings on one benchmark), and (iv) demonstrates that LMs with *chain-of-thought* are relatively insensitive to prompt design. We conclude that scalable and automated performance ceiling estimation enables more robust medical benchmarks.

Data and Code Availability

1. DSPy Integration for HELM: github.com/stanford-crfm/helm/pull/3893
2. Prompt Optimization for HELM Benchmarks: github.com/StanfordMIMI/dspy-helm
3. We evaluate four open-source datasets: MedCalc-Bench (Khandekar et al., 2024), Medec (Abacha et al., 2024), HeadQA (Vilares and Gómez-Rodríguez, 2019), MedBullets (Medbullets, 2025).

Institutional Review Board (IRB) This research did not require IRB approval.

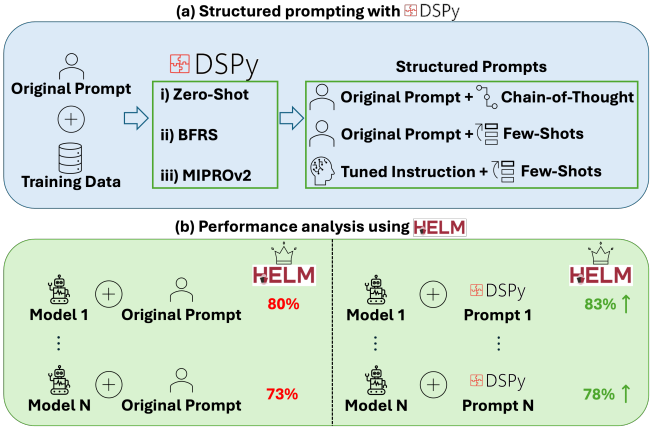

Figure 1: Pipeline describing (a) DSPy-based structured prompting methods, and (b) performance analysis of baseline prompt vs structured prompt across models on MedHELM.

1. Introduction

Language models (LMs) have rapidly advanced in text generation, spurring deployment for medical decision support, documentation, and education (Thirunavukarasu et al., 2023; Van Veen et al., 2024; Seo et al., 2024). Yet, integrating LMs into medical workflows remains challenging. LMs frequently commit errors (Aali et al., 2025) (only 39% of GPT-4 diagnoses match final diagnoses; up to 18% of Med-PaLM responses are incorrect (Wang et al., 2024; Sivarajkumar et al., 2024)). Such concerns are compounded by LMs' sensitivity to prompt design (Razavi et al., 2025), introducing variability in performance.

While medical benchmarking frameworks like MedHELM (Bedi et al., 2025) enable holistic evaluation via a comprehensive clinician-validated suite covering diverse tasks, public leaderboards typically evaluate multiple LMs under a single, fixed prompt per bench-

mark. However, fixed prompts rarely generalize well across LMs, leading to underestimated scores. Hence, broader LM adoption necessitates scalable estimation of performance ceilings, such that evaluation of LMs is more robust and useful for deployment decisions.

Prompt engineering has emerged as a practical alternative to fine-tuning. Well-designed prompts have been shown to improve performance, as demonstrated by run-time strategies like Medprompt (Nori et al., 2024) and OpenMedLM (Maharjan et al., 2024). However, these methods rely on hand-engineered prompts, demanding domain expertise and iterative experimentation, making them impractical (Wang et al., 2025).

Consequently, researchers have explored automatic prompt optimization (Bogireddy et al., 2025) by using LMs to propose and refine candidate prompts. Methods like PromptAgent (Wang et al., 2023), OPRO (Yang et al., 2023), and DRPO (Singla et al., 2024) can surpass human-engineered prompts on general tasks. Within this space, DSPy (Khattab et al., 2023) is a declarative programming framework that compiles LM programs into modules and uses prompt optimization techniques like MIPROv2 (Opsahl-Ong et al., 2024) to turn high-level specifications into effective instructions and few-shot examples. However, these methods have not been systematically evaluated in various *medical* applications, and there is limited guidance on how prompt search compares with fixed prompt baselines on medical benchmarks.

To address this gap, we integrate DSPy with MedHELM (Figure 1) and present:

1. A reproducible framework that introduces structured prompting methods within MedHELM that elicit reasoning, enabling more robust evaluation of LMs across medical tasks.

2. An evaluation of prompting methods (Zero-Shot, Bootstrap Few-Shot with Random Search, MIPROv2) against MedHELM's baseline across four LMs and four benchmarks.

3. Empirical evidence showing that reasoning-enhanced structured prompting: (i) outperforms the MedHELM baseline (+3.2% absolute across-benchmark average), (ii) reduces variance associated with prompt design (−2.9% absolute across-benchmark standard deviation), (iii) alters performance gaps (flips LM rankings on one benchmark), and (iv) demonstrates that LMs with *chain-of-thought* (CoT) are relatively insensitive to prompt design.

2. Methodology

Formally, let Φ denote a LM program with m modules. Each module i has a prompt template p_i containing a set of variables (open slots) for the instruction and K demonstration examples. Let V be the set of all such prompt variables across Φ, and let $V \to S$ denote an assignment of each variable $v \in V$ to a concrete string $s \in S$. We write $\Phi_{V \to S}$ to denote running program Φ under a particular prompt assignment. Given a dataset $D = (x, x')$ of inputs x with ground-truth x' and an evaluation metric μ that compares the program's output $\Phi(x)$ against x', the optimization maximizes μ over all instructions and demonstrations:

$$\Phi^* = \underset{V \to S}{\operatorname{argmax}} \frac{1}{|D|} \sum_{(x,x') \in D} \mu\Big(\Phi_{V \to S}(x), x'\Big). \quad (1)$$

2.1. Baseline Prompting

1. MedHELM Baseline. MedHELM leaderboard reports performance using fixed, zero-shot prompting (no CoT), which we use as a baseline for comparing against structured prompting methods.

2. Zero-Shot Predict. DSPy's Zero-Shot Predict configuration is an unoptimized non-adaptive baseline, which we instantiate with the `dspy.Predict` module. Each module's instruction prompt is fixed (taken from MedHELM) with no demonstrations (i.e. $K = 0$).

2.2. Structured Prompting

1. Zero-Shot CoT. DSPy's Zero-Shot CoT utilizes the same prompting structure as Zero-Shot Predict, but instead instantiates `dspy.ChainOfThought` to elicit step-by-step rationales.

2. BFRS. Bootstrap Few-Shot with Random Search (BFRS) leverages the idea of bootstrapping and random sampling to select the best few shots (fixed instructions): (i) Bootstrapping demonstrations: the LM program Φ is run on a subset of training inputs to gather traces. Whenever the output of $\Phi(x)$ for an example x achieves a sufficiently high score μ, the input-output pair is picked. (ii) Random few-shot search: Given candidate demonstration pools, BFRS samples sets of K demonstrations per module, inserts them into the prompts, and evaluates the program on a validation split. After trying N combinations, the program with the highest score is returned.

3. MIPROv2. MIPROv2 is an optimizer that jointly selects instructions and K few shots via: (i)

Benchmark	Input → Output	Task	Samples
MedCalc-Bench	Note → Value	Reasoning	1,000
Medec	Narrative → Errors	Classification	597
HeadQA	Question → Answer	USMLE QA	1,000
MedBullets	Question → Answer	USMLE QA	308

Table 1: Benchmarks from MedHELM evaluated in our study, spanning diverse tasks.

bootstrapping demos, (ii) grounded instruction proposals from a LM, and (iii) Bayesian search over instruction-demo pairs. Treating each configuration $\mathbf{v}$ as a hyperparameter, it learns $p(y \mid \mathbf{v})$ from trial outcomes and steers toward high-score regions. Candidates are scored on mini-batches of size B, with periodic full-dataset D evaluations of top contenders; the best full-data configuration is returned.

2.3. Benchmarks

We choose four benchmarks from MedHELM (Table 1) based on (i) public availability, and (ii) task diversity (reasoning, error detection, knowledge QA):

MedCalc-Bench. MedCalc-Bench (Khandekar et al., 2024) is a medical calculation benchmark, where the input is a patient note and a question asking for a numerical value. The evaluation metric μ is exact match for the *risk*, *severity*, and *diagnosis* categories, and a within-range correctness check for others.

Medec. Medec (Abacha et al., 2024) is an error detection and correction benchmark, where each input contains a narrative that may contain errors, and the task is to identify/correct these errors. The evaluation metric μ involves checking how accurately LMs identify whether a note contains an error (binary).

HeadQA. HeadQA (Vilares and Gómez-Rodríguez, 2019) is a collection of biomedical multiple-choice questions for testing medical knowledge, where questions cover medical knowledge and resemble medical board exams. The performance metric μ is exact match between the prediction and the correct option.

MedBullets. MedBullets (Medbullets, 2025) is a benchmark of USMLE-style medical questions with multiple-choice answers. MedBullets covers broad topics and is designed to reflect the difficulty of medical licensing exams. Like HeadQA, the primary metric μ is exact match accuracy on the correct answer.

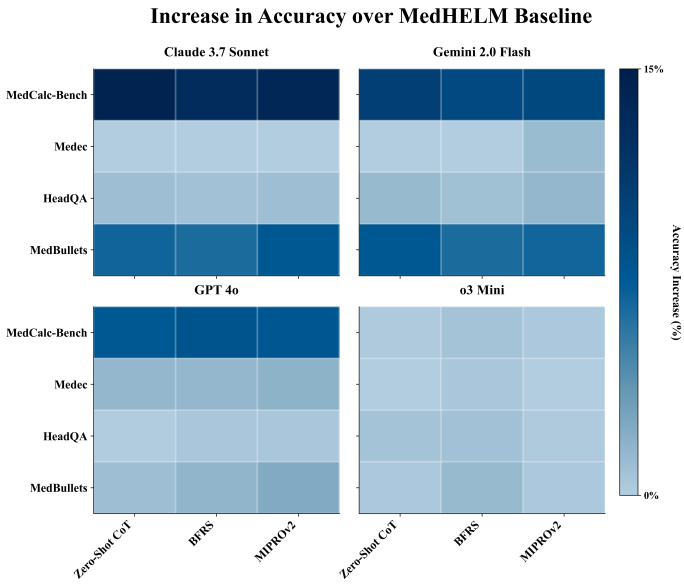

Figure 2: Heat map showing Δ of each prompting method over MedHELM's baseline (light=small, dark=large).

3. Results

We discuss the impact of structured prompting on the MedHELM leaderboard (Figure 2 and Table 3).

Improved performance over MedHELM baseline. Across LMs and benchmarks, reasoning-enhanced structured prompting methods consistently improve over the MedHELM baseline (Table 3). On average, LMs gain +3.2% in absolute accuracy. Non-reasoning LMs benefit most, while the reasoning-optimized *o3 Mini* sees smaller gains (marginal increase from 68.3% → 69.1%).

Flipped leaderboard rankings. Taking ceiling performance into account alters leaderboard rankings. On MedCalc-Bench, the ranking flips: baseline *o3 Mini* > *Claude 3.7 Sonnet* (34.0% vs. 21.0%) becomes *Claude 3.7 Sonnet* > *o3 Mini* under structured prompting (35.3% vs. 34.7%), suggesting that prompting can act as a confounder for leaderboards.

Altered inter-model performance gaps. When evaluated at ceiling performance (best score across prompting methods), models can either narrow or widen their relative performance gaps, providing a more accurate view of true capability differences. Averaging across benchmarks, the gap between the top two models (*o3 Mini* and *Claude 3.7 Sonnet*) shrinks

Benchmark	Prompting	DSPy Module	Claude 3.7 Sonnet	Gemini 2.0 Flash	GPT 4o	o3 Mini
Average	MedHELM Baseline	None	60.0%	56.6%	59.6%	68.3%
	Zero-Shot	Predict	58.2%	57.4%	57.7%	68.5%
	Zero-Shot	ChainOfThought	65.3%	**61.5%**	62.2%	68.5%
	BFRS	ChainOfThought	64.5%	60.7%	62.6%	**69.1%**
	MIPROv2	ChainOfThought	**65.5%**	61.4%	**62.8%**	68.4%
Std Dev σ	MedHELM Baseline	None	29.0%	30.0%	30.3%	24.4%
	Zero-Shot	Predict	28.3%	29.8%	30.4%	25.2%
	Zero-Shot	ChainOfThought	23.7%	26.5%	27.0%	24.5%
	BFRS	ChainOfThought	24.1%	26.7%	27.0%	24.5%
	MIPROv2	ChainOfThought	24.0%	26.8%	27.2%	24.4%

Table 2: Benchmark (accuracy) of four language models and five prompting methods, averaged across four MedHELM benchmarks. **Green** marks the "ceiling" performance for a model.

from 8.3% at baseline (68.3 vs. 60.0) to 3.6% (69.1 vs. 65.5) when evaluated at ceiling.

Reduced across-benchmark variance. Structured prompting methods reduce dispersion. Across-benchmark σ drops for *Claude 3.7 Sonnet* (29.0% $\rightarrow$ 23.7%), *Gemini 2.0 Flash* (30.0% $\rightarrow$ 26.5%), and *GPT 4o* (30.3% $\rightarrow$ 27.0%), while *o3 Mini* is unchanged (24.4%), indicating lower sensitivity.

Benchmark-dependent sensitivity. Performance gains vary substantially across benchmarks. Tasks requiring computational reasoning, such as MedCalc-Bench and MedBullets, show the largest gains. In contrast, HeadQA and Medec exhibit smaller improvements. HeadQA appears bottlenecked by ceiling effects from high baselines, while Medec likely reflects knowledge saturation.

Performance gains primarily attributed to CoT. For each model–benchmark pair, we identify a ceiling (best performance across prompting methods) to measure how far MedHELM's baseline is from the maximum achievable via prompt-only changes. Moving from MedHELM's baseline to Zero-Shot Predict yields minimal improvement. In contrast, introducing CoT reasoning results in substantial gains for Zero-Shot CoT. Interestingly, moving from Zero-Shot CoT to more sophisticated optimizers such as BFRS and MIPROv2 adds only marginal improvement, indicating that once CoT reasoning is introduced, LMs become relatively insensitive to further optimization.

Why CoT reasoning reduces sensitivity to prompt design? We demonstrate that reasoning models such as *o3 Mini*, or non-reasoning models with prompt-induced reasoning (CoT), are relatively insensitive to prompt design. Now, we provide mathematical insight into this phenomenon. With CoT, the model marginalizes over many reasoning paths; prompt optimization mostly reweights which paths are sampled. A model with parameters θ reads an input x (a benchmark item) under a prompt p, samples a reasoning path τ (a chain or tree-of-thought), and produces a final answer y. CoT/ToT factorizes the joint as $P_\theta(\tau, y \mid x, p) = P_\theta(\tau \mid x, p)\, P_\theta(y \mid x, \tau, p)$ and predicts by marginalizing over paths (self-consistency) $P_\theta(y \mid x, p) = \sum_\tau P_\theta(\tau \mid x, p)\, P_\theta(y \mid x, \tau, p)$. Because the second is a marginalization of the first, the change in the answer distribution between two prompts p and p' is upper-bounded by the change in the path distribution $\big\|P_\theta(y \mid x, p) - P_\theta(y \mid x, p')\big\|_{TV} \le \big\|P_\theta(\tau \mid x, p) - P_\theta(\tau \mid x, p')\big\|_{TV} \le \sqrt{\frac{1}{2}\, D_{KL}\big(P_\theta(\tau \mid x, p) \,\|\, P_\theta(\tau \mid x, p')\big)}$ where $\|\cdot\|_{\mathrm{TV}}$ is total-variation distance and $\mathrm{KL}(\cdot\|\cdot)$ is Kullback–Leibler divergence (Pinsker + data processing). We define the decision margin at item x under prompt p as $m(x; p) = P_\theta(y^\star \mid x, p) - \max_{y \neq y^\star} P_\theta(y \mid x, p), \quad y^\star = \arg\max_y P_\theta(y \mid x, p)$. If the left-hand side of the previous bound is $< \frac{1}{2}\, m(x; p)$, then the prediction is invariant: $\arg\max_y P_\theta(y \mid x, p') = \arg\max_y P_\theta(y \mid x, p)$. Since CoT enlarges $m(x; p)$ by averaging di-

Benchmark	Prompting	DSPy Module	Claude 3.7 Sonnet	Gemini 2.0 Flash	GPT 4o	o3 Mini
MedCalc Bench	MedHELM Baseline	None	21.0%	15.8%	18.8%	34.0%
	Zero-Shot	Predict	20.6%	17.0%	15.7%	33.4%
	Zero-Shot	ChainOfThought	**35.3%** ↑	**26.3%**	26.6%	34.2%
	BFRS	ChainOfThought	34.1%	25.2%	**27.0%**	**34.7%** ↓
	MIPROv2	ChainOfThought	34.7%	25.4%	26.8%	34.3%
Medec	MedHELM Baseline	None	**62.8%**	59.6%	58.0%	68.7%
	Zero-Shot	Predict	58.3%	59.3%	57.3%	68.3%
	Zero-Shot	ChainOfThought	61.8%	59.5%	59.5%	68.2%
	BFRS	ChainOfThought	60.5%	59.1%	59.5%	**69.2%**
	MIPROv2	ChainOfThought	62.5%	**60.8%**	**59.8%**	68.3%
HeadQA	MedHELM Baseline	None	91.2%	88.0%	90.6%	89.3%
	Zero-Shot	Predict	88.7%	88.5%	86.4%	**90.9%**
	Zero-Shot	ChainOfThought	**92.2%**	89.3%	90.7%	90.0%
	BFRS	ChainOfThought	92.0%	88.9%	**91.1%**	90.1%
	MIPROv2	ChainOfThought	**92.2%**	**89.5%**	**91.1%**	89.5%
MedBullets	MedHELM Baseline	None	64.9%	63.0%	71.1%	81.2%
	Zero-Shot	Predict	65.3%	64.9%	71.4%	81.5%
	Zero-Shot	ChainOfThought	71.8%	**70.8%**	72.1%	81.5%
	BFRS	ChainOfThought	71.4%	69.5%	72.7%	**82.5%**
	MIPROv2	ChainOfThought	**72.7%**	69.8%	**73.4%**	81.5%

Table 3: Benchmark (accuracy) of four language models across five prompting methods and four MedHELM benchmarks. **Green** marks the "ceiling" performance for a model. ↑ and ↓ indicate a one-step increase or decrease in leaderboard rank, respectively.

verse valid paths, typical prompt optimization, which mainly reweights $P_\theta(\tau \mid x, p)$, cannot flip decisions except on near-tied items, making reasoning models largely prompt-insensitive. If $\mathrm{KL}\big(P_\theta(\tau \mid x, p) \,\|\, P_\theta(\tau \mid x, p')\big) \le \kappa$ and $m(x; p) \ge 2\varepsilon$, then predictions are invariant under $p \to p'$ whenever $\sqrt{\kappa/2} < \varepsilon$.

4. Conclusion

By integrating DSPy with MedHELM, we introduce reasoning-enhanced structured prompting and empirically estimate LM performance ceilings, obtaining more representative estimates of performance. Across four benchmarks and four LMs, structured prompting: (i) outperforms the MedHELM baseline (+3.2% absolute across-benchmark average), (ii) reduces variance associated with prompt design (−2.9% absolute across-benchmark standard deviation), (iii) alters performance gaps (flips LM rankings on one benchmark), and (iv) demonstrates that LMs with CoT are relatively insensitive to prompt design. Sensitivity is heterogeneous: reasoning LMs show marginal gains, whereas some benchmarks for non-reasoning LMs benefit more. Gains are largely agnostic to the structured prompting method; the substantive effect is moving from baseline to *any* CoT variant, with much of the lift attributable to eliciting step-by-step reasoning. Together, scalable and automated performance ceiling estimation enables more robust, decision-useful medical benchmarks.

References

Asad Aali, Vasiliki Bikia, Maya Varma, Nicole Chiou, Sophie Ostmeier, Arnav Singhvi, Magdalini Paschali, Ashwin Kumar, Andrew Johnston, Karimar Amador-Martinez, et al. Medval: Toward expert-level medical text validation with language models. *arXiv preprint arXiv:2507.03152*, 2025.

Asma Ben Abacha, Wen-wai Yim, Yujuan Fu, Zhaoyi Sun, Meliha Yetisgen, Fei Xia, and Thomas Lin. Medec: A benchmark for medical error detection and correction in clinical notes. *arXiv preprint arXiv:2412.19260*, 2024.

Suhana Bedi, Hejie Cui, Miguel Fuentes, Alyssa Unell, Michael Wornow, Juan M. Banda, Nikesh Kotecha, Timothy Keyes, Yifan Mai, Mert Oez, et al. Medhelm: Holistic evaluation of large language models for medical tasks. *arXiv preprint arXiv:2505.23802*, 2025. URL https://arxiv.org/abs/2505.23802.

Sai Prasanna Teja Reddy Bogireddy, Abrar Majeedi, Viswanatha Reddy Gajjala, Zhuoyan Xu, Siddhant Rai, and Vaishnav Potlapalli. Neural at archehr-qa 2025: Agentic prompt optimization for evidence-grounded clinical question answering. *arXiv preprint arXiv:2506.10751*, 2025.

Nikhil Khandekar, Qiao Jin, Guangzhi Xiong, Soren Dunn, Serina Applebaum, Zain Anwar, Maame Sarfo-Gyamfi, Conrad Safranek, Abid Anwar, Andrew Zhang, et al. Medcalc-bench: Evaluating large language models for medical calculations. *Advances in Neural Information Processing Systems*, 37:84730–84745, 2024.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, et al. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

Jenish Maharjan, Anurag Garikipati, Navan Preet Singh, Leo Cyrus, Mayank Sharma, Madalina Ciobanu, Gina Barnes, Rahul Thapa, Qingqing Mao, and Ritankar Das. Openmedlm: prompt engineering can out-perform fine-tuning in medical question-answering with open-source large language models. *Scientific Reports*, 14(1):14156, 2024.

Medbullets. Medbullets. https://step2.medbullets.com/, 2025. Accessed 2025-08-25.

Harsha Nori, Naoto Usuyama, Nicholas King, Scott Mayer McKinney, Xavier Fernandes, Sheng Zhang, and Eric Horvitz. From medprompt to o1: Exploration of run-time strategies for medical challenge problems and beyond. *arXiv preprint arXiv:2411.03590*, 2024.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. *arXiv preprint arXiv:2406.11695*, 2024.

Amirhossein Razavi, Mina Soltangheis, Negar Arabzadeh, Sara Salamat, Morteza Zihayat, and Ebrahim Bagheri. Benchmarking prompt sensitivity in large language models. In *European Conference on Information Retrieval*, pages 303–313. Springer, 2025.

Junhyuk Seo, Dasol Choi, Taerim Kim, Won Chul Cha, Minha Kim, Haanju Yoo, Namkee Oh, YongJin Yi, Kye Hwa Lee, and Edward Choi. Evaluation framework of large language models in medical documentation: Development and usability study. *Journal of Medical Internet Research*, 26:e58329, 2024.

Somanshu Singla, Zhen Wang, Tianyang Liu, Abdullah Ashfaq, Zhiting Hu, and Eric P Xing. Dynamic rewarding with prompt optimization enables tuning-free self-alignment of language models. *arXiv preprint arXiv:2411.08733*, 2024.

Sonish Sivarajkumar, Mark Kelley, Alyssa Samolyk-Mazzanti, Shyam Visweswaran, and Yanshan Wang. An empirical evaluation of prompting strategies for large language models in zero-shot clinical natural language processing: algorithm development and validation study. *JMIR Medical Informatics*, 12: e55318, 2024.

Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. Large language models in medicine. *Nature medicine*, 29(8):1930–1940, 2023.

Dave Van Veen, Cara Van Uden, Louis Blankemeier, Jean-Benoit Delbrouck, Asad Aali, Christian Bluethgen, Anuj Pareek, Malgorzata Polacin, Eduardo Pontes Reis, Anna Seehofnerová, et al. Adapted large language models can outperform medical experts in clinical text summarization. *Nature medicine*, 30(4):1134–1142, 2024.

David Vilares and Carlos Gómez-Rodríguez. Head-qa: A healthcare dataset for complex reasoning. *arXiv preprint arXiv:1906.04701*, 2019.

Li Wang, Xi Chen, XiangWen Deng, Hao Wen, MingKe You, WeiZhi Liu, Qi Li, and Jian Li. Prompt engineering in consistency and reliability with the evidence-based guideline for llms. *NPJ digital medicine*, 7(1):41, 2024.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P Xing, and Zhiting Hu. Promptagent: Strategic planning with language models enables expert-level prompt optimization. *arXiv preprint arXiv:2310.16427*, 2023.

Zifeng Wang, Hanyin Wang, Benjamin Danek, Ying Li, Christina Mack, Luk Arbuckle, Devyani Biswal, Hoifung Poon, Yajuan Wang, Pranav Rajpurkar, et al. A perspective for adapting generalist ai to specialized medical ai applications and their challenges. *npj Digital Medicine*, 8(1):429, 2025.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2023.

Algorithm 1 Bootstrap Few-Shot with Random Search (BFRS)

Require: Seed program Φ_{seed}; train/val sets $D_{\text{tr}}, D_{\text{val}}$; threshold τ; demos per module K_i; trials R; minibatch size B.

Bootstrap: For each $(x, y) \in D_{\text{tr}}$: run Φ_{seed}; if $\mu(\Phi_{\text{seed}}(x), y) \geq \tau$, then for each module i add $\big(u_i(x), \Phi^{(i)}_{\text{seed}}(u_i(x))\big)$ to $\mathcal{B}_i$.
Search: For $r = 1{:}R$:
for $i = 1{:}m$ **do**
  Sample $S_i^{(r)} \leftarrow \text{SampleK}(\mathcal{B}_i, K_i)$
  Let $\mathbf{v}^{(r)} \leftarrow (I_1^{\text{seed}}, S_1^{(r)}, \ldots, I_m^{\text{seed}}, S_m^{(r)})$.
  Draw minibatch $\mathcal{B} \subset D_{\text{val}}$ with $|\mathcal{B}| = B$; compute $\widehat{J}_B(\mathbf{v}^{(r)})$ by (5).
Select: $\mathbf{v}^\star \in \arg\max_r \widehat{J}_B(\mathbf{v}^{(r)})$; optionally re-evaluate $J(\mathbf{v}^\star)$ on full D_{val}.
Return $\mathbf{v}^\star$ and the resulting $\Phi_{\mathbf{v}^\star}$.

Algorithm 2 MIPROv2: Joint Optimization of Instructions & Demos

Require: Train/val sets $D_{\text{tr}}, D_{\text{val}}$; candidate sizes T_i (instructions), K_i (demos per module); minibatch size B; escalation period E; TPE quantile γ; trials T.

Bootstrap demos: Build $\{\mathcal{B}_i\}_{i=1}^m$ as in (3).
Propose instructions: For each i, sample $\mathcal{I}_i = \{I_i^{(t)}\}_{t=1}^{T_i}$ from proposer LM using task/program-aware context.
Initialize history $\mathcal{H}_0 \leftarrow \varnothing$; best full-eval $(\mathbf{v}^\dagger, J^\dagger) \leftarrow (\text{seed}, 0)$.
for $t = 1{:}T$ **do**
  Fit/update TPE from $\mathcal{H}_{t-1}$ to obtain ℓ, g in (6).
  Acquire candidate:

$$\mathbf{v}^{(t)} \in \arg \max_{\mathbf{v} \in \prod_i (\mathcal{I}_i \times \mathcal{B}_i^{K_i})} \frac{\ell(\mathbf{v})}{g(\mathbf{v})}.$$

  Draw minibatch $\mathcal{B} \subset D_{\text{val}}$, $|\mathcal{B}| = B$, score $y^{(t)} = \widehat{J}_B(\mathbf{v}^{(t)})$.
  Append to history: $\mathcal{H}_t \leftarrow \mathcal{H}_{t-1} \cup \{(\mathbf{v}^{(t)}, y^{(t)})\}$.
  if $t \bmod E = 0$ **then**
    Select top-K by running mean; evaluate each on full D_{val} to get $J(\cdot)$.
    If any $J(\mathbf{v}) > J^\dagger$ then update $(\mathbf{v}^\dagger, J^\dagger) \leftarrow (\mathbf{v}, J(\mathbf{v}))$.
Return $\mathbf{v}^\dagger$ and $\Phi_{\mathbf{v}^\dagger}$.

Figure 3: Algorithms for (i) Bootstrap Few-Shot with Random Search (BFRS) and (ii) MIPROv2.

Model	API Identifier	Release	Context
Claude 3.7 Sonnet	`anthropic/claude-3-7-sonnet-20250219`	02/19/2025	200k
Gemini 2.0 Flash	`google/gemini-2.0-flash-001`	02/01/2025	1000k
GPT 4o	`openai/gpt-4o-2024-05-13`	05/13/2024	128k
o3 Mini	`openai/o3-mini-2025-01-31`	01/31/2025	200k

Figure 4: Language models evaluated in our study. The endpoint names and release dates match MedHELM's setup for comparability. All experiments were run in August 2025.

Appendix A. Experimental Setup

Implementation details. We evaluate four LMs summarized in Table 4. We seed each DSPy program with the baseline instruction for the corresponding benchmark from MedHELM's prompt. For BFRS and MIPROv2 optimizers, we follow DSPy's data separation: the demonstration pool is bootstrapped *exclusively* from the training split, while candidate prompts are evaluated on a *disjoint* held-out validation split from the original training partition; neither optimizer ever sees the MedHELM leaderboard test set. Each benchmark's loader creates a fixed train/val partition (default 90/10 with the same seed), and we cap both bootstrapped and labeled demonstrations at $K \leq 3$ per module. All final scoring is performed via MedHELM, so outputs are judged identically regardless of how they were produced. All results reflect single, deterministic runs (temperature = 0), matching MedHELM's experimental setup. For MedHELM baselines, we report MedHELM's public leaderboard scores because the setup matches ours: (i) identical LM API version, (ii) zero-shot prompting, and (iii) no CoT reasoning.

Metric calculation. To summarize overall gains, we take the mean of the three structured prompting methods (Zero-Shot CoT, BFRS, MIPROv2); for each LM, we first macro-average across benchmarks, and then average the Δ (absolute % change over baseline) across LMs. The change in variability is reported analogously using the per-model across-benchmark standard deviation, σ. For each model and prompting method, σ is computed over per-benchmark scores (equal weight per benchmark).

Appendix B. Computational Cost Analysis

While our primary goal is not to maximize absolute accuracy but to empirically estimate LM performance ceilings, we also assess the computational implications. We find that evaluation cost scales linearly with input tokens, as output lengths are capped ($<$200 tokens). DSPy optimizers such as BFRS and MIPROv2 increase input length through added demonstrations and task-specific instructions, leading to roughly 5–7$\times$ higher token usage per query across benchmarks. In contrast, Zero-Shot CoT introduces only a short reasoning header, yielding negligible overhead but capturing most of the performance gain. This makes Zero-Shot CoT the most cost-effective method, achieving substantial accuracy improvement without incurring significant evaluation cost. Optimization-phase compute, by comparison, is a one-time expense amortized over future runs and is lightweight in our configuration ($2 and $\sim$20 minutes per task). Overall, Zero-Shot CoT offers the best benefit-per-cost ratio among structured prompting methods at improving robustness.

Acknowledgments

AA is supported by NIH grant R01 HL167974 and ARPA-H contract AY2AX000045. ASC receives research support from NIH grants R01 HL167974, R01HL169345, R01 AR077604, R01 EB002524, R01 AR079431, P41 EB027060; ARPA-H contracts AY2AX000045 and 1AYSAX0000024-01; and NIH contracts 75N92020C00008 and 75N92020C00021. This research was, in part, funded by the Advanced Research Projects Agency for Health (ARPA-H). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Government.