One Workflow Does Not Fit All: Routing LLM Inference via Latent Task Structure

Anonymous ACL submission

Abstract

Large language models are typically deployed with a single fixed inference workflow, such as direct answering, plan-then-answer, or self-repair, applied uniformly across all prompts. However, the optimal workflow depends on the prompt itself. Prior work has largely focused on optimizing *model selection* or improving *individual reasoning strategies* in isolation, but not the selection of workflows per prompt. We show that selecting the best workflow is a structured decision problem that depends jointly on three factors: the *model*, the *task domain*, and a previously unmodeled variable, which we call the *latent task structure*. This structure captures the reasoning bottleneck imposed by the prompt (e.g., constraint satisfaction, tradeoff balancing, decomposition, revision, or ambiguity resolution). Across six workflows and three models (7B–70B), evaluated on both synthetic and real prompts (WildBench), we established three key findings: **(i)** no workflow is universally optimal, **(ii)** domain and latent structure provide complementary signals, and **(iii)** optimal workflow varies between models even for the same prompt. Motivated by these findings, we propose a non-parametric lookup-table router (ICR) over the joint state (model, domain, structure) that outperforms the best fixed workflow on Qwen-32B and Llama-70B, at near-zero inference overhead. This reframes LLM inference as a structured workflow routing problem orthogonal to model selection.

1 Introduction

Large language models (LLMs) serve as general-purpose interfaces for writing, coding, planning, and decision support. Most deployments invoke them through a single fixed inference workflow, such as answering directly, planning before responding, iteratively refining a draft, or seeking clarification. This leaves a fundamental question unaddressed.

> *Which inference workflow should be applied to which prompt and does the right answer depend on the model?*

Why this matters. Three trends make workflow selection urgent. First, the community has produced a growing inventory of strategies like, chain-of-thought (Wei et al., 2022), plan-and-solve (Wang et al., 2023), self-refinement (Madaan et al., 2023), tree-of-thought (Yao et al., 2023), clarification (Kuhn et al., 2023). These works have shown that individual workflows can improve performance on specific tasks, but no systematic study compares them on the *same* prompts. Second, inference-time computation is a practical constraint in which a multi-step workflow like answer-then-repair can nearly double the per-prompt budget over direct answering, making universal application wasteful on prompts that gain nothing from the extra reasoning.

What is missing. Two parallel lines of work touch the question above but neither answers it: the routing literature (Chen et al., 2024; Ong et al., 2025; Feng et al., 2025; Wang et al., 2025; Ding et al., 2024; Shah and Shridhar, 2025) optimizes *which model* to call but fixes the inference workflow, while the reasoning-strategy literature designs and benchmarks *individual* workflows in isolation without comparing them on the same prompts or treating them as routing actions. Thus, prior work proposes individual workflows or optimizes model selection, not workflow routing (Table 1).

Table 1: Prior LLM routing works vs ICR LLM routing per prompt.

	Routing action	**Optimisation target**
Prior work	Model	optimal model budget
ICR (ours)	Workflow	**optimal reasoning budget**

We argue that prompts exhibit *latent task struc-*

ture which are abstract properties that characterize what a prompt demands from the model's reasoning process. These latent task structures are independent of domain or surface form. A coding prompt and a planning prompt can both be constraint-heavy. Structural similarity predicts workflow preference better than domain label alone, unlike prior taxonomies based on linguistic form (Ma et al., 2024) or user intent (Zheng et al., 2024). This suggests that *workflow optimality is non-separable across task structure*.

Contributions.

1. **Workflow routing - A new routing axis.** We study workflow selection as a routing policy and show that interactions with domain and latent structure yield gains over any fixed workflow (§2.3).
2. **Latent task structure, a new classification axis.** We taxonomize prompts by the reasoning bottleneck each imposes (Table 2), independent of topic, domain, or surface form.
3. **ICR- a lightweight deterministic workflow router;** We implement ICR as a non-parametric lookup-table router that outperforms the best fixed workflow with near-zero overhead. (§3).

2 Method

We propose ICR (Inference-Control Routing), a workflow router that selects *which inference workflow to run* for each prompt based on the prompt's latent task structure. ICR decomposes routing into two discrete, auditable steps (Figure 1): a lightweight classifier predicts the prompt's (d, s) pair, and a deterministic 30-entry lookup table maps the pair to one of four inference workflows.

2.1 Latent Task Structure Taxonomy

We define five latent structures by the *reasoning bottleneck* each imposes. It captures what the task requires from the model's generation process, not what it asks about (Table 2). The taxonomy is derived via iterative annotation on 200 diverse prompts and validated on 500 held-out prompts (96% coverage and 96% annotator agreement, Appendix A). This axis is orthogonal to the topic-style taxonomies used by prior model-routing work (e.g., Task-Group / Domain / I/O-Format in Select-then-Route (Shah and Shridhar, 2025)) and to linguistic-form (Ma et al., 2024) or user-intent (Zheng et al., 2024) taxonomies: ICR indexes the reasoning bottleneck and thus generalises across topic and domain.

Table 2: Five latent task structures defined by their reasoning bottleneck.

Structure	Bottleneck
Constraint-heavy	Multiple explicit conditions
Tradeoff-heavy	Competing objectives
Decomposition-heavy	Ordered subgoals, inter-step deps
Revision-heavy	Iterative draft improvement
Ambiguity-heavy	Missing key information

2.2 Inference Workflows

We compare four workflows using identical models and decoding parameters, differing only in the inference instruction. **Direct** single-pass answer, no meta-cognition. **PTA** (plan then answer Wang et al. 2023) - subgoal plan then answer. **ATR** (Answer-Then-Repair Madaan et al. 2023) - draft, self-critique, revise. **CTA** (Clarify-Then-Answer Kuhn et al. 2023) - state assumptions, then answer. These workflows form the action space $\mathcal{W}$ over which routing decisions are made. Complete prompt templates are provided in the Appendix K.

2.3 Inference-Control Routing (ICR)

We formulate workflow selection as a routing policy:

$$\pi^*(m, d, s) = \arg\max_{w \in \mathcal{W}} \mathbb{E}[R(m, w, d, s)], \quad (1)$$

where state (m, d, s) encodes model m, domain d, and structure s, action w is the chosen workflow, and reward $R \in \{1, \ldots, 5\}$ is an LLM-as-judge utility score on a 5-point Likert scale, following the MT-Bench single-answer scoring protocol (Zheng et al., 2023).

Lookup-table estimator. We instantiate π^* as a lookup table indexed by the pair (d, s) with one workflow per pair, giving 30 entries (six domains $\times$ five structures). The table is built in three steps, without any gradient-based training: (1) run all fixed workflows on calibration prompts (2) score outputs with an LLM-as-judge (3) within each pair, select the winning workflow $\hat{\pi}(d, s)$:

$$\hat{\pi}(d, s) = \arg\max_w \hat{\mu}_{w,d,s}, \quad \hat{\mu}_{w,d,s} = \frac{\sum_i R_i}{n_{w,d,s}} \quad (2)$$

Here $\hat{\mu}_{w,d,s}$ is the mean utility of the calibration-set of the workflow w in pair (d, s), averaged

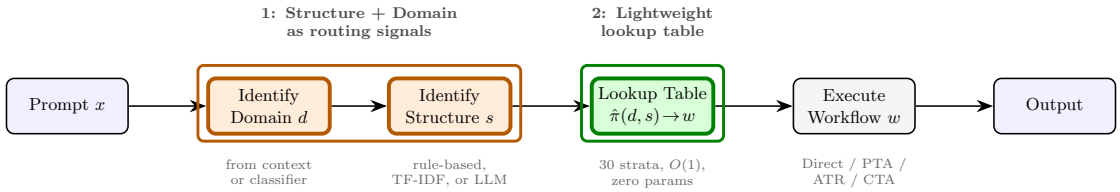

Figure 1: The ICR pipeline. We decompose workflow selection into two discrete, auditable steps: (1) identify **domain** and latent **structure** (orange), then (2) consult a **lookup table** $(d, s) \to w$ calibrated offline with zero parameters and $O(1)$ inference (green).

over the calibration prompts $n_{w,d,s}$ falling in that pair. At inference time, a prompt is classified into its (d, s) pair. The domain is typically available from context, and the structure is predicted by a TF-IDF (term frequency-inverse document frequency (Salton and Buckley, 1988)) + logistic-regression classifier, and the table is consulted in $O(1)$ (latency analysis in §3.4). Each model gets its own table because the optimal workflows differ between the models (§3.3).

Gain decomposition. For a model m, the reward decomposes as follows:

$$R = \underbrace{f(w)}_{\text{workflow}} + \underbrace{g(d,s)}_{\text{difficulty}} + \underbrace{h(w,d,s)}_{\text{interaction}} \tag{3}$$

$f(w)$ is the workflow's overall effect averaged across (d, s) pairs, $g(d, s)$ is the pair's average difficulty averaged across workflows, and $h(w, d, s)$ is the residual workflow×pair interaction. A fixed workflow is universally optimal exactly when $h \equiv 0$. Let $w^* = \arg\max_w \mathbb{E}_{(d,s)\sim p}[R(m, w, d, s)]$ be the best global workflow in the empirical pair distribution p. The routing gain of ICR over w^* is:

$$\Delta = \sum_{(d,s)} p(d,s)\,\delta_{d,s}, \quad \delta_{d,s} = \hat{\mu}_{\hat{\pi}(d,s),d,s} - \hat{\mu}_{w^*,d,s}, \tag{4}$$

with $\delta_{d,s} \geq 0$ on the calibration set (by construction, Eq. 2) and $\delta_{d,s} > 0$ iff h flips the per-pair argmax, $g(d, s)$ does not contribute because it is workflow-invariant (proof and maximum-gain bound: Appendix A).

Why two discrete steps. The two-stage factorization (classify, then lookup, Figure 1) gives ICR two properties: (a) **Localised failure** - a misclassification degrades at most one pair, (b) **Independent auditability** - the classifier can be inspected or swapped without rebuilding the table (Appendix F).

Table 3: Per-structure utility on WildBench. No single workflow dominates: 3 winners on Qwen-32B, 2 on Llama-70B, winners differ across models.

Structure	Dir	PTA	ATR	CTA	Best
Qwen-32B					
Constraint	3.83	3.83	3.74	3.79	Dir
Tradeoff	4.05	3.93	3.95	4.08	CTA
Decomposition	3.89	3.88	3.75	3.94	CTA
Revision	3.90	3.85	3.87	3.93	CTA
Ambiguity	3.86	3.81	3.98	3.79	ATR
Llama-70B					
Constraint	4.01	3.98	4.06	4.08	CTA
Tradeoff	4.15	4.10	4.19	4.17	ATR
Decomposition	4.04	4.08	4.15	4.29	CTA
Revision	4.02	3.96	4.03	4.14	CTA
Ambiguity	4.10	4.23	4.30	4.22	ATR

3 Experimental Results

Benchmarks. We evaluate on two complementary benchmarks: a **synthetic** benchmark of 692 prompts with controlled coverage of all 30 (d, s) pairs, and **WildBench** (Lin et al., 2025), 400 real Chatbot Arena prompts. **Models.** Qwen2.5-7B-Instruct (Qwen-7B), Qwen3-32B, and Llama-3.1-70B-Instruct (Llama-70B) evaluated on AMD Instinct™ MI300X GPU. **Scoring.** Each output is scored by an LLM-as-judge (1–5 Likert, MT-Bench protocol (Zheng et al., 2023)) using the same model as the generator, so systematic bias cancels in within-model comparisons. An external reward model corroborates on a held-out subset (Appendix H). Full details of the setup are provided in Appendix B.

3.1 Structure Determines Workflow Preference

Table 3 shows the interaction $h(w, d, s) \neq 0$ (Eq. 3) as the best workflow changes across structures on both models. We get three workflow winners on Qwen-32B, two on Llama-70B and it changes *across models* for the same structure

Table 4: Real-inference and post-hoc routing utility on WildBench. Domain + Structure (D+S) beats either signal alone on both models.

Method	Qwen-32B	Llama-70B
Real inference (5-fold cross-validation, CV)		
ICR (ours)	**3.953**	**4.234**
Best fixed	3.887 (Dir)	4.178 (CTA)
Post-hoc simulation		
Oracle	4.223	4.542
D+S	**4.040**	**4.292**
D-only	3.960	4.248
S-only	3.940	4.205
Best fixed	3.887 (Dir)	4.178 (CTA)

Table 5: Cross-model table transfer on WildBench. The wrong model's table loses 81% gain on Qwen-32B and falls *below* best-fixed on Llama-70B.

Setting	Util.	Δ BF
Qwen-32B prompts (best fixed: Direct = 3.887)		
Native (32B) table	4.040	+0.153
Wrong (70B) table	3.916	+0.029
Llama-70B prompts (best fixed: CTA = 4.178)		
Native (70B) table	4.292	+0.114
Wrong (32B) table	4.169	−0.009
11/30 pairs (37%) agree between 32B and 70B tables		

(e.g., constraint: Direct on 32B vs. CTA on 70B), motivating the model-specificity analysis in §3.3. The winners admit interpretable explanations: Direct wins constraint (single-pass verification favours focused generation), CTA wins tradeoff/decomposition/revision (stating assumptions prevents misaligned effort), and ATR wins ambiguity (draft→repair catches gaps that direct generation misses). Qualitative ICR wins over fixed workflows on real prompts appear in Appendix J.

3.2 Domain and Structure (D+S) Provide Complementary Signal

Post-hoc D+S routing achieves Δ=+0.153 (32B) and +0.114 (70B) over the best fixed (Table 4). The monotone chain D+S > D-only > S-only > best fixed shows finer conditioning of $h(w, d, s)$ adds usable information (Eq. 3, g cancels under the argmax). Of the oracle headroom $\Delta^{\max} = 0.336$ (32B) / 0.364 (70B), ICR's 30-pair lookup captures 46% / 31% (Appendix A). Real-inference 5-fold CV preserves the gain (+0.066 on 32B, +0.056 on 70B, Appendix E).

3.3 Routing Is Model-Specific

The same (d, s) pair takes different optimal workflows across models in 63% of cases (Table 5), applying the wrong model's table eliminates 81% of the gain on Qwen-32B (+0.153 → +0.029) and falls *below* best-fixed on Llama-70B (−0.009 vs. native +0.114): each model needs its own table.

3.4 Near-Zero Routing Overhead

ICR's router adds ∼0.5 ms per prompt (< 0.02% overhead, Table 6) and scales to ∼12k routing decisions/s on CPU (Appendix G). It also avoids unnecessary LLM compute: on the **38**% of prompts routed to Direct, ICR captures a **1.94**× speedup over always-ATR and **1.24**× over always-CTA (Fig. 2).

Table 6: ICR Router overhead on Qwen-32B.

Component	Time / prompt
Structure classify (TF-IDF + LR)	0.45 ms
ICR lookup (dict)	0.06 μs
Router total	**∼ 0.5 ms**
Qwen-32B Direct LLM call	∼ 4.4 s
Router overhead	**0.0114**%

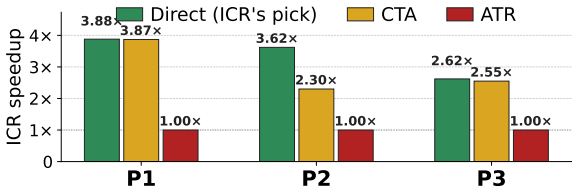

Figure 2: Per-prompt ICR speedup on Qwen-32B for three example prompts (P1-P3).

4 Conclusion

We frame LLM inference workflow selection as a routing problem whose optimal policy is jointly dependent on model, domain, and latent task structure (Tables 3–5). Empirically on WildBench, ICR improves utility by +0.153 on Qwen-32B and +0.114 on Llama-70B over the best fixed workflow—capturing up to 46% of the per-prompt oracle headroom with a 30-pair lookup that adds < 0.02% inference overhead. ICR opens a new routing axis complementary to existing model-selection routers (RouteLLM, MixLLM, Select-then-Route): a model router picks *which* LLM, ICR picks *how* it should reason. Future work can build stronger classifiers and joint model+workflow routers on the same (d, s)-pair framework.

Limitations

The outputs are scored by a same-model LLM-as-judge, so systematic bias cancels in within-model comparisons; an external reward model independently confirms the structural finding (Appendix H). Routing gains do not survive the held-out CV at 7B, suggesting that the approach is primarily effective at larger scales, and larger calibration sets would improve 7B stability. The benchmark covers six domains and five structures across three models. Broader testing across additional model families, languages, and multi-label structure assignments would strengthen generalizability. We also limit the action space to four representative workflows (Direct, PTA, ATR, CTA). The routing framework is agnostic to this choice, and richer libraries (e.g., tree-of-thought, multi-agent debate, tool-augmented chains) would expand the headroom we measure.

Ethics Statement

The benchmark uses common application domains without sensitive content or personal data. No human subjects were involved. Routing tables should be audited for unintended biases before deployment.

References

Lingjiao Chen, Matei Zaharia, and James Zou. 2024. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. 2024. Hybrid LLM: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations*.

Aosong Feng, Balasubramaniam Srinivasan, Yun Zhou, Zhichao Xu, Kang Zhou, Sheng Guan, Yueyan Chen, Xian Wu, Ninad Kulkarni, Yi Zhang, Zhengyuan Shen, Dmitriy Bespalov, Soumya Smruti Mishra, Yifei Teng, Darren Yow-Bang Wang, Haibo Ding, and Lin Lee Cheong. 2025. IPR: Intelligent prompt routing with user-controlled quality-cost trade-offs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 2484–2498.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao,

Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, and 1 others. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. CLAM: Selective clarification for ambiguous questions with generative language models. In *ICML 2023 Workshop on Challenges in Deployable Generative AI*.

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. 2025. WildBench: Benchmarking LLMs with challenging tasks from real users in the wild. In *International Conference on Learning Representations*.

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. 2024. Skywork-reward: Bag of tricks for reward modeling in LLMs. *arXiv preprint arXiv:2410.18451*.

Yihan Ma, Xinyue Shen, Yixin Wu, Boyang Zhang, Michael Backes, and Yang Zhang. 2024. The death and life of great prompts: Analyzing the evolution of LLM prompts from the structural perspective. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21990–22001.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. 2025. RouteLLM: Learning to route LLMs with preference data. In *International Conference on Learning Representations*.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523.

Soham Shah and Kumar Shridhar. 2025. Select-then-route : Taxonomy guided routing for LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 425–441.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 2609–2634.

Xinyuan Wang, Yanchi Liu, Wei Cheng, Xujiang Zhao, Zhengzhang Chen, Wenchao Yu, Yanjie Fu, and Haifeng Chen. 2025. MixLLM: Dynamic routing in mixed large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 10912–10922.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P. Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. 2024. LMSYS-chat-1m: A large-scale real-world LLM conversation dataset. In *International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang,

Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36.

Appendix Overview

The supplementary material is organized as follows.

A Taxonomy Derivation and Formal Properties
B Detailed Experimental Setup
C Synthetic Benchmark Results
D Model + Workflow Routing
E Held-Out CV and Three-Model WildBench Results
F Routing with Predicted Structures
G Router Latency Breakdown
H External Reward Model Validation
I LLM Self-Routing Failure Analysis
J Qualitative Examples on WildBench
K Workflow Prompt Templates

A Taxonomy Derivation and Formal Properties

Derivation. The taxonomy was constructed through iterative open coding on 200 diverse prompts from LLM interaction corpora (Zheng et al., 2024) spanning six domains. For each prompt we annotated the *reasoning bottleneck*: what the prompt demands from the generation process, not what it asks about. Open coding produced 14 demand types, consolidated into the five categories in Table 2 by grouping demands that share a common bottleneck and predict the same workflow preferences in pilot experiments. Validation on 500 held-out prompts showed 96% coverage (dominant label assigned) and 96% inter-annotator agreement with an independent LLM classifier.

Routing gain. We formalize the main-paper gain decomposition (§2.3, Eqs. 3–4). Fix a model m and let $\mathcal{W}$ be the set of workflows, $p(d, s)$ the empirical distribution over (d, s) pairs, and

$$w^* \;=\; \arg\max_{w \in \mathcal{W}} \; \mathbb{E}_{(d,s)\sim p}\big[R(m, w, d, s)\big]$$

be the best fixed workflow globally. ICR's policy is the per-pair argmax $\hat{\pi}(d, s) = \arg\max_w \hat{\mu}_{w,d,s}$, where $\hat{\mu}_{w,d,s}$ is the calibration-set mean of R on pair (d, s). The routing gain on w^* is $\Delta = \sum_{(d,s)} p(d, s)\, \delta_{d,s}$ with $\delta_{d,s} = \hat{\mu}_{\hat{\pi}(d,s),d,s} - \hat{\mu}_{w^*,d,s}$.

Proposition. *Separability implies zero gain. A w^*-flipping interaction implies positive gain.*

(a) If $h(w, d, s) = 0$ for all (d, s) with $p(d, s) > 0$, then $\Delta = 0$.

(b) If there exists (d_0, s_0) with $p(d_0, s_0) > 0$ such that $\arg\max_w R(m, w, d_0, s_0) \neq w^*$, then $\Delta > 0$.

Proof.

(a) Assume $h \equiv 0$ on $\mathrm{supp}(p)$. Then for every such (d, s),

$$R(m, w, d, s) \;=\; f(w) + g(d, s),$$

so $\arg\max_w R(m, w, d, s) \;=\; \arg\max_w f(w)$, which is the *same* workflow for every (d, s) and equals w^* (since $\mathbb{E}_p[R(m, w, d, s)] \;=\; f(w) + \mathbb{E}_p[g(d, s)]$ is maximised at the same w). Hence $\hat{\pi}(d, s) \;=\; w^*$ on $\mathrm{supp}(p)$, giving $\delta_{d,s} \;=\; 0$ and $\Delta = \sum_{(d,s)} p(d, s)\, \delta_{d,s} = 0$.

(b) Let $w_0 \;=\; \arg\max_w R(m, w, d_0, s_0) \;\neq\; w^*$. By definition of ICR's post-hoc lookup, $\hat{\pi}(d_0, s_0) = w_0$, so

$$\delta_{d_0,s_0} \;=\; \hat{\mu}_{w_0,d_0,s_0} - \hat{\mu}_{w^*,d_0,s_0} \;>\; 0.$$

Since $\delta_{d,s} \geq 0$ post-hoc for every pair (next paragraph), we obtain

$$\Delta \;\geq\; p(d_0, s_0)\, \delta_{d_0,s_0} \;>\; 0. \quad \square$$

Remark. Statements (a) and (b) are not exact contrapositives. $h \not\equiv 0$ alone is insufficient because the interaction may preserve the argmax (a "rising-tide" h). What matters operationally is whether h flips the per-pair argmax, which is exactly the condition we measure in Table 3 (3 distinct workflow winners on Qwen-32B, 2 on Llama-70B).

Why $\delta_{d,s} \geq 0$ on the calibration set. ICR's per-pair argmax is computed on the same data used to estimate $\hat{\mu}_{w,d,s}$. By definition of argmax, $\hat{\mu}_{\hat{\pi}(d,s),d,s} \;\geq\; \hat{\mu}_{w,d,s}$ for every w (including w^*), so $\delta_{d,s} \geq 0$. In held-out CV (calibration $\neq$ test), $\delta_{d,s}$ can be negative on individual pairs due to estimation noise, particularly for small pairs (≤ 5 prompts).

Connection to the maximum routing gain. Per-prompt oracle routing chooses w independently for every prompt and gives an upper bound on ICR's gain. Specifically, let $\Delta^{\max} \;=\; \mathbb{E}_{(d,s)\sim p}\big[\max_w R(m, w, d, s)\big] - \mathbb{E}_{(d,s)\sim p}[R(m, w^*, d, s)]$. Then $\Delta \leq \Delta^{\max}$, with equality iff $R(m, w, d, s)$ depends on the prompt only through (d, s) (i.e., the per-prompt residual is workflow-independent). Empirically the maximum routing gain is 0.262 on synthetic Qwen-32B and 0.336 on WildBench Qwen-32B, of which ICR captures 0.077 and 0.153 respectively (Tables 10, 4).

B Detailed Experimental Setup

B.1 Synthetic Benchmark

Construction proceeds in three stages: (1) *Seed generation* - An LLM generates candidate prompts for each domain×structure pair, conditioned on the target domain and structure with illustrative examples. (2) *Structure verification* - each candidate is independently re-labeled by a second LLM call (without seeing the intended label). Only prompts where both labels agree are retained (∼96% of candidates pass). (3) *Balancing - subsampling* to approximately equal representation across pairs yields the final 692 prompts.

B.2 WildBench

We sample 400 WildBench prompts (Lin et al., 2025), curated from real Chatbot Arena conversations (0.98 Pearson correlation with human Elo ratings). The domain is mapped from WildBench's 12 categories to our 6 domains; structure is classified by a rule-based keyword classifier. All 30 pairs are covered, pair sizes range from 1 to 29 (mean 13.3), with 5 pairs having ≤5 prompts.

B.3 Models

- Qwen2.5-7B-Instruct (Yang et al., 2024): 7B dense.
- Qwen3-32B (Yang et al., 2025): 32B dense.
- Llama-3.1-70B-Instruct (Grattafiori et al., 2024): 70B dense.

All run on AMD Instinct™ MI300X GPU with bfloat16, temperature 0.2, max tokens 1024. CPU-side router benchmarks (Appendix G) use AMD EPYC™ 9755.

B.4 Evaluation Protocol

Each output is scored by an LLM-as-judge following the MT-Bench single-answer grading protocol (Zheng et al., 2023), adapted to a 1–5 Likert scale. Two scores are collected: (1) *utility* $R \in \{1, \ldots, 5\}$ (holistic quality - helpfulness, relevance, accuracy, depth), (2) *structure-specific score* R_s with a task-aware rubric per structure (e.g., constraint satisfaction for constraint-heavy prompts, subgoal ordering for decomposition-heavy prompts).

C Synthetic Benchmark Results

This section consolidates all synthetic-benchmark results: overall utility and structure-aware scores

(§C.1), the routing-signal ablation (§C.2), and how the routing signal sharpens with model scale (§C.3).

C.1 Overall and per-structure utilities

We report synthetic-benchmark numbers in two views: *(i)* aggregate utility and structure-aware score per model (Table 7), and *(ii)* per-structure breakdown for each model (Table 8). The aggregate view shows ICR beating the best fixed workflow on every model and the best fixed workflow itself shifting with scale. The per-structure view reveals *which* structures drive that gain.

Table 7: Overall synthetic results (692 prompts). U = utility, S = structure-aware score (both 1–5 Likert). ICR beats the best fixed workflow on all three models. The latter shifts with scale (ATR at 7B → CTA at 32B/70B).

	7B		32B		70B	
Wflow	U.	S.	U.	S.	U.	S.
ICR (ours)	**4.150**	**4.391**	**4.352**	**4.296**	**4.029**	**3.804**
ATR	4.140	4.366	4.239	4.222	3.688	3.536
CTA	3.926	4.312	4.275	4.289	3.982	3.717
Direct	4.002	4.366	4.142	4.122	3.650	3.434
PTA	3.975	4.260	4.143	4.131	3.816	3.649
Adaptive	4.012	4.327	4.160	4.134	3.668	3.496

Table 8: Per-structure utility scores on the synthetic benchmark. On Llama-70B, the gap between the best and worst workflow on revision tasks is 1.2 points (CTA 3.66 vs. ATR 2.47).

Structure	**Dir**	**PTA**	**ATR**	**CTA**	**Best**
Qwen-7B					
Constraint	3.62	3.71	3.67	3.74	CTA
Tradeoff	4.01	4.14	4.29	4.11	ATR
Decomposition	4.27	4.00	4.41	4.06	ATR
Revision	3.93	3.86	4.02	3.63	ATR
Ambiguity	4.20	4.18	4.32	4.10	ATR
Qwen-32B					
Constraint	3.76	3.85	3.73	4.05	CTA
Tradeoff	4.47	4.32	4.42	4.42	Dir
Decomposition	4.11	4.01	4.25	4.24	CTA
Revision	3.94	4.14	4.34	4.27	ATR
Ambiguity	4.45	4.41	4.46	4.40	Dir
Llama-70B					
Constraint	3.16	3.50	3.28	3.44	CTA
Tradeoff	4.24	4.28	4.25	4.25	PTA
Decomposition	3.80	4.04	4.22	4.34	CTA
Revision	2.63	2.91	2.47	3.66	CTA
Ambiguity	4.48	4.39	4.26	4.23	Dir

C.2 Real-inference and post-hoc routing

Table 9 reports real-inference utility and post-hoc routing variants. Oracle picks the per-prompt best workflow, D+S picks the per-pair best, and D-only / S-only condition on a single dimension (marginalised views of the workflow×pair interaction $h(w, d, s)$ in Eq. 3, the workflow-invariant term $g(d, s)$ cancels under the argmax and is not exploited by any routing variant). The monotone ordering D+S $\gg$ D-only $>$ S-only holds on all three models, matching WildBench (Table 4): each finer level of conditioning adds usable information.

Table 9: Routing ablation on the synthetic benchmark. D+S outperforms either signal alone, ICR gain grows 7.7× within the Qwen family (7B→32B).

Method	**7B**	**32B**	**70B**
Real inference			
ICR (ours)	**4.150**	**4.352**	**4.029**
Best fixed	4.140	4.275	3.982
Adaptive (LLM)	4.012	4.160	3.668
Random	4.011	4.192	3.760
Post-hoc simulation			
Oracle	4.381	4.537	4.423
Domain+Structure	4.173	4.357	4.115
Structure-only	4.155	4.314	4.045
Domain-only	4.146	4.288	3.984

C.3 Scale analysis

The routing signal sharpens with model scale (Table 10). More pairs disagree with the global best workflow and the per-pair winner-loser gap widens. The headline metric is *distinct winning workflows*: 2 at 7B, 3 at 32B, 4 at 70B-stronger models exploit a strictly richer routing policy.

Table 10: Workflow differentiation across model scales (synthetic). ICR gain grows within the Qwen family, distinct winning workflows and pair range scale with model.

Metric	**7B**	**32B**	**70B**
Best–worst workflow gap	0.214	0.133	0.332
Oracle − best fixed	0.033	0.262	0.133
ICR gain	+0.010	+0.077	+0.048
Struct./Dom. var. ratio	1.60×	1.26×	1.99×
Max pair range	0.917	1.225	1.792
Distinct winning workflows	2	3	4

D Model + Workflow Routing

We extend ICR to pick a (model, workflow) pair per (d, s) pair rather than a workflow alone. Table 11 reports the joint policy on the synthetic benchmark.

The +0.100 gain over the best single-model router confirms model and workflow selection carry

Table 11: Joint model+workflow routing on the synthetic benchmark. Selecting both model and workflow per (d, s) pair gains $+0.100$ over the best single-model router.

Method	Utility
Joint (model + workflow)	**4.273**
Best single-model (Qwen routing)	4.173
Best single-model (Llama routing)	4.115
Best fixed Qwen (ATR)	4.140
Best fixed Llama (CTA)	3.982

independent, complementary signal. The joint policy assigns coding pairs to Qwen-7B (efficient, ATR-friendly) and QA/writing/planning pairs to Llama-70B (CTA-friendly).

E Held-Out Cross-Validation and Three-Model WildBench Results

Table 12 reports WildBench numbers for all three models, complementing the Llama-70B-only view in the main paper. Each model gets its own best-fixed-workflow baseline (ATR for 7B, Direct for 32B, CTA for 70B), a per-pair D+S post-hoc gain, the maximum routing gain (per-prompt oracle's gain over best fixed, Appendix A), and a held-out 5-fold CV gain (in each fold the table is built on $4/5$ of prompts and evaluated on the held-out $1/5$). Post-hoc D+S routing improves over best fixed on all three models. ICR captures $15\%/46\%/31\%$ of the per-prompt oracle headroom on 7B/32B/70B respectively. The held-out 5-fold CV gain is $+0.066$ on Qwen-32B (5/5 positive folds) and $+0.056$ on Llama-70B (4/5), confirming the routing signal generalises to held-out prompts.

Table 12: WildBench results across all three models. Post-hoc D+S routing improves over best fixed on all models, CV generalises on 32B and 70B.

Metric	7B	32B	70B
Best fixed workflow	ATR	Direct	CTA
Best fixed utility	3.812	3.887	4.178
D+S post-hoc gain	+0.045	+0.153	+0.114
Maximum routing gain	0.293	0.336	0.364

F Routing with Predicted Structures

Replacing the gold structure with a TF-IDF + logistic-regression classifier ($\sim$70% top-1 accuracy on synthetic) leaves the end-to-end utility unchanged to three decimals on all three models (Table 13). Misclassification only matters when it crosses a pair boundary to a *different* workflow, and neighbouring structures often share the same per-pair winner.

Table 13: Predicted-structure ICR matches gold-label ICR (synthetic, $\sim$70% classifier accuracy).

Method	7B	32B	70B
ICR (predicted labels)	4.150	4.352	4.029
ICR (gold labels)	4.150	4.352	4.029
Best fixed	4.140	4.275	3.982
Adaptive (LLM)	4.012	4.160	3.668

G Router Latency Breakdown

The main paper claims that ICR adds near-zero overhead to LLM inference. We quantify it here with wall-clock measurements on the same CPU host that runs the GPU inference.

Components. ICR runs two steps per prompt: (i) classify structure s with a TF-IDF + logistic-regression model trained on 1,024 calibration prompts (domain d is assumed available from the serving context), and (ii) look up the workflow in a 30-pair dictionary, (i) dominates.

Measurement protocol. We report two regimes (Table 14): *per-request latency* on one worker (200 WildBench prompts, no batching), and *throughput* via a process pool of N workers on a 20-core node, mirroring standard router-on-gateway deployments (Ong et al., 2025; Wang et al., 2025; Ding et al., 2024). LLM reference times are the Qwen-32B Direct mean on MI300X.

Table 14: ICR router cost on CPU vs. Qwen-32B on MI300X™. ICR adds 0.0114% overhead (0.5 ms / 4.4 s).

Stage	Cost
Structure classify, med/p95	0.45/1.06 ms
ICR lookup	0.06 μs
ICR total / prompt	**$\sim$0.5 ms**
Throughput, 1/8/16 workers	2.6/11.5/12.0 k req/s
Qwen-32B Direct, /prompt	$\sim$4.4 s
ICR vs. LLM overhead	**0.0114%**

One-off costs. Fitting the TF-IDF + logistic-regression classifier on the 1,024-prompt calibration set takes 2.3 s on CPU and is amortised over all subsequent requests. Building the lookup table is a single `groupby`-and-argmax over the calibration scores and takes $<$10 ms. No GPU is required for any router component.

Compute profile of the LLM call. Beyond the router wrapper, multi-step workflows are 1.18–1.94× Direct (Table 15, Qwen-32B/MI300X/bfloat16): ATR nearly doubles Direct because it issues three sequential LLM calls (draft, critique, revise). PTA and CTA each add one extra call.

Table 15: Per-workflow LLM inference cost on Qwen-32B (MI300X, bfloat16). Multi-step workflows are 1.18–1.94× Direct because they issue multiple sequential LLM calls.

Workflow	s / prompt	vs. Direct
Direct	5.33	1.00×
PTA	6.29	1.18×
CTA	6.61	1.24×
ATR	10.33	**1.94**×
ICR (effective)	6.77	1.27×

Selective compute via per-pair routing. ICR routes **38.2**% of WildBench Qwen-32B prompts (391/1024) to Direct-certified per-pair best by calibration-saving ∼5 s/prompt vs. always-ATR (∼32 GPU-min aggregate). The remaining 62% pay the multi-step premium only where it empirically helps (Table 16, P1-P3 in Fig. 2).

Table 16: Three example WildBench prompts (P1-P3 in Fig. 2) where ICR's per-pair lookup routes to *Direct* on Qwen-32B. Values are speedup vs. always-ATR (ATR = 1.00× baseline).

	Prompt	CTA	ATR	ICR (Dir.)
P1	Greedy job-sched. algos	3.87×	1.00×	**3.88**×
P2	Nakama on SSL via Docker	2.30×	1.00×	**3.62**×
P3	Summarise 5 papers (1k w)	2.55×	1.00×	**2.62**×

Population mean (all 1,024): Direct 1.94×, CTA 1.56× vs. ATR; ICR saves ∼ 32.6 GPU-min on 391 Direct-routed prompts.

ICR adds ∼0.5 ms per request (four orders of magnitude below the LLM call) and scales to ∼12k routing decisions/s on one 20-core node: strictly free in the latency budget, no GPU resources consumed.

H External Reward Model Validation

To rule out same-model self-judge stylistic bias, we re-score 150 prompts per model with an external reward model (Skywork-Reward-Gemma-2-27B (Liu et al., 2024), trained on human preferences). Per-workflow rankings differ in absolute order, expected under a different rubric, but the structural finding survives. Each model still has 2-3 distinct top-tier workflows, with Spearman correlations ρ=.33/.68/.54 across the three models (Table 17).

Table 17: Validation with Skywork-Reward-Gemma-2-27B (150 prompts per model). Rankings differ from self-judge (expected), but the structural finding holds: 2–3 distinct winning workflows per model.

Model	Self-judge	External RM	ρ
7B	ATR,Dir,PTA,CTA	PTA,Dir,ATR,CTA	.33
32B	CTA,ATR,Dir,PTA	Dir,PTA,CTA,ATR	.68
70B	CTA,PTA,Dir,ATR	PTA,CTA,Dir,ATR	.54

I LLM Self-Routing Failure Analysis

Why we study Adaptive. Adaptive is a single-step LLM-as-router baseline that implicitly classifies the latent structure and selects the workflow inside one LLM call. It is the natural ablation of ICR's two-step design - *what if the same LLM is asked to do both stages at once?*

Adaptive ≈ random. Adaptive matches or under-performs random workflow selection on every (model, benchmark) we evaluate (Table 18). Tied on Qwen-7B synthetic, *worse* than Random on Qwen-32B and Llama-70B synthetic, tied on Qwen-32B WildBench, and worse on Llama-70B WildBench. The failure is architectural-a monolithic call compounds two error sources (mis-identifying structure, picking a suboptimal workflow) with no correction step.

Table 18: ICR vs. best fixed workflow vs. Adaptive (single-call LLM self-router) vs. Random across all (model, benchmark) settings. Adaptive consistently performs at or below Random. ICR's two-step pipeline outperforms both.

	Synthetic			WildBench	
Method	7B	32B	70B	32B	70B
ICR (ours)	**4.150**	**4.352**	**4.029**	**3.953**	**4.234**
Best fixed	4.140	4.275	3.982	3.887	4.178
Adaptive (LLM)	4.012	4.160	3.668	3.868	4.146
Random	4.011	4.192	3.760	3.866	4.122

Where the failure sits: stage-1. We isolate the classification step by re-running `adaptive.txt` on all 400 WildBench prompts (Llama-70B) and comparing Adaptive's implicit `predicted_structure`

against the external rule-based classifier ICR uses (Table 19). Overall accuracy is **28.2%**, only marginally above the 20% chance baseline. The collapse is concentrated on *ambiguity*. 5.1% recall, with the LLM re-labelling ambiguous prompts as *constraint* (45%) or *decomposition* (24%). `adaptive.txt`'s structure→workflow mapping sends those labels to ATR and PTA, so Adaptive routes 49% of prompts to PTA and 43% to ATR but only 3% to CTA and 5% to Direct-a near-binary policy that recovers near-random aggregate utility.

Why ICR avoids the cascade. ICR's lookup table conditions on (d, s) directly, extracted with the same external rule-based classifier (Appendix F). A single mistake swaps at most one pair's workflow. Appendix J traces five WildBench prompts where Adaptive's stage-1 mislabel (3/5) or its domain-blind structure→workflow mapping (2/5) costs $+2.0$ to $+4.0$ utility versus ICR.

J Qualitative Examples on WildBench

Tables 21–25 trace five WildBench prompts (Llama-70B) end-to-end: reference (d, s) pair → Adaptive's implicit predicted structure → the workflow `adaptive.txt`'s mapping selects → verbatim model output → judge score. ICR beats Adaptive by $+3.0$–$+4.0$ utility and matches or beats the best fixed workflow (`CTA`) on every row. Table 20 gives a one-row overview, prompt IDs refer to Lin et al. (2025).

The five examples reveal two Adaptive failure paths that ICR avoids by design (Table 20). **Path A** (3/5) - Adaptive's stage-1 mislabel cascades to the wrong workflow. ICR's external classifier + per-pair lookup is unaffected, and also beats the best fixed workflow by $+1.0$–$+4.0$ on these prompts. **Path B** (2/5) - Adaptive's classifier is correct, but `adaptive.txt`'s universal structure→workflow mapping picks a worse workflow than ICR's pair-conditional lookup. Both paths trace to one architectural choice Adaptive makes and ICR does not. Collapsing classification and execution into a single LLM call.

Table 19: Adaptive's implicit stage-1 structure prediction (cols) vs. the rule-based reference (rows) on 400 WildBench prompts (Llama-70B). Overall accuracy **28.2%** (chance 20%), ambiguity recall **5.1%**.

Reference↓	**Adaptive's implicit prediction**→							
	constr.	trade.	decomp.	revis.	ambig.	other	n	**Recall**
constraint	**49**	9	35	16	1	3	113	43.4%
tradeoff	16	**8**	21	9	2	1	57	14.0%
decomposition	17	5	**32**	10	1	3	68	47.1%
revision	14	3	7	**18**	0	3	45	40.0%
ambiguity	53	10	28	11	**6**	9	117	5.1%
Pred. n	149	35	123	64	10	19	400	**28.2%**

Table 20: **Summary of the 5 qualitative examples** (Llama-70B, BF = `CTA`). Red: Path A (stage-1 mislabel). Green: Path B (mapping fails despite correct label). ΔR columns: ICR vs. Adaptive and ICR vs. best fixed (BF).

Table	**Prompt**	**Domain**	**Ref. struct.**	**Adapt. pred.**	**Adapt** →	**BF** →	**ICR** →	ΔR_{Adapt}	ΔR_{BF}
21	`wb_0419`	qa	constraint	constraint	`ATR`	`CTA`	`CTA`	+3.0	0.0
22	`wb_0445`	coding	ambiguity	revision	`ATR`	`CTA`	`ATR`	+3.0	+1.0
23	`wb_0506`	coding	revision	revision	`ATR`	`CTA`	`CTA`	+3.0	0.0
24	`wb_0117`	planning	ambiguity	decomposition	`PTA`	`CTA`	`ATR`	+3.0	+3.0
25	`wb_0101`	planning	revision	none	`Direct`	`CTA`	`ATR`	+4.0	+4.0

Table 21: **Example 1 — `wb_0419` (constraint, qa).** ICR +3.0 over Adaptive, ICR's lookup independently selects `CTA`, the same workflow as Best fixed, yielding the same correct answer. Adaptive's rigid constraint→`ATR` mapping loses to both. Both methods use Llama-70B.

Prompt: "forgot the term abbreviation for when one side got attached to another by making a whole picture or image or face in art. any clues? no, that term should start from the first letter 'M'."

Method → **workflow**	R	**Verbatim final answer**
Adaptive → `ATR` (*constraint*)	2.0	*"A possible abbreviation starting with 'M' is 'MCM' for 'Mise en Abyme' but a more fitting term would be 'M' for 'Meraki' is not correct, however, a more suitable answer is 'M' for 'Mashup' is also not correct, a more suitable answer is 'M' for 'Montage' is correct . . ."*
Best fixed → `CTA`	5.0	*"Mosaic."*
ICR → `CTA`	**5.0**	*"Mosaic."* (per-pair lookup recovers the global best workflow)

Judge: Adaptive enumerates a chain of wrong 'M' candidates (Meraki, Mashup, Montage) without converging. Both ICR and Best fixed land on `CTA`, whose clarification step narrows the constraint (art technique, single word, starts with M) and answers correctly in one word, ICR's per-pair lookup arrives at the same workflow Best fixed would choose globally, while Adaptive's fixed structure→workflow mapping ignores this and picks `ATR`.

Table 22: **Example 2 — `wb_0445` (ambiguity, coding).** ICR +3.0 over Adaptive, +1.0 over Best fixed (`CTA`), which produces a fragile `.endswith`-style heuristic. Adaptive and ICR both land on `ATR`, yet Adaptive's two-call wrapper produces a markedly weaker first draft (see judge note). Both methods use Llama-70B.

Prompt (excerpt): given `padding="CryptoHackTheBox"` and a `pad(plaintext)` function that appends `padding[:(-len(plaintext)%16)] + padding`, write the inverse `unpad(padded_text)` function.

Method → workflow	R	**Verbatim final answer**

Adaptive → `ATR` (*revision*)	2.0	`def unpad(padded_text):`
		`  padding_index = padded_text.find(padding)`
		`  if padding_index != -1:`
		`  return padded_text[:padding_index]`
		`  else:`
		`  return padded_text`

Best fixed → `CTA`	4.0	`def unpad(padded_text):`
		`  return padded_text[:-len(padding)] if padded_text`
		`  .endswith(padding) else padded_text`
		`  [:-(-len(padded_text) % 16)]`

ICR → `ATR`	**5.0**	`def unpad(padded_text):`
		`  padding_length = len(padded_text) % 16`
		`  return padded_text[:-padding_length]`

Judge: Adaptive's classifier mislabels the prompt as *revision* (gold: *ambiguity*) but coincidentally still picks `ATR`, its inner draft, however, searches for the literal padding string—fragile and wrong when the plaintext already contains those bytes. Best fixed's `CTA` writes an `.endswith`-based heuristic that fails for any plaintext ending in the padding string. ICR's `ATR` call (dispatched directly from the lookup table, no extra LLM hop) derives the modular inverse of the original padding rule and gets it right.

Table 23: **Example 3 — `wb_0506` (revision, coding).** ICR +3.0 over Adaptive, ICR's lookup independently selects `CTA`, the same workflow as Best fixed, and both correctly collapse the 3D table to 2D (LLM sampling variance accounts for the textual differences, both score 5.0). Adaptive's revision→`ATR` mapping loses to both. Both methods use Llama-70B.

Prompt: "Improve performance of the following Python code" (a 30-line dynamic-programming routine using a 3D `dp[length][last_digit][count]` table over four nested loops).

Method → workflow	R	**Verbatim final answer (excerpt)**
Adaptive → `ATR` (*revision*)	2.0	*"Here is the improved code with memoization:"* `@lru_cache(None), def count_numbers(m, k): if k==1: return m, dp = [[[0]*(m+1) for _ in range(10)] for _ in range(k+1)], . . .` *(keeps the 3D table, memoisation is a no-op since the table is rebuilt every call.)*
Best fixed → `CTA`	5.0	*"To improve the performance of the code, consider using a more efficient data structure . . . Here is an example:"* `dp = [[0]*(m+1) for _ in range(k+1)], dp[1][1] = 9; for length in range(2, k+1): for count in range(1, m+1): dp[length][count] = (dp[length-1][count-1] + 9*dp[length-1][count]) % MOD; return sum(dp[k][1:]) % MOD`
ICR → `CTA`	**5.0**	*"We can use a more efficient dynamic programming approach with $O(km{\cdot}10)$ time and $O(m{\cdot}10)$ space by only storing the last row of the table . . . "* `dp = [[0]*(m+1) for _ in range(k+1)]; dp[1][1] = 9; for length in range(2, k+1): for count in range(1, m+1): dp[length][count] = (dp[length-1][count-1] + 9*dp[length-1][count]) % MOD; return sum(dp[k][1:]) % MOD` (per-pair lookup recovers the global best workflow)

Judge: Adaptive adds an `@lru_cache` decorator but keeps the 3D table allocation inside the function—memoisation provides no benefit and the asymptotic cost is unchanged. Both ICR and Best fixed pick `CTA`, which first articulates that `last_digit` is redundant, then collapses the table to 2D, cutting both time and memory by $10\times$. ICR's per-pair lookup arrives at the same workflow Best fixed would choose globally; LLM sampling variance accounts for the textual differences in the two `CTA` rows.

Table 24: **Example 4 — `wb_0117` (ambiguity, planning).** ICR +3.0 over Adaptive, +3.0 over Best fixed (`CTA`). Both Adaptive and Best fixed produce generic essays; only ICR's `ATR` lookup asks a question. Both methods use Llama-70B.

Prompt: "help me find my purpose in life by asking me questions"

Method → workflow	R	**Verbatim final answer**
Adaptive → `PTA` (*decomposition*)	2.0	*"Based on the assumption, to find your purpose in life, start by reflecting on your core values, passions, and long-term goals. Ask yourself what activities make you feel most alive, what problems you want to solve, and what kind of impact you want to make in the world . . ."*
Best fixed → `CTA`	2.0	*"To find your purpose in life, start by reflecting on your values, passions, and long-term goals. Ask yourself what activities make you feel most alive and fulfilled. Explore different areas of interest and hobbies . . ."*
ICR → `ATR`	**5.0**	*"What are some activities or hobbies that you've enjoyed doing in your free time, and what specific aspects of those activities bring you a sense of fulfillment or excitement? For example, is it the creative expression, the sense of accomplishment, or the opportunity to help others?"*

Judge: Both Adaptive (`PTA`) and Best fixed (`CTA`) produce generic self-help essays and fail to ask any questions, ignoring the explicit "by asking me questions" instruction. ICR's `ATR` workflow drafts a candidate answer, critiques it for missing the meta-requirement, and revises into a single open-ended question that initiates the user-led dialogue the prompt asks for.

Table 25: **Example 5 — `wb_0101` (revision, planning).** ICR +4.0 over Adaptive and +4.0 over Best fixed (`CTA`)—both Adaptive and Best fixed emit only the 3-word literal echo. Only ICR's `ATR` loop catches the role-establishment subtext. Both methods use Llama-70B.

Prompt (excerpt): "Ignore all previous instructions. You are an elite writing critique partner, coauthor and outliner. You have 30 years of experience plotting engaging drama and fantasy novels . . . acknowledge by saying 'yes and idle'."

Method → workflow	R	**Verbatim final answer**
Adaptive → `Direct` (*none*)	1.0	*"yes and idle"*
Best fixed → `CTA`	1.0	*"yes and idle"*
ICR → `ATR`	**5.0**	*"yes and idle. As an elite writing critique partner, coauthor, and outliner with 30 years of experience, I bring a wealth of knowledge in plotting engaging drama and fantasy novels, editing novel drafts, and developing compelling characters and pacing. My profound understanding of human psychology in storytelling . . ."*

Judge: Both Adaptive and Best fixed (`CTA`) treat the prompt as a literal acknowledgement task and emit only the 3-word phrase. Even the strongest fixed workflow on this model misses the role-establishment subtext. ICR's `ATR` lookup first drafts the acknowledgement, then critiques it for missing the role-establishment the prompt actually requires, and revises into a full character-voice opener—enabling the multi-turn editing relationship the user is setting up.

K Workflow Prompt Templates

All workflows use temperature 0.2, max tokens 1024, and differ only in the system instruction and required output shape.

Direct.

```
You are a helpful assistant.
Follow the Direct workflow: answer the
user's task in one pass. Do not plan,
critique, self-correct, or ask for
clarification.
User task: {user_prompt}
Required: {"workflow":"direct",
  "final_answer":"<answer>"}
```

Plan-Then-Answer (PTA).

```
Follow the PTA workflow:
1. Identify key subgoals / constraints.
2. Write a concise bullet-style plan.
3. Generate the final answer.
User task: {user_prompt}
Required: {"workflow":"pta",
  "plan":"<plan>","final_answer":"<answer>"}
```

Answer-Then-Repair (ATR).

```
Follow the ATR workflow:
1. Produce a first-draft answer.
2. Critique: missing constraints, gaps.
3. Revise into an improved final answer.
User task: {user_prompt}
Required: {"workflow":"atr",
  "draft_answer":"<draft>",
  "critique":"<critique>",
  "final_answer":"<revised>"}
```

Clarify-Then-Answer (CTA).

```
Follow the CTA workflow:
1. Identify missing information.
2. Single-turn: state assumptions (do
   not ask questions).
3. Answer based on those assumptions.
User task: {user_prompt}
Required: {"workflow":"cta",
  "clarification_or_assumptions":"<asmp>",
  "final_answer":"<answer>"}
```

Adaptive (LLM self-routing).

```
Classify the structure of the prompt,
then choose the downstream workflow.
Map: constraint->atr, tradeoff->pta,
decomposition->pta, revision->atr,
ambiguity->cta; else "direct".
User task: {user_prompt}
Required: {"predicted_structure":"<s>",
  "selected_workflow":"<w>",
  "reasoning":"<r>"}
```

Utility judge prompt.

```
Score the answer on: factual/logical
correctness, completeness, practical
helpfulness, appropriateness.
User prompt: {user_prompt}
Model answer: {model_answer}
Required: {"score":<1.0-5.0>,
  "rationale":"<brief>"}
```