IC-Cache: Efficient Large Language Model Serving via In-context Caching

Authors: Yifan Yu, Yu Gan , Nikhil Sarda, Lillian Tsai, Jiaming Shen,
Yanqi Zhou, Arvind Krishnamurthy, Fan Lai, Henry M. Levy, David E. Culler

Presented by **Sen Han**
2025–12-16

LLM inference High Cost

- **The drive for high-quality outputs has led to the deployment of models with hundreds of billions of parameters**

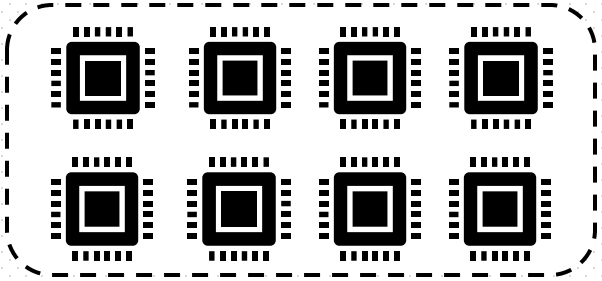
deepseek

Deepseek-R1 : 671B

Meta

Llama-3.1 405B

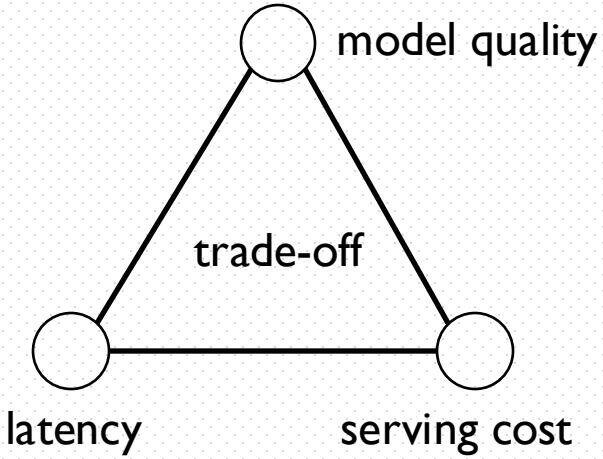
high latency for users



soaring operational costs for service providers

Recent advances in LLM serving systems

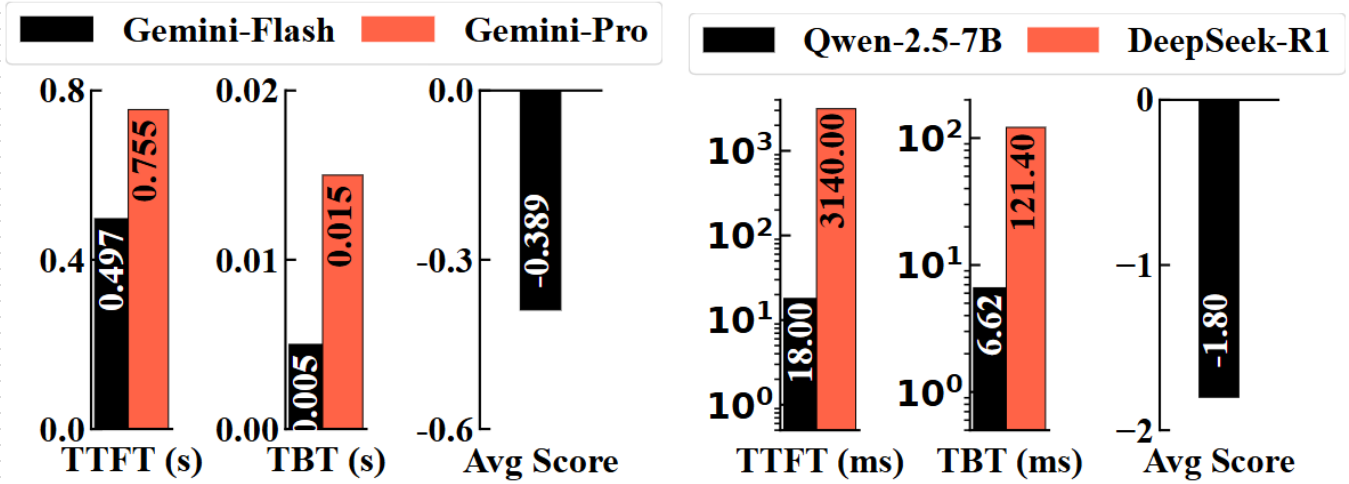
- Improving parallelism
- GPU utilization
  - chunked-prefill, disaggregating PD
- Memory efficiency
  - pageattention, radixattention
- Reducing serving latency
  - parrot



IC-Cache

Accuracy-Cost-Latency Tensions in LLM Serving.

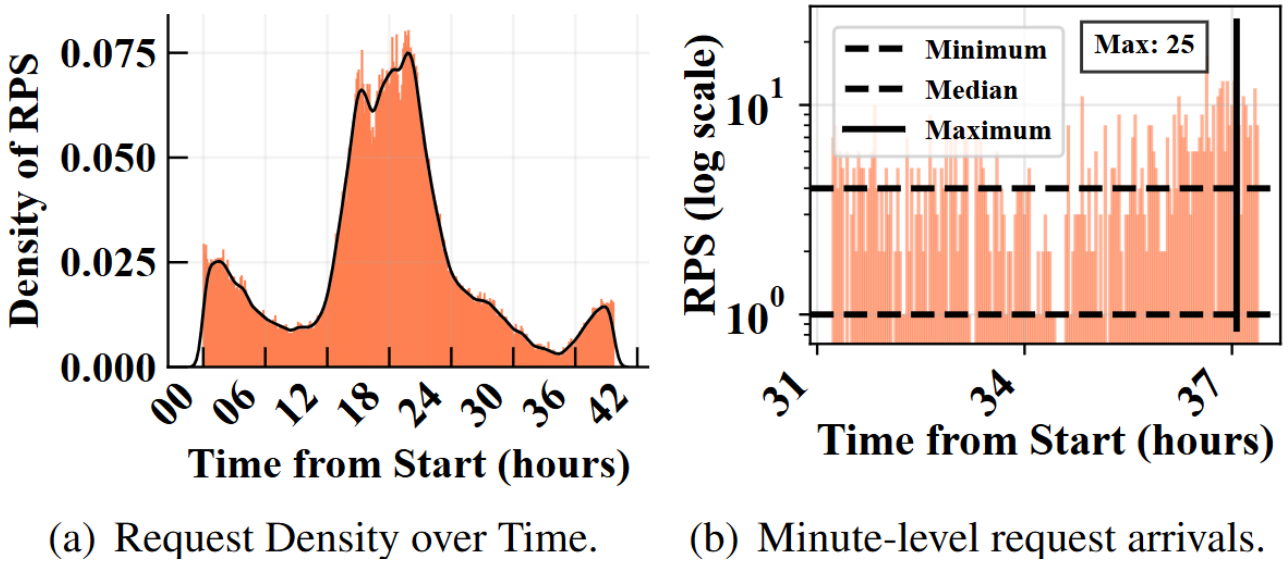
- workload：LMSys-Chat(10K real-world user requests)
- LLM as a Judge
  - DeepSeek-R1 as the Judge for Gemini experiments
  - Gemini-1.5Pro as the Judge for DeepSeek-R1 experiments



(a) Gemini on conversation task. (b) Qwen and DeepSeek.

Substantial Scaling Demands

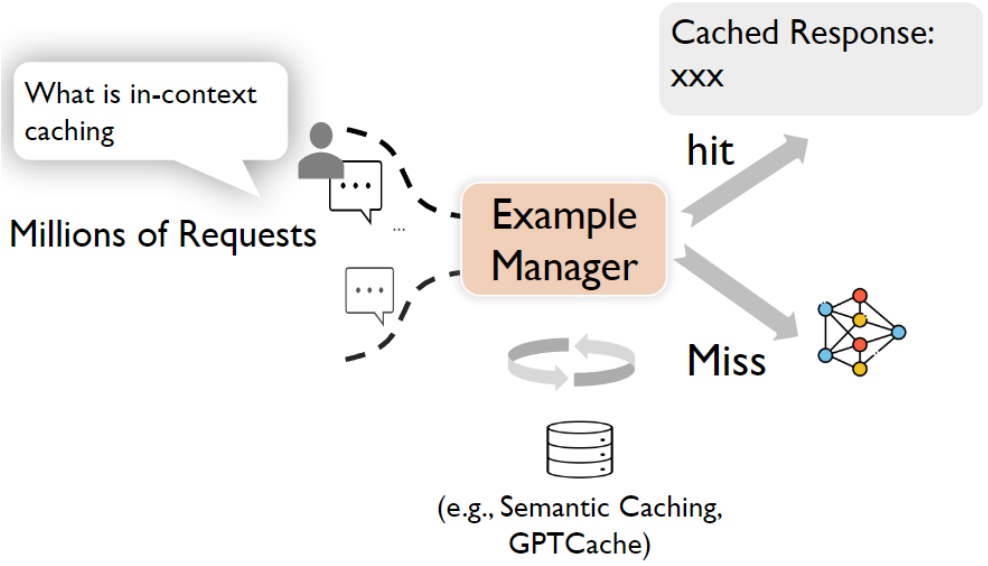
- (a)workload：Azure LLM serving trace
  - transient load surges need significant resource overprovisioning
- (b)workload: Helicone (Open source LLM observability platform)
  - severe latency fluctuation

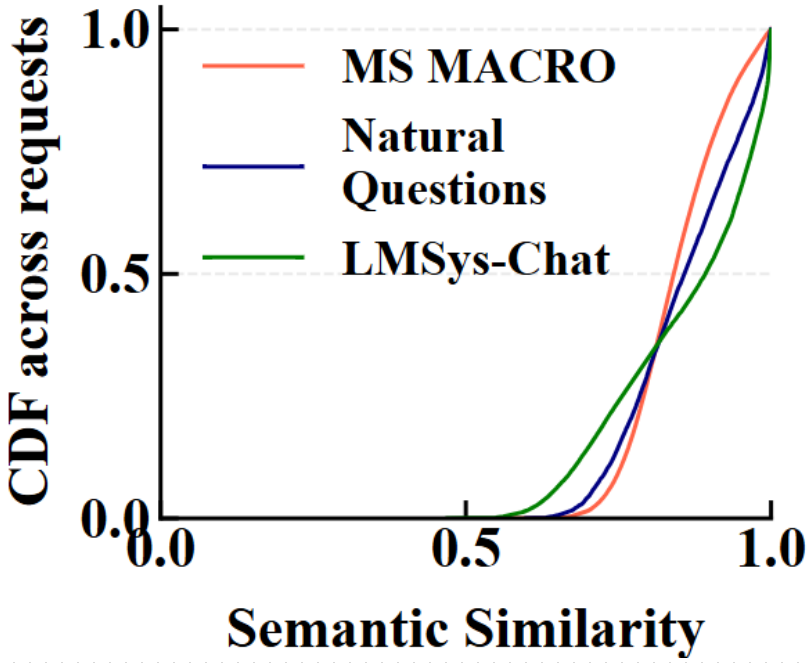


(a) Request Density over Time. (b) Minute-level request arrivals.

Repurposing Historical Requests

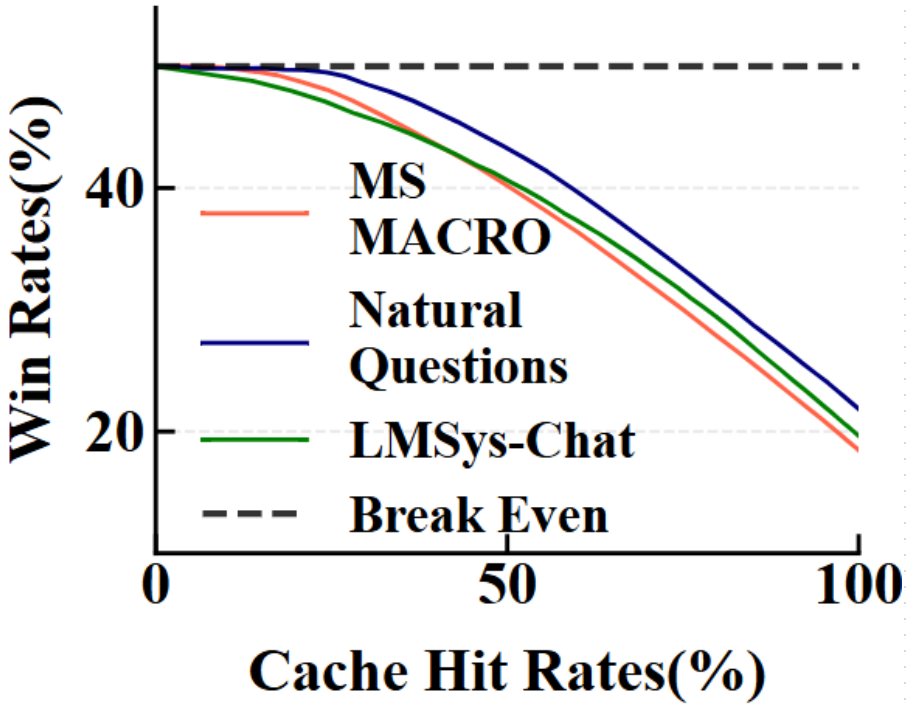
- Requests lead to pervasive similarity
- workload :
  - MS MACRO(1 million Bing Search requests)
  - Natural Question(300K requests)
  - LMSys-Chat(1 million requests)





Repurposing Historical Requests

- Semantic caching is widely adopted yet limits efficiency and quality
- workload：
  - MS MACRO(1 million Bing Search requests)
  - Natural Question(300K requests)
  - LMSys-Chat(1 million requests)



Naive semantic caching is unsatisfactory

IC-Cache

How to better leverage history toward low latency and cost for users & high throughput for providers?

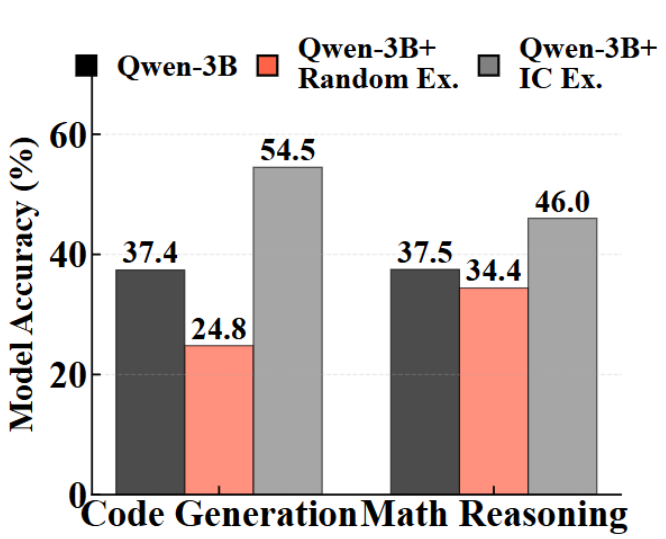
Opportunities for In-Context Learning (ICL)

- **In-Context Learning: LLMs learn from examples w/o parameter updates**
  - **Enable real-time capability augmentation for small LLMs**

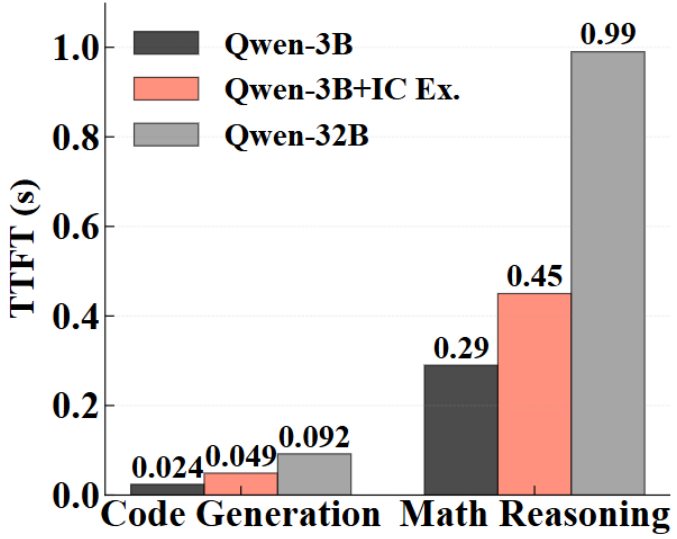
In-Context from Large Model :
Input: "The plot of this movie is draggy, the acting is awkward, it wasted my time." Output: Negative
Input: "The visuals are stunning, the performances are outstanding, highly recommended!" Output: Positive
Request :
Input: "The service at this restaurant is just average." Output:
Response :
Negative

Repurposing Historical Requests

- History can enable live LLM capability augmentation
- Qwen2.5-3B augmented with examples from Qwen2.5-32B
- workload：
  - Math-500-Hard reasoning benchmarks
  - NL2Bash code generation



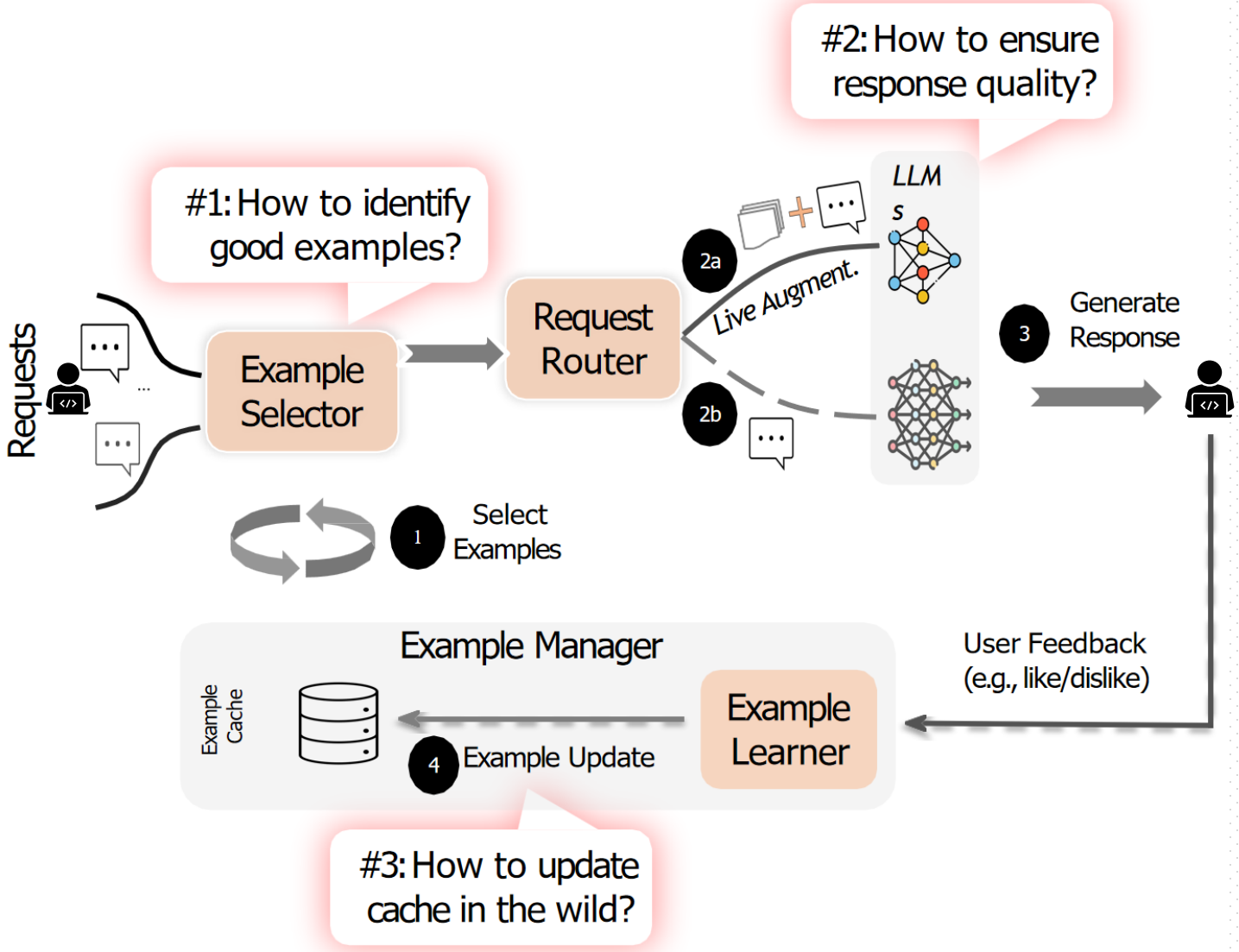
(a) Response quality with examples.



(b) TTFT performance.

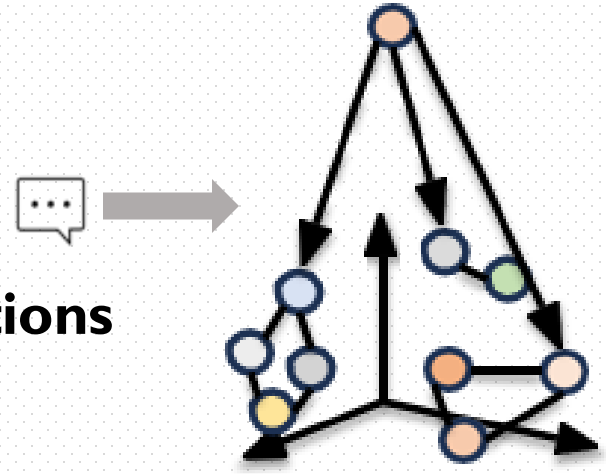
In-context Caching

Leveraging history for real-time LLM augmentation



How to identify good examples

- What is "High Utility"?
  - Examples that significantly improve response quality
  - Enable smaller models to mimic larger ones
  - Enable offloading to cheaper models
- Intuitive Approach: Semantic Relevance
  - Use BERT/T5 embeddings for semantic representations
  - Retrieve similar examples via KNN
  - Select top-k using cosine similarity

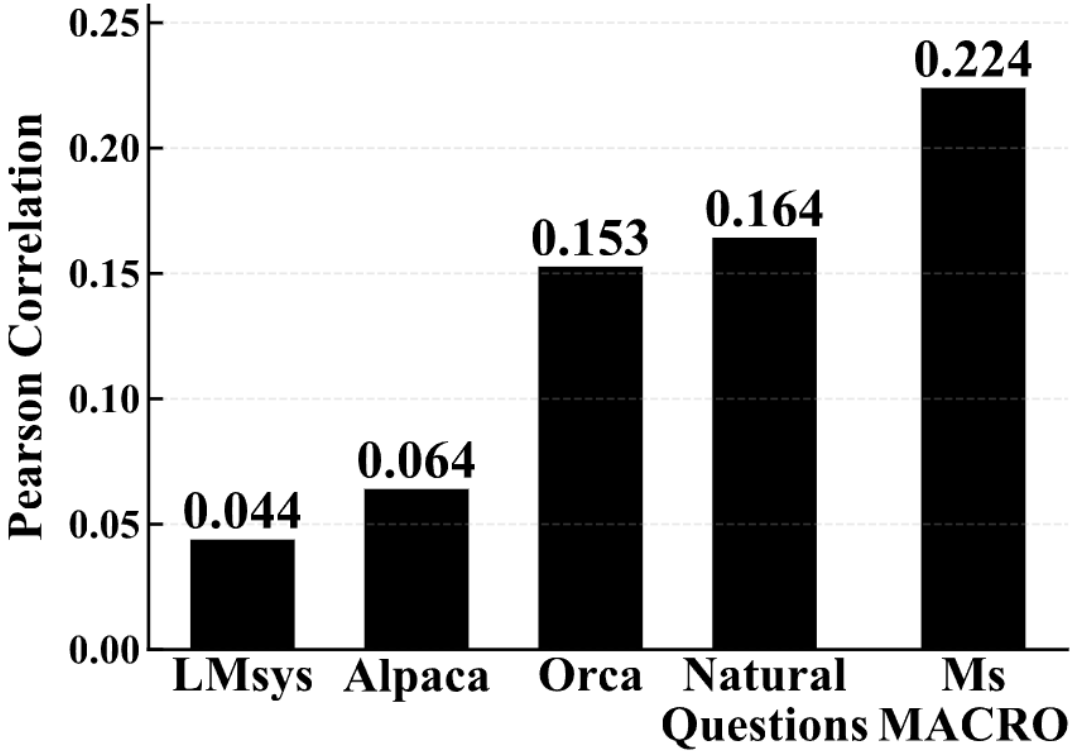


Relevancy-based method is insufficient

- Weak Correlation with Actual Utility

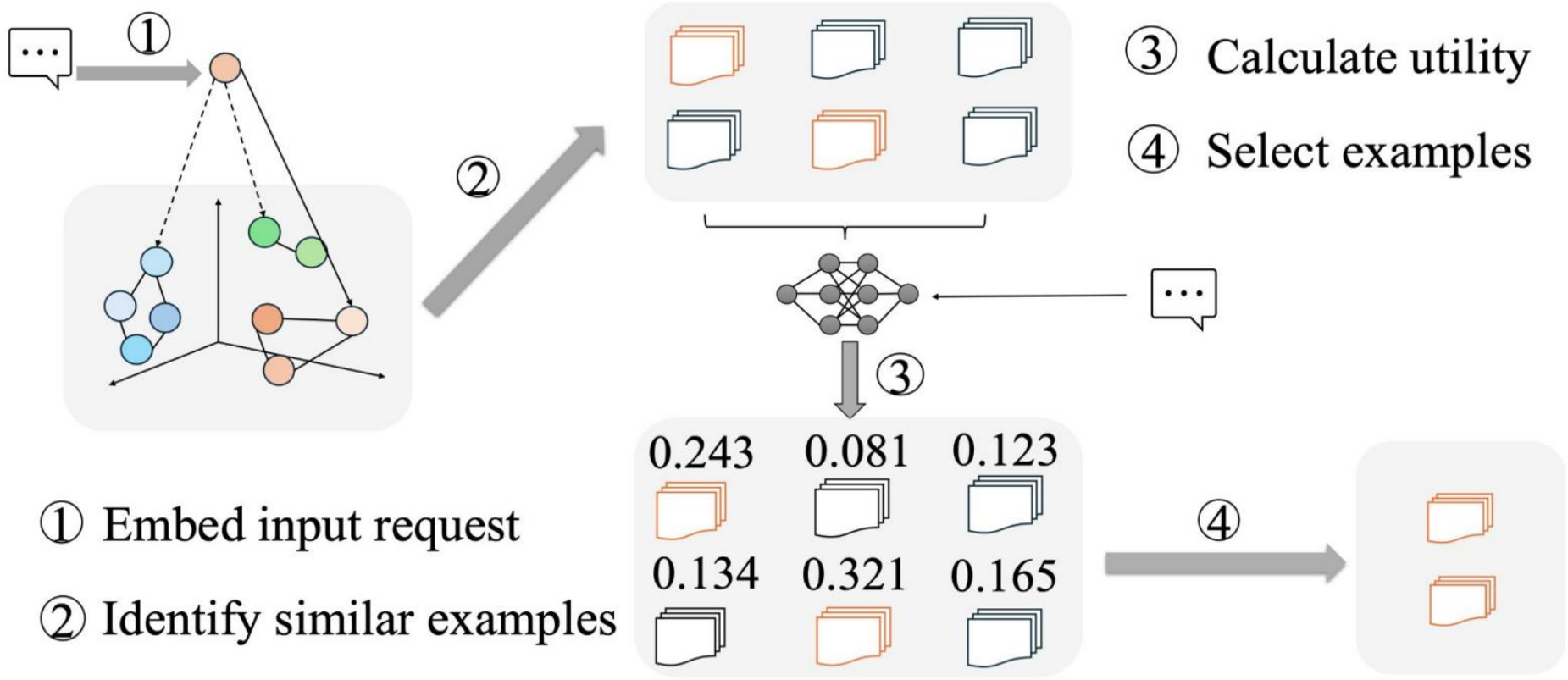
  =>Semantic relevance ≠ response quality improvement

- Reasons for Weak Correlation
  - Low-quality response
  - Redundant skills
  - Format/style mismatches
  - Examples too advanced



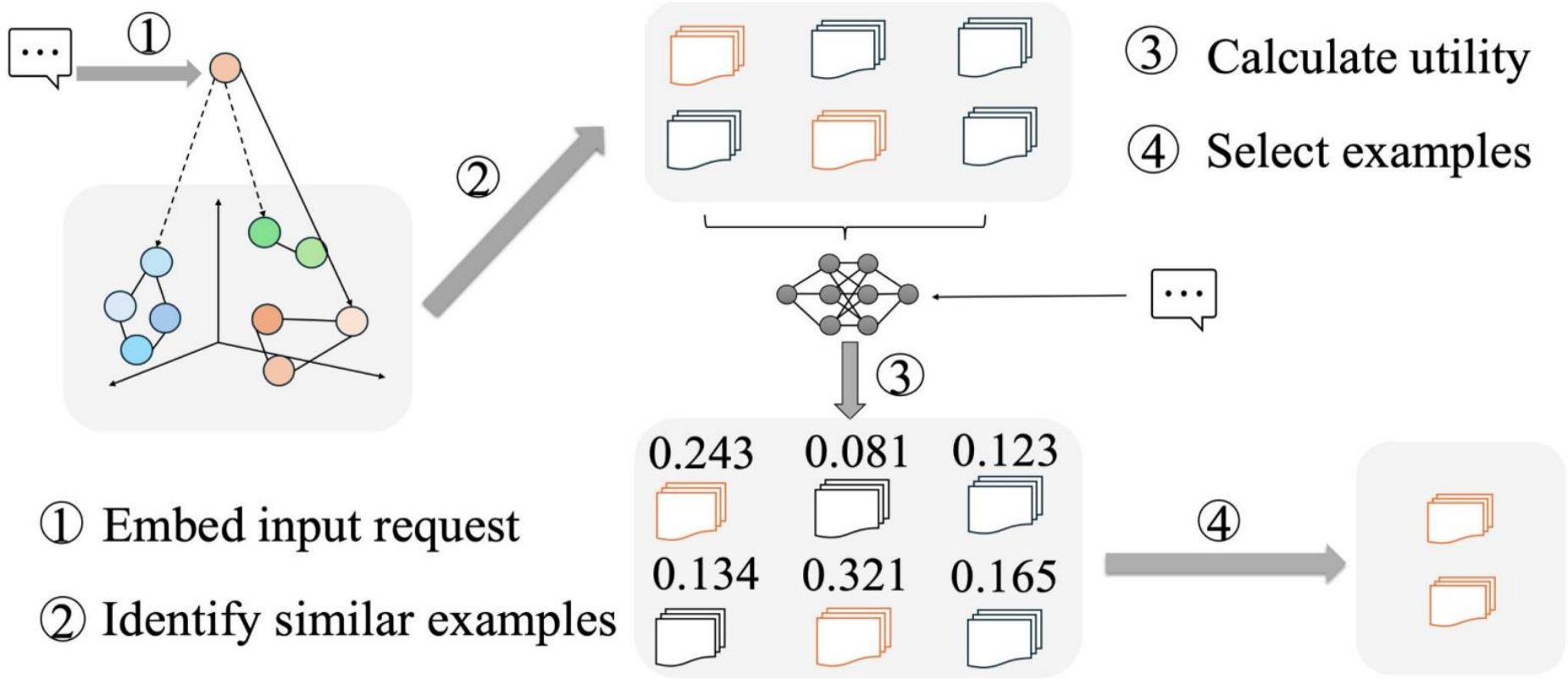
Two-stage example selector

- Relevancy + Small Proxy Model (e.g., TinyBert)



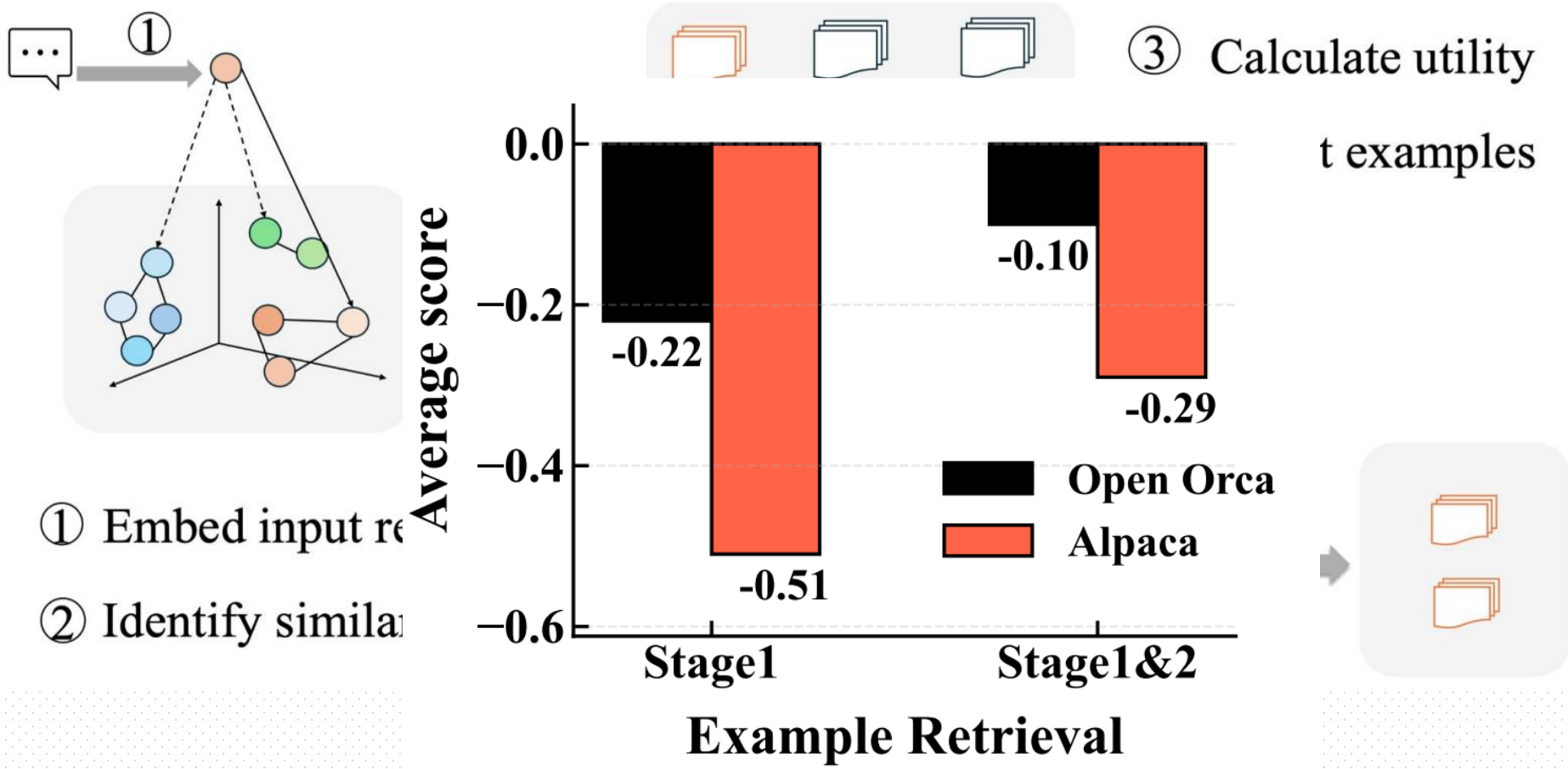
Two-stage example selector

- Relevancy + Small Proxy Model (e.g., TinyBert)



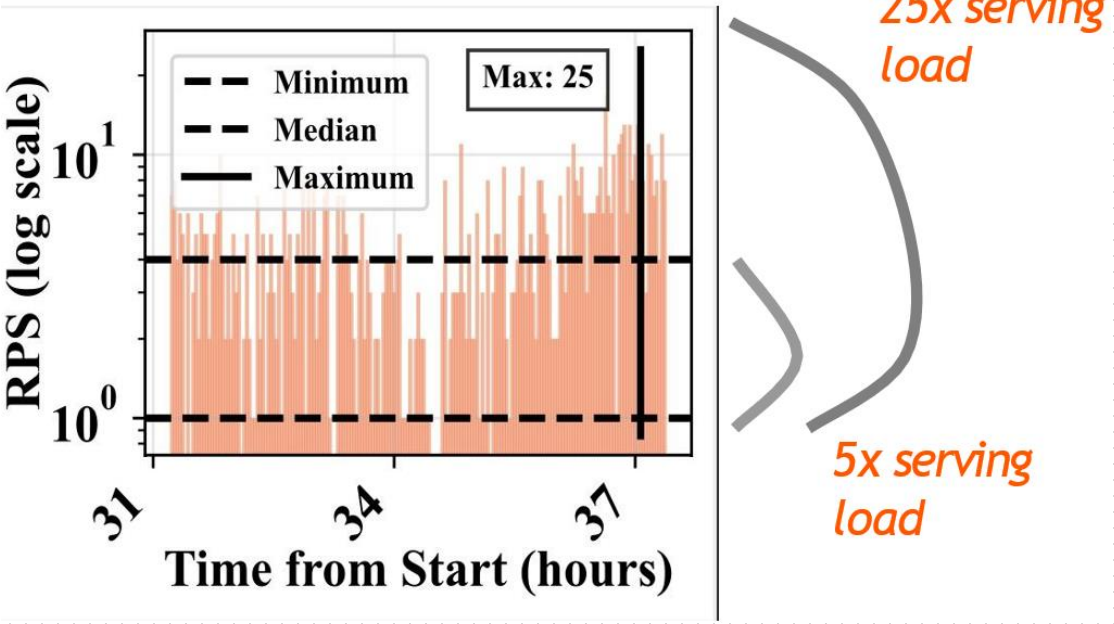
Two-stage example selector

- Relevancy + Small Proxy Model (e.g., TinyBert)



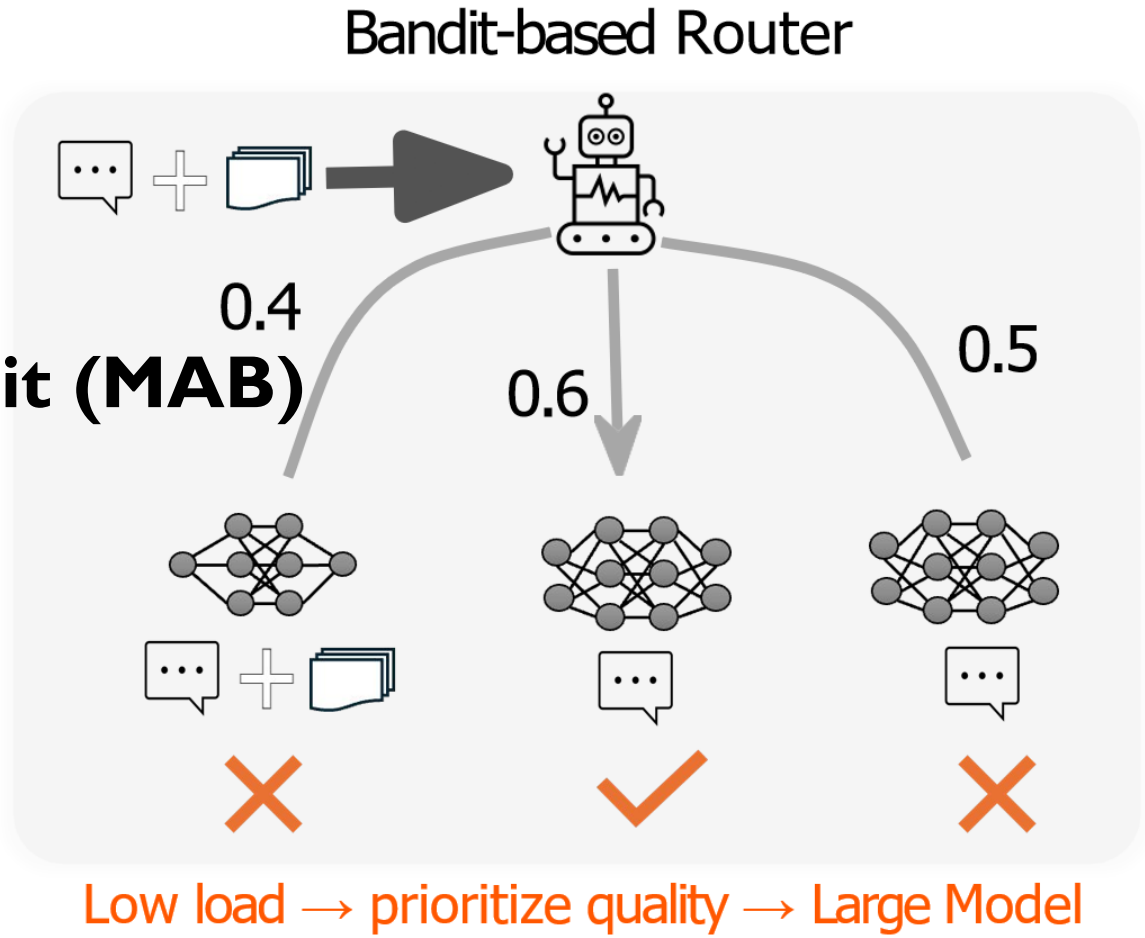
How to ensure response quality?

- **Problem: Good Examples ≠ No Quality Drop**
  - Small models with examples may still underperform
  - Aggressive offloading hurts quality
  - Conservative offloading limits efficiency
- **Dynamic Deployment Challenges**
  - Bursty workloads
  - Real-time adaptive routing decisions



Request routing for response quality

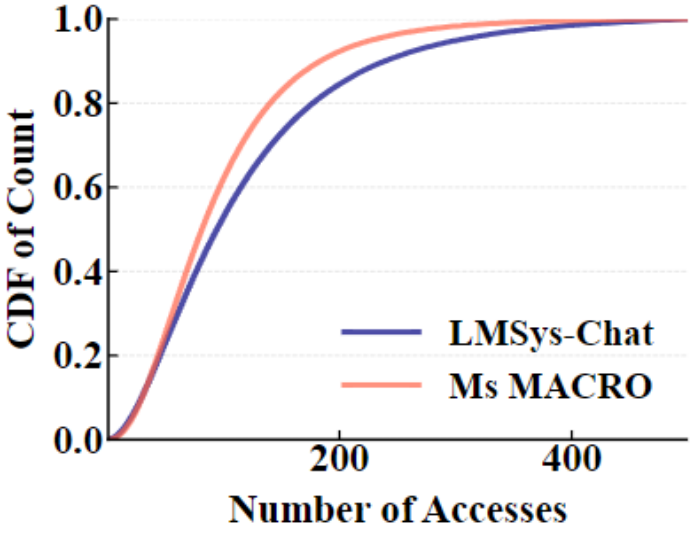
- What would be a good routing design?
  - Load- and quality-aware
  - Compute- and data-efficient
  - Real-time adaption
- Contextual Multi-Armed Bandit (MAB)
  - Context: Request + selected examples
  - Arms: Available models
  - Reward: Quality metrics
  - Policy: Load-aware biasing

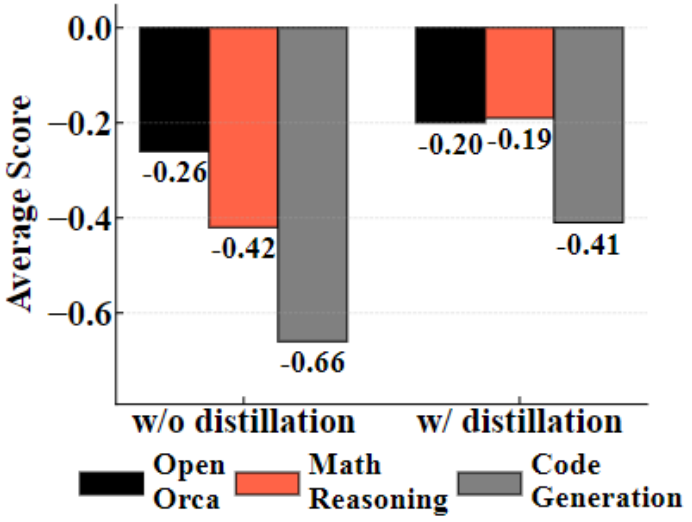


Example Manager: Effective Caching in the Wild

❑Cost-aware Example Replay

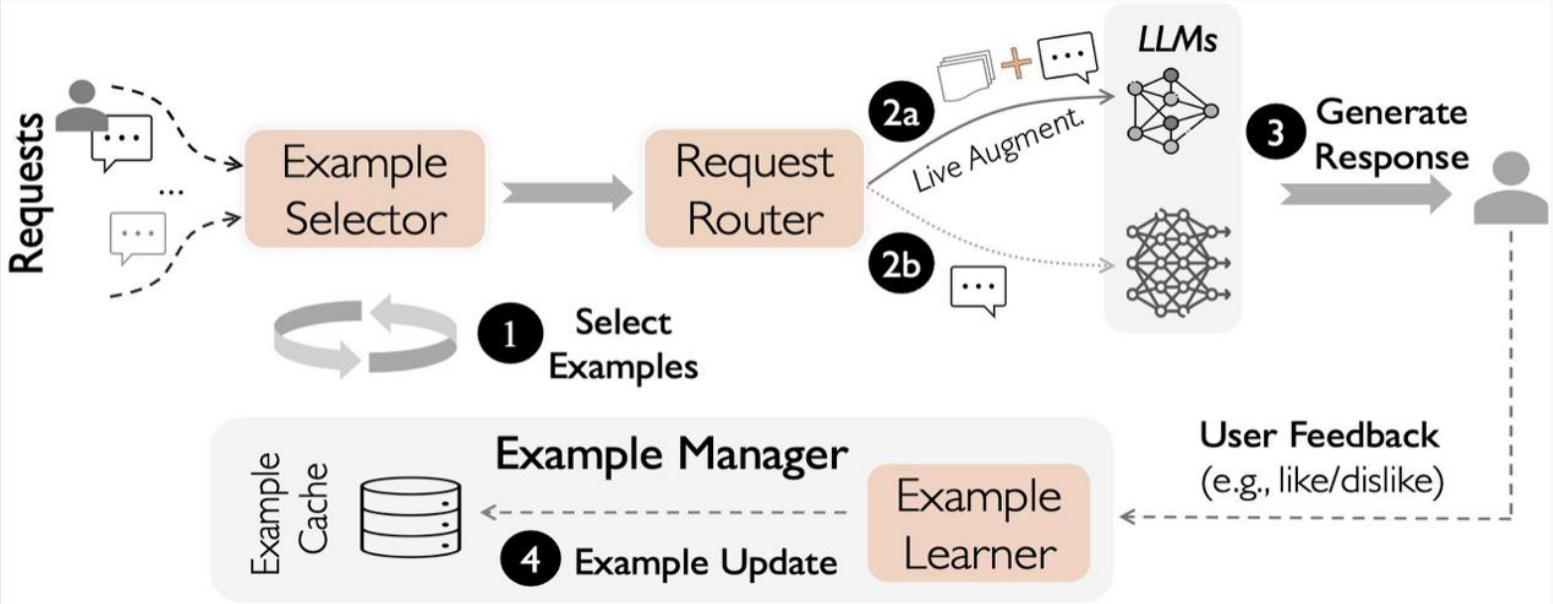
- ❑ large variance in response quality due to the stochastic nature of generation
- ❑ both the frequency and effectiveness of example reuse exhibit large heterogeneity, often following a long-tail distribution
- ❑ $G(e) = (1 - normalized\_response\_quality) \times normalized\_model\_cost$





Evaluation

Five models including Gemini and DeepSeek R1
Millions of queries on 8 datasets with LLM as a judge



28-71% latency and 1.4-5.9 throughput improvements
w/o hurting qualities

Evaluation Setups

❑Tasks and Datasets

Task	Dataset	Example Size	Request Size
Conversation	Alpaca[66]	32,392	1,800
	lmsys-chat-1m [79]	273,043	15,170
	OpenOrca [49]	774,285	43,016
Question Q&A	MS MARCO [54]	808,731	101,092
	Natural Questions [41]	300,000	7,830
Translation	WMT-16-PM [22]	600,000	1000
Code Gen.	Nl2bash [75]	8090	609
Math reasoning	Math500-Level5[45]	7500	5000

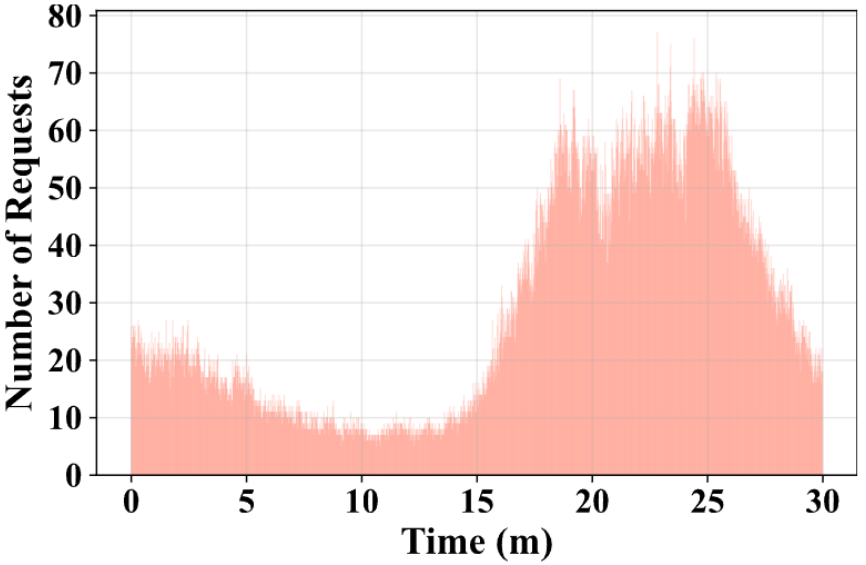
❑Baselines

- ❑w/o IC-Cache
- ❑RouteLLM
- ❑LongRAG
- ❑Semantic Cache

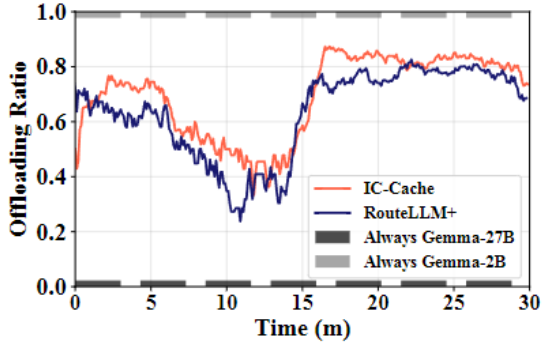
❑TTFT/TBT/thpt

❑Quality Metrics

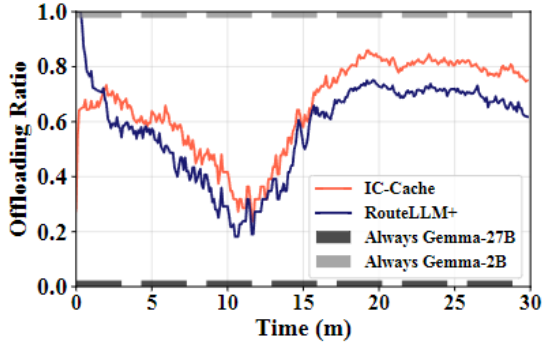
- ❑LLM-as-a-Judge with 7-point scale (-3 to +3)
- ❑Win Rate = (#wins + 0.5 × #ties) / #total



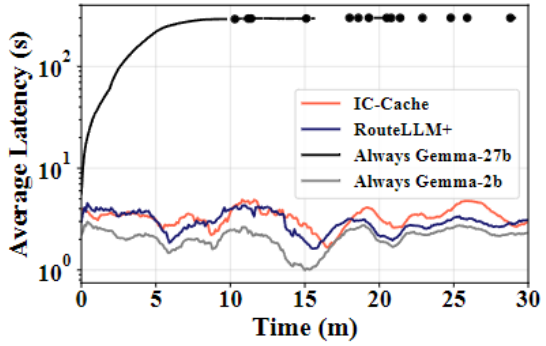
IC-Cache adapts effectively for better efficiency



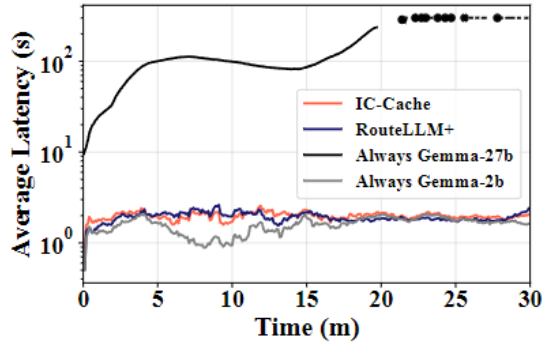
(a) Load Offloading (MS MARCO).



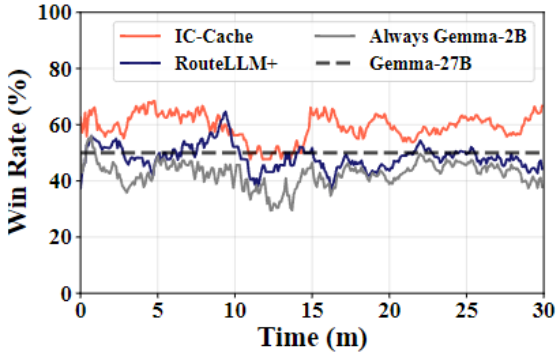
(b) Load Offloading (Natural Questions).



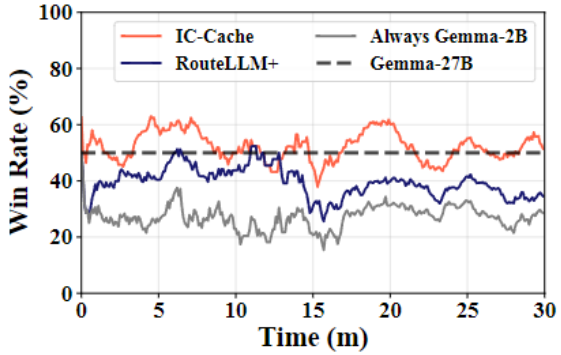
(c) Serving latency (MS MARCO).



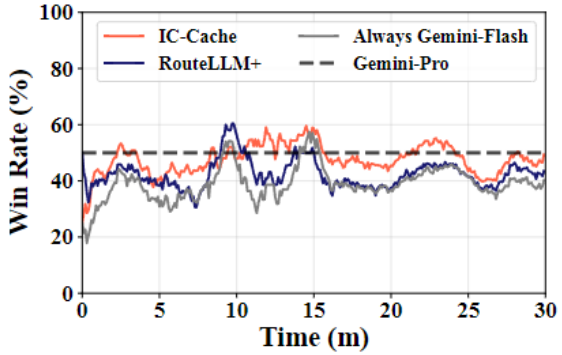
(d) Serving latency (Natural Questions).



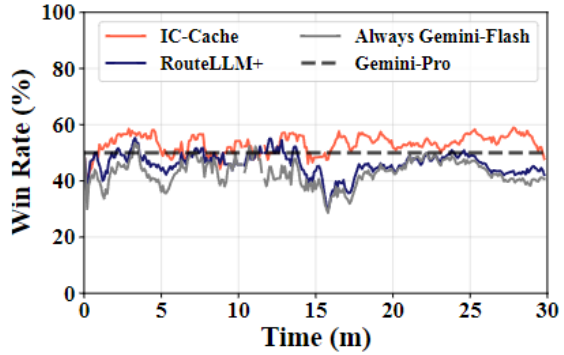
(e) Response Quality (MS MARCO).



(f) Response Quality (Natural Questions).

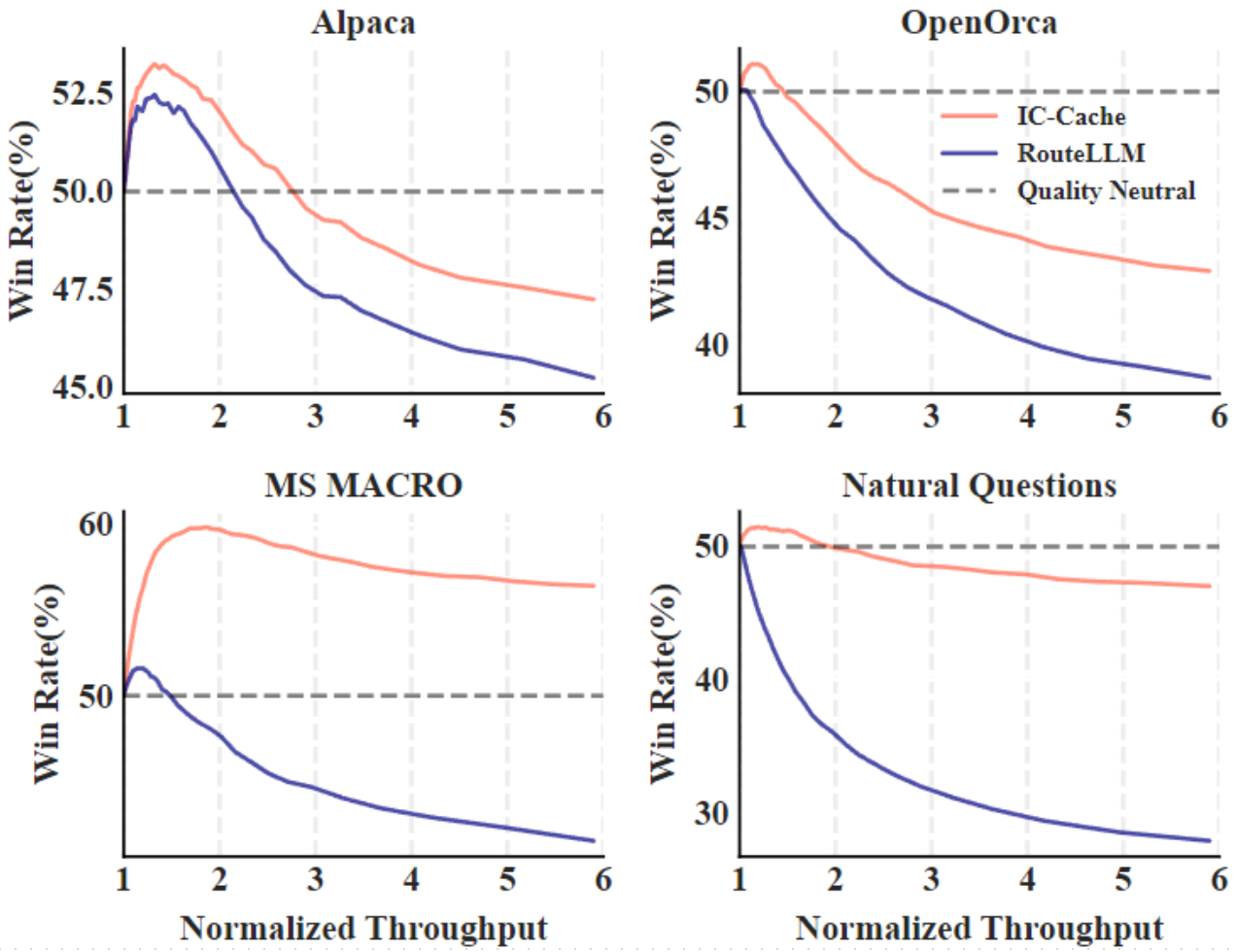


(g) Response Quality (LMSys).



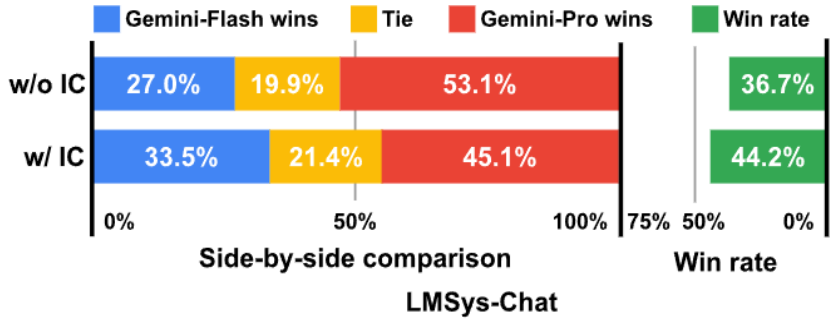
(h) Response Quality (Orca).

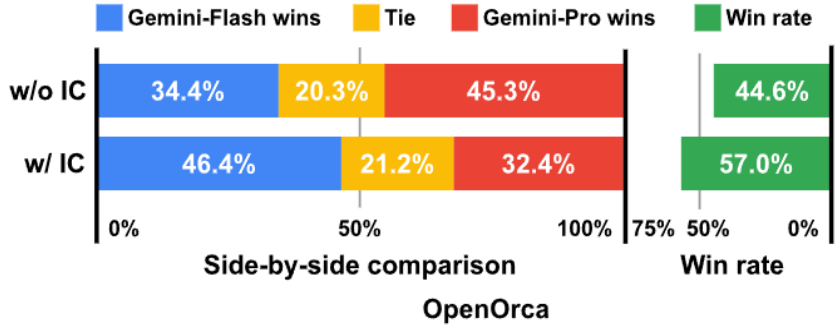
IC-Cache enables better quality-efficiency tradeoff

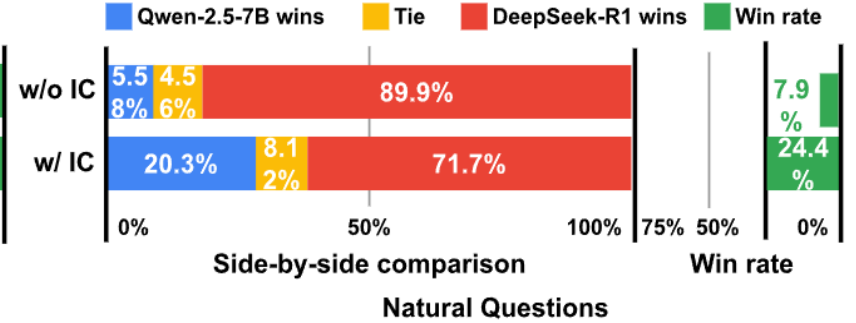


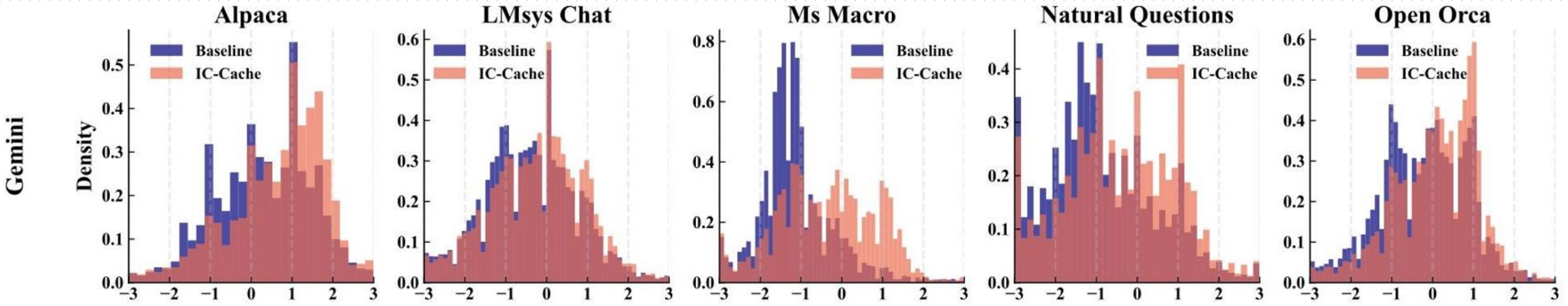
Evaluation

- **IC-Cache improves the quality of generation across different tasks for Gemini, Qwen, and DeepSeek series models.**



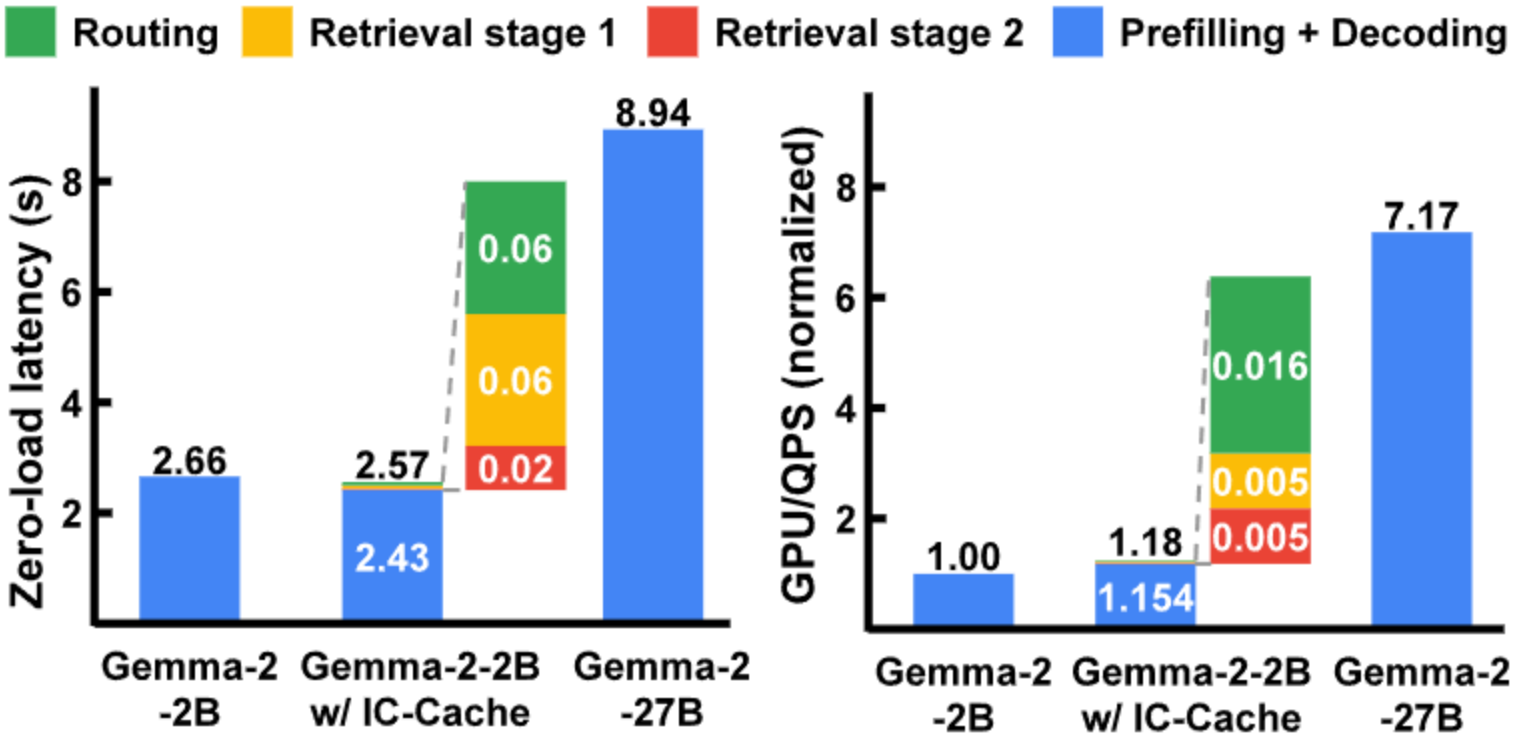






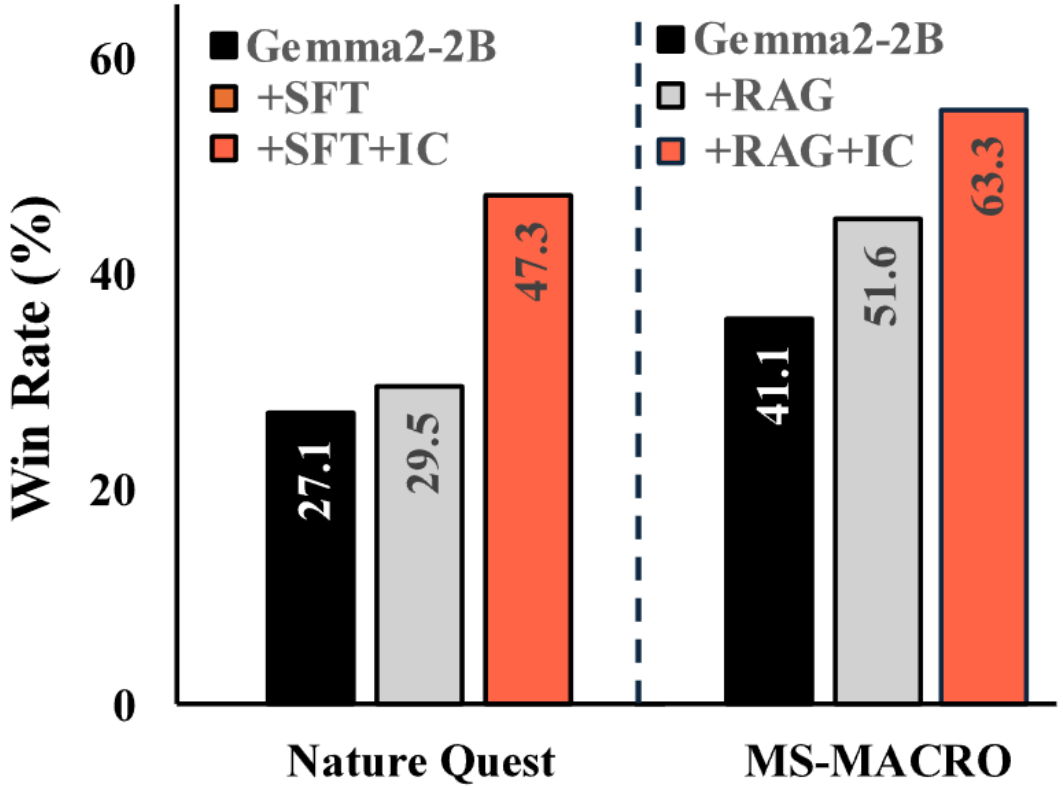
Evaluation

- IC-Cache introduces little overhead (left) while improving cost-efficiency in sustaining serving throughput (right).



IC-Cache augments existing serving infra effectively

- **IC-Cache augments supervised fine-tuning (SFT) and RAG deployments.**



Evaluation

- IC-Cache delivers improvement under different the example cache sizes.
- IC-Cache improves serving efficiency across serving loads.