NEURAL INFORMATION PROCESSING SYSTEMS 2025

NUS National University of Singapore

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Agency for Science, Technology and Research SINGAPORE

EAI Challenge in NeurIPS 2025

Team SingaX

Team members:
Xinyuan Niu, Zhiliang Chen, Vernon Yan Han Toh,
Yanchao Li, Zhengyuan Liu, Nancy F. Chen

Our Team

Xinyuan Niu
National University of Singapore

Zhiliang Chen
National University of Singapore

Vernon Yan Han Toh
Nanyang Technological University, Singapore

Yanchao Li
Nanyang Technological University, Singapore

Zhengyuan Liu
Agency for Science, Technology and Research (A*STAR), Singapore

Nancy F. Chen
Agency for Science, Technology and Research (A*STAR), Singapore

Overview of the solution by Team SingaX

Areas we aim to tackle

- Prompt optimization — Generate better output
- Inference generation — Generate output candidates
- Output verification — Iterate/select best output

Our observation: ambiguity in prompts causes confusion in LLM reasoning

- LLM explicitly express confusion

```
But in the OPEN description, it says "toggle off the target object first if want to open it" - this is confusing.
```

- “Conflicting” ambiguity

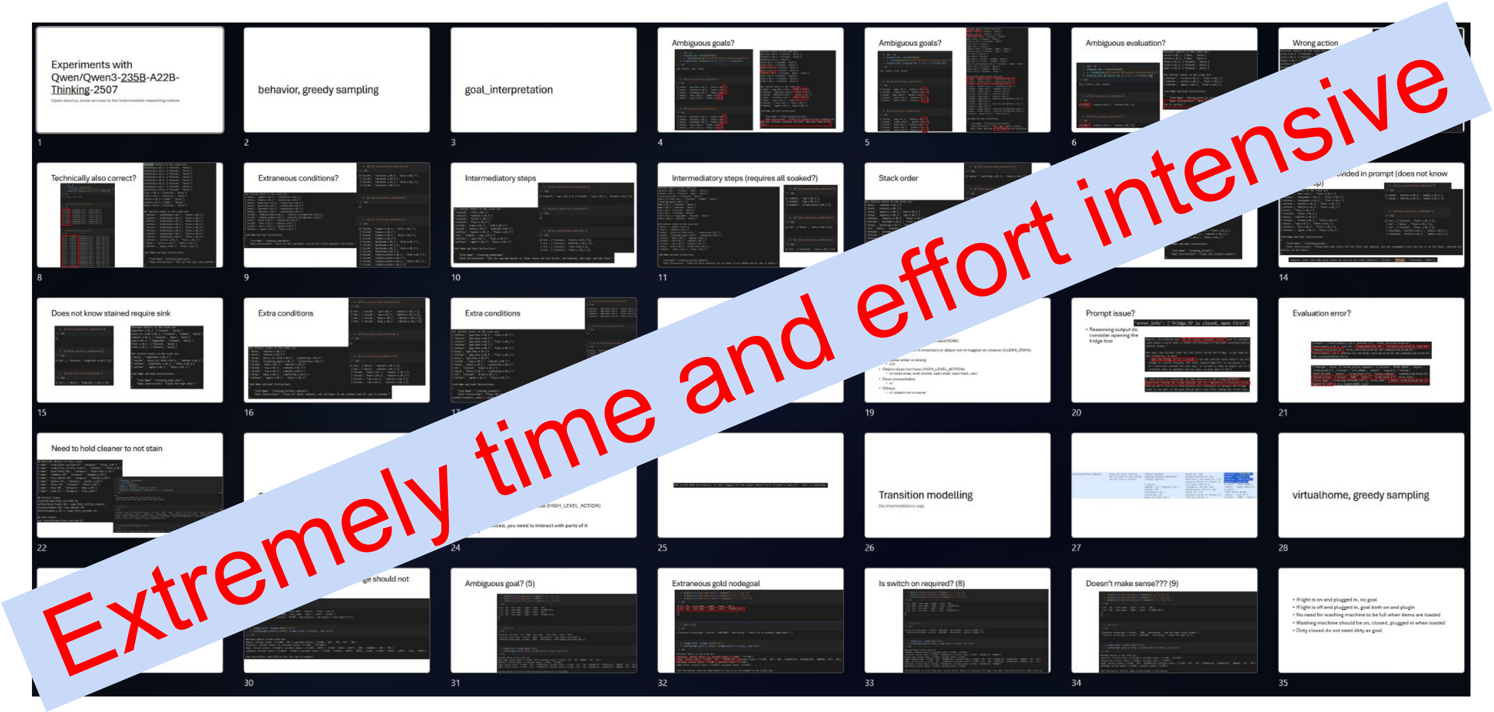
```
However, the problem says: "Do not output redundant states." and "A redundant state means a state that is either not necessary or has been satisfied before without broken."

But note: the initial state has the fruits inside the fridge. To get them on the countertop, we must:
  - Open the fridge (if it's closed) -> but the initial state doesn't say the fridge is closed? Actually, the state "open(fridge.97)" is not given, so it's initially closed? The rule says: if we want to take an object out of a container that is openable and not open, we must open it first.
```

```
1. Both fruits on countertop: we take them out of the fridge (without explicitly stating the fridge opening, but it's implied as a necessary action that we don't list as a state because it's transient? or because the fridge state is not part of the goal and we don't care after taking the fruits out).
```

```
'error_info': ['fridge_97 is closed, open first']
```

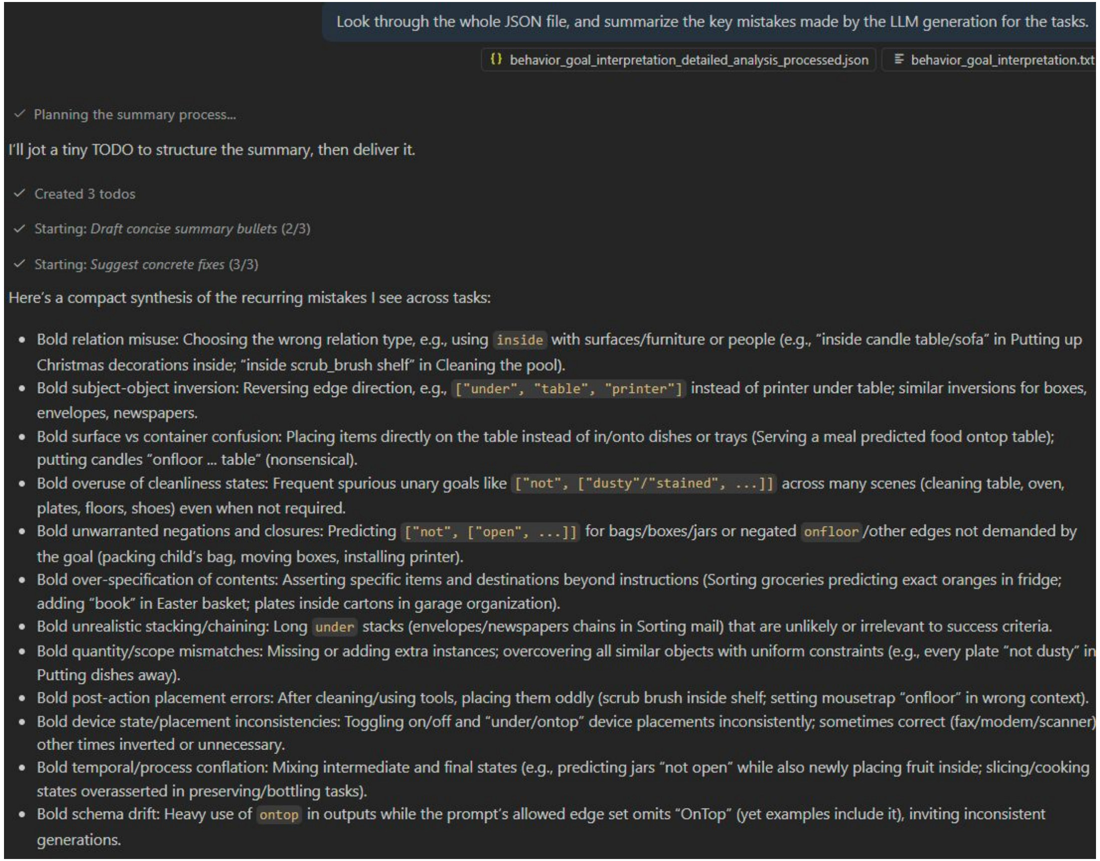
We should reduce the ambiguity in prompts, but how?

- **Manual approach**: identifying common errors from logs



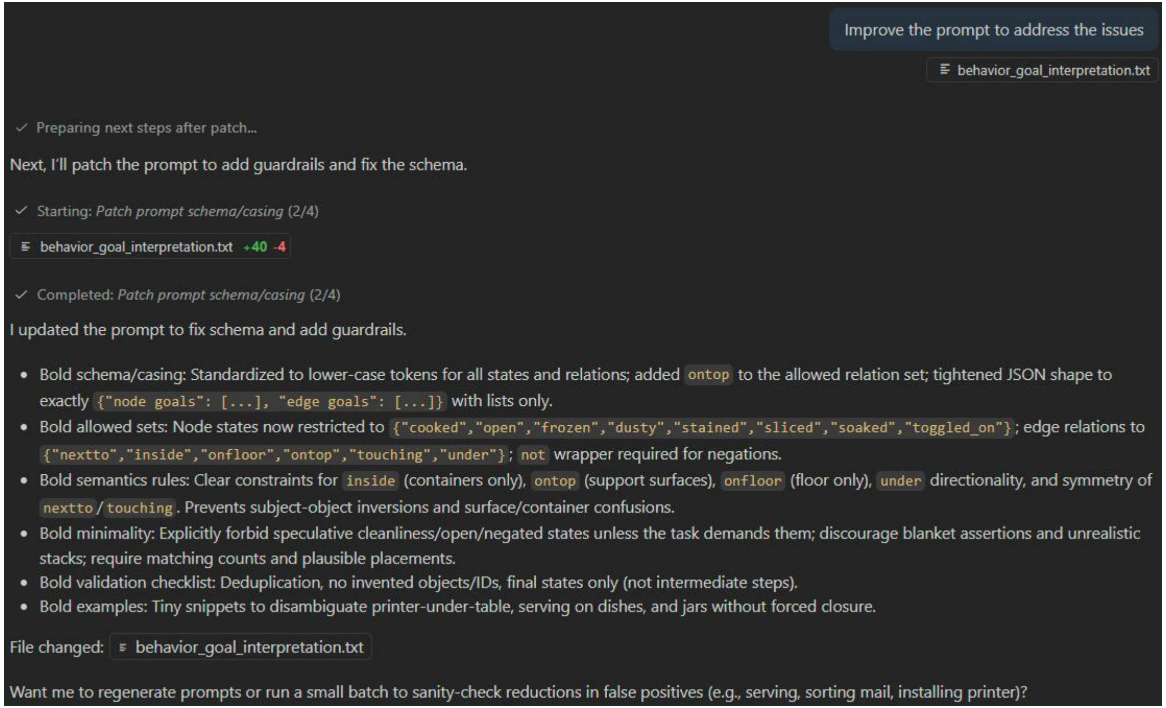
Preliminary idea: one automated way is… asking LLMs to do this

- GPT-5 agent on VS-Code Co-pilot extension
- "Look through the whole JSON file, and ***summarize*** the key mistakes made by the LLM generation for the tasks."
- Attach log file
- Long context window: ingest entire log file



Prompt Induction: Learning from History or Mistakes ⇒ Better Instructions

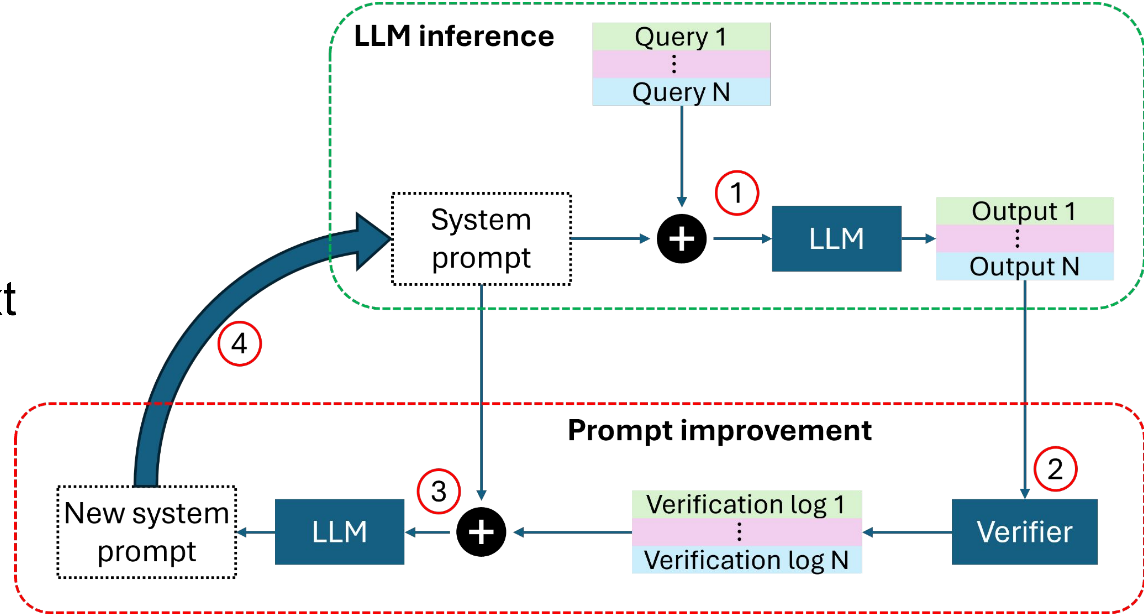
- “Improve the prompts to address the issues”
- Attach prompt file
- LLM agent directly applies patch to the system prompt file
- Distinct from existing prompt optimization works (e.g., [1,2]), we directly use the verifier logs as feedback during the prompt improvement step



[1] Zhou et. al. (2023). Large language models are human-level prompt engineers. In Proc ICLR.
[2] Guo et. al. (2025). Evoprompt: Connecting llms with evolutionary algorithms yields powerful prompt optimizers. arXiv:2309.08532.

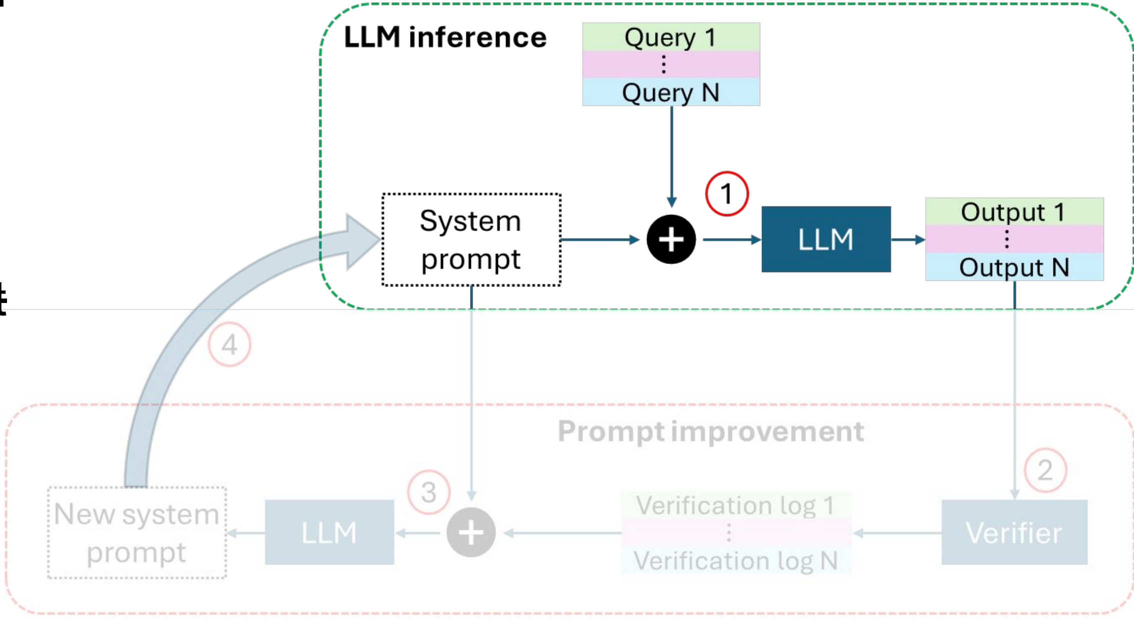
Overall pipeline of Iterative Prompt Induction (Development phase)

1. Use initial system prompt and generate initial LLM outputs
2. Pass LLM outputs through verifier
3. Put all verifier logs in-context into LLM, and ask LLM to improve system prompt
4. Repeat step 1 onwards with new system prompt (as required)



Overall pipeline (Test phase) — When verifier is not available

1. Use **improved** system prompt and generate LLM outputs
2. ~~Pass LLM outputs through verifier~~
3. ~~Put all verifier logs in-context into LLM, and ask LLM to improve system prompt~~
4. ~~Repeat step 1 onwards with new system prompt (as required)~~



Evaluate on actual large LLM: Clear improvements across the board

	behavior dev set			virtualhome dev set		
	Default prompt	Ours		Default prompt	Ours	
goal_interpretation (f1)	79.7	82.3	+ 2.6	43.6	61.1	+17.5
subgoal_decomposition (task sr)	69.0	77.0	+ 8.0	90.5	92.9	+ 2.4
action_sequencing (task sr)	79.0	84.0	+ 5.0	65.9	77.0	+11.1
transition_modeling (f1)	67.9	84.5	+16.6	47.3	81.0	+33.7
transition_modeling (sr)	86.0	98.4	+12.4	75.0	97.9	+22.9
avg_perf	76.2	83.7		65.3	80.1	

Improved prompts also work well on small LLMs

Table 2: Overview of results (%) on the evaluation phase. V: VirtualHome, B: BEHAVIOR.

	Goal Interpretation		**Action Sequencing**				**Subgoal Decomposition**				**Transition Modeling**				**Average Perf.**		
	F_1		$TaskSR$		$ExecSR$		$TaskSR$		$ExecSR$		F_1		$PlannerSR$		$ModuleSR$		**Overall Perf.**
Model	V	B	V	B	V	B	V	B	V	B	V	B	V	B	V	B	
Qwen 3 4B	23.9	39.7	58.4	39.0	67.0	56.0	54.9	43.0	79.1	54.0	30.3	35.9	43.5	46.0	43.53	40.66	42.09
(+ Optimized Prompt)	38.6	30.6	63.7	41.0	74.6	51.0	55.7	55.0	79.8	70.0	68.8	52.0	47.0	70.0	53.98	46.90	50.44
	(↑ 14.7)	(↓ 9.1)	(↑ 5.3)	(↑ 2.0)	(↑ 7.6)	(↓ 5.0)	(↑ 0.8)	(↑ 12.0)	(↑ 0.7)	(↑ 16.0)	(↑ 38.5)	(↑ 16.1)	(↑ 3.5)	(↑ 24.0)	(↑ 10.4)	(↑ 6.2)	(↑ 8.4)
Qwen 3 8B	23.4	69.7	58.4	44.0	69.5	58.0	58.9	40.0	81.3	50.0	38.3	53.7	80.2	73.0	49.99	54.26	52.13
(+ Optimized Prompt)	39.5	73.3	67.9	56.0	80.7	65.0	61.1	56.0	82.7	70.0	81.0	63.7	92.5	91.0	63.81	65.66	64.74
	(↑ 16.1)	(↑ 3.6)	(↑ 9.5)	(↑ 12.0)	(↑ 11.2)	(↑ 7.0)	(↑ 2.2)	(↑ 16.0)	(↑ 1.4)	(↑ 20.0)	(↑ 42.7)	(↑ 10.0)	(↑ 12.3)	(↑ 18.0)	(↑ 13.8)	(↑ 11.4)	(↑ 12.6)
Qwen 3 14B	24.8	71.0	66.0	46.0	82.1	58.0	62.5	45.0	82.1	53.0	43.0	59.6	63.5	44.0	51.64	53.45	52.54
(+ Optimized Prompt)	41.5	73.0	65.5	57.0	80.9	65.0	66.3	66.0	86.6	78.0	79.9	64.6	82.4	83.0	63.61	67.45	65.53
	(↑ 16.7)	(↑ 2.0)	(↓ 0.5)	(↑ 11.0)	(↓ 1.2)	(↑ 7.0)	(↑ 3.8)	(↑ 21.0)	(↑ 4.5)	(↑ 25.0)	(↑ 36.9)	(↑ 5.0)	(↑ 18.9)	(↑ 39.0)	(↑ 12.0)	(↑ 14.0)	(↑ 13.0)
Qwen 3 32B	28.0	65.5	63.2	59.0	77.3	71.0	65.9	47.0	86.3	55.0	45.4	62.5	76.2	78.0	54.48	60.44	57.46
(+ Optimized Prompt)	39.4	68.5	67.4	63.0	81.1	72.0	66.3	66.0	85.7	79.0	78.2	68.3	84.4	89.0	63.60	69.04	66.32
	(↑ 11.4)	(↑ 3.0)	(↑ 4.2)	(↑ 4.0)	(↑ 3.8)	(↑ 1.0)	(↑ 0.4)	(↑ 19.0)	(↓ 0.6)	(↑ 24.0)	(↑ 32.8)	(↑ 5.8)	(↑ 8.2)	(↑ 11.0)	(↑ 9.1)	(↑ 8.6)	(↑ 8.9)
Qwen 3 30B A3B	26.80	79.10	69.30	53.00	81.50	68.00	61.10	56.00	83.90	66.00	36.70	49.70	82.10	69.00	54.15	61.86	58.01
(+ Optimized Prompt)	42.80	69.70	70.00	54.00	83.60	65.00	64.30	73.00	86.10	88.00	75.60	61.50	92.60	88.00	65.30	67.86	66.58
	(↑ 16.00)	(↓ 9.40)	(↑ 0.70)	(↑ 1.00)	(↑ 2.10)	(↓ 3.00)	(↑ 3.20)	(↑ 17.00)	(↑ 2.20)	(↑ 22.00)	(↑ 38.90)	(↑ 11.80)	(↑ 10.50)	(↑ 19.00)	(↑ 11.15)	(↑ 6.00)	(↑ 8.6)

Main changes in improved prompts via our interactive induction

- Rules and checklist
- Templates for common tasks

However, LLMs can still hallucinate and make mistakes

- Double check templates for correctness
- Add in a few more rules
- *Minimal manual effort*, as LLM has already done the bulk of the work

```
8. If you want to apply an action on an object, you should WAL

9. If an object is placed with a grabbable container, the cont

Rules and recipes to avoid common mistakes (follow strictly):

R1. Always WALK before acting on an object.
- Before applying ANY action to an object, first WALK to that
- You may skip WALK only if the input explicitly states the ch

R2. Use PLUGIN only when required by the state.
- Use PLUGIN only if the device has HAS_PLUG and its node stat
- If it is already PLUGGED_IN (or no plug state is given), do

R3. Canonical action sequences (maintain this order):
- Power and use an electric device (e.g., computer, light, ste
  WALK device → (PLUGIN if PLUGGED_OUT) → SWITCHON device → th
- Washing hands/dishes: WALK faucet or sink area → SWITCHON fa
- Getting a drink: WALK container (cup/glass) → GRAB container

R4. Containers and access:
- If an object is INSIDE a CLOSED container, WALK the containe

R5. Goal-aligned final actions:
- For "character is ON to X": make SIT or LIE on X the last ac
- For "character is FACING/CLOSE to X": include TURNTO/LOOKAT/

R6. Avoid redundancy and no-ops:
- Do not repeat the same action on the same object consecutive
- Do not act on irrelevant objects (e.g., WASH faucet itself).

R7. Prefer WALK over FIND when IDs are provided.
- Since inputs include object IDs, directly WALK to the target

R8. Never include <character> as an action object.
- The subject is always the character; action arguments must b

Submission checklist (verify before outputting):
- Every acted-on object has a preceding WALK unless NEAR is ex
- Devices to be used are SWITCHON before TYPE/WATCH/other usag
- For washing: WASH occurs before RINSE.
- For items inside containers: OPEN precedes GRAB/PUTIN/PUTBAC
- No duplicate consecutive actions on the same object; stop af
- Output is a pure JSON array with correct action objects and
```

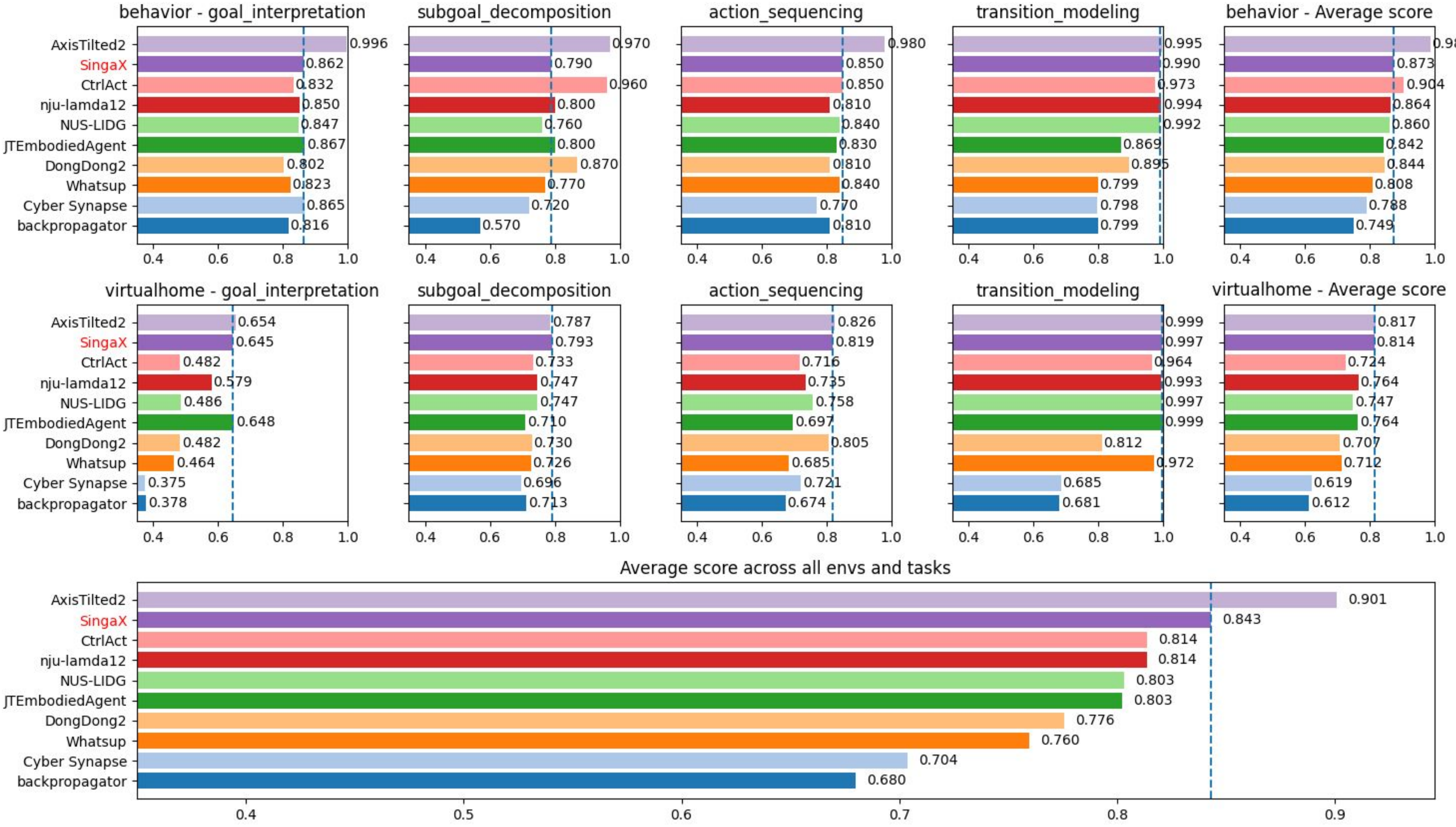
Further improvements (Test set)

	behavior			virtualhome			
	dev/test set			dev set		test set	
	Default prompt	Ours	Ours + edits	Default prompt	Ours	Ours	Ours + edits
goal_interpretation (f1)	79.7	82.3	86.2	43.6	61.1	46.5	64.5
subgoal_decomposition (task sr)	69.0	77.0	79.0	90.5	92.9	72.6	79.3
action_sequencing (task sr)	79.0	84.0	85.0	65.9	77.0	70.8	92.0
transition_modeling (f1)	67.9	84.5	98.9	47.3	81.0	96.0	99.5
transition_modeling (sr)	86.0	98.4	99.0	75.0	97.9	98.5	99.9
avg_perf	76.2	83.7	87.3	65.3	80.1	71.8	81.4

Key strength — Cost effective

- Training-free
- Only require *1* round of query using dev set
- Qwen3-235B-Thinking total cost: *3.75* USD in dev phase, and *17.41* USD in test phase.

Environment	Task	Dev set			Test set		
		Mean token length		Cost for task ($)	Mean token length		Cost for task ($)
		Prompt	Output		Prompt	Output	
behavior	goal_interpretation	1216	5717	0.36	2119	5229	0.35
	subgoal_decomposition	2657	12014	0.75	3683	8851	0.58
	action_sequencing	3415	8188	0.53	3701	7298	0.50
	transition_modeling	3414	10399	0.66	3660	8352	0.57
virtualhome	goal_interpretation	1874	7706	1.65	2994	5513	5.46
	subgoal_decomposition	3056	7225	1.58	3571	8386	8.14
	action_sequencing	2256	5264	1.16	3336	5777	5.75
	transition_modeling	3635	9902	1.88	5398	8681	8.70
Total cost				3.75			17.41



behavior - goal_interpretation
subgoal_decomposition
action_sequencing
transition_modeling
behavior - Average score
AxisTilted2
SingaX
CtrlAct
nju-lamda12
NUS-LIDG
JTEmbodiedAgent
DongDong2
Whatsup
Cyber Synapse
backpropagator
0.996
0.862
0.832
0.850
0.847
0.867
0.802
0.823
0.865
0.816
0.970
0.790
0.960
0.800
0.760
0.800
0.870
0.770
0.720
0.570
0.980
0.850
0.850
0.810
0.840
0.830
0.810
0.840
0.770
0.810
0.995
0.990
0.973
0.994
0.992
0.869
0.895
0.799
0.798
0.799
0.985
0.873
0.904
0.864
0.860
0.842
0.844
0.808
0.788
0.749
virtualhome - goal_interpretation
subgoal_decomposition
action_sequencing
transition_modeling
virtualhome - Average score
0.654
0.645
0.482
0.579
0.486
0.648
0.482
0.464
0.375
0.378
0.787
0.793
0.733
0.747
0.747
0.710
0.730
0.726
0.696
0.713
0.826
0.819
0.716
0.735
0.758
0.697
0.805
0.685
0.721
0.674
0.999
0.997
0.964
0.993
0.997
0.999
0.812
0.972
0.685
0.681
0.817
0.814
0.724
0.764
0.747
0.764
0.707
0.712
0.619
0.612
0.4
0.6
0.8
1.0
Average score across all envs and tasks
0.901
0.843
0.814
0.814
0.803
0.803
0.776
0.760
0.704
0.680
0.4
0.5
0.6
0.7
0.8
0.9

Remaining performance gap in behavior env

- Some problems have specific ground truth
- Cannot be resolved without overfitting to specific problems

```
1  v["all_unsatisfied_conditions"]
✓ 0.0s

[['ontop', 'gym_shoe.n.01_1', 'table.n.02_2'],
 ['ontop', 'necklace.n.01_1', 'table.n.02_2'],
 ['ontop', 'notebook.n.01_1', 'table.n.02_2'],
 ['ontop', 'sock.n.01_1', 'table.n.02_2'],
 ['ontop', 'sock.n.01_2', 'table.n.02_2']]
```

```
1  v["false_positive_conditions"]
✓ 0.0s

[['ontop', 'gym_shoe.n.01_1', 'table.n.02_1'],
 ['ontop', 'necklace.n.01_1', 'table.n.02_1'],
 ['ontop', 'notebook.n.01_1', 'table.n.02_1'],
 ['ontop', 'sock.n.01_1', 'table.n.02_1'],
 ['ontop', 'sock.n.01_2', 'table.n.02_1']]
```

```
Relevant objects in the scene are:
gym_shoe.n.01_1: ['Stained', 'Dusty']
necklace.n.01_1: ['Stained', 'Dusty']
notebook.n.01_1: ['Dusty']
sock.n.01_1: ['Stained', 'Dusty']
sock.n.01_2: ['Stained', 'Dusty']
table.n.02_1: ['Stained', 'Dusty']
table.n.02_2: ['Stained', 'Dusty']
cabinet.n.01_1: ['Stained', 'Open', 'Dusty']
sofa.n.01_1: ['Stained', 'Dusty']
floor.n.01_1: ['Stained', 'Dusty']
floor.n.01_2: ['Stained', 'Dusty']

All initial states in the scene are:
['under', 'gym_shoe.n.01_1', 'table.n.02_1']
['onfloor', 'gym_shoe.n.01_1', 'floor.n.01_2']
['inside', 'necklace.n.01_1', 'cabinet.n.01_1']
['under', 'notebook.n.01_1', 'table.n.02_2']
['ontop', 'sock.n.01_1', 'sofa.n.01_1']
['onfloor', 'sock.n.01_2', 'floor.n.01_1']
['onfloor', 'agent.n.01_1', 'floor.n.01_1']

Task Name and Goal Instructions:
{
    "Task Name": "collect_misplaced_items",
    "Goal Instructions": "Collect all misplaced items including the
gym shoe, necklace, notebook, and socks. Then place them on the
table."
}
```

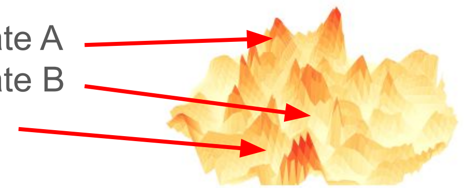
Other stuff we tried

Prompt Optimization

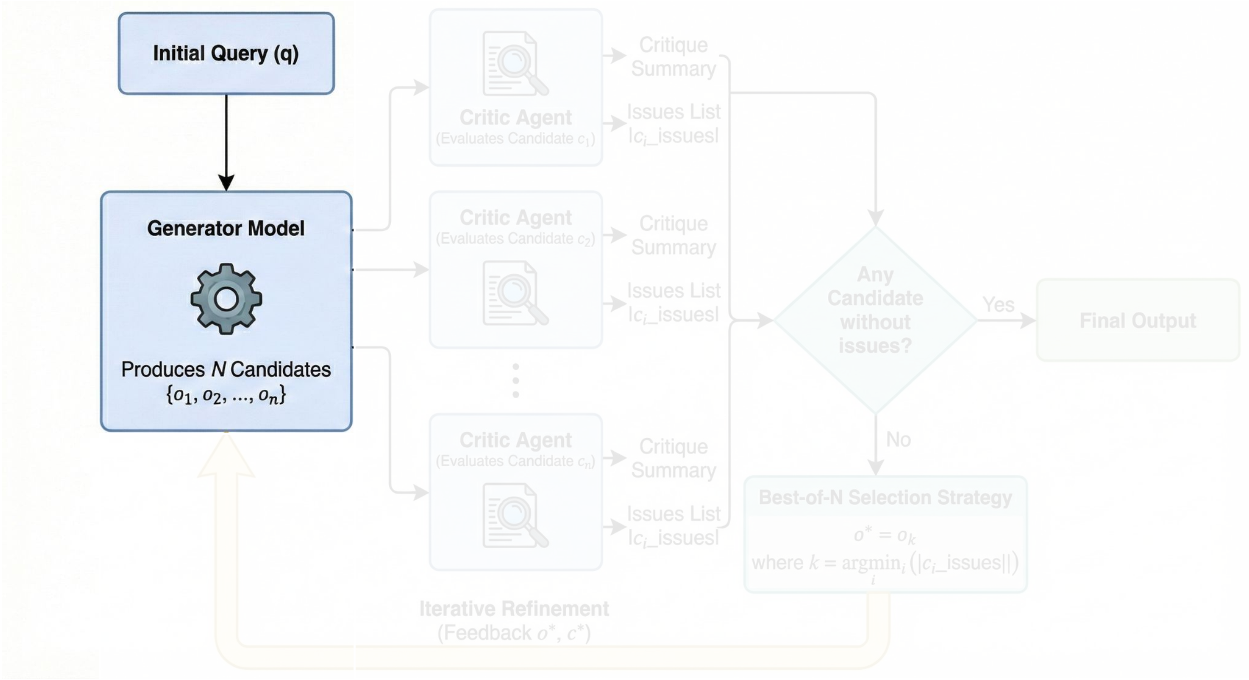
We optimize the prompts used for each task in the following manner:

1. Use the LLM to generate 200 different possible prompt candidates (candidate pool) from the original prompts given by the organizer.
2. Project each prompt into a latent semantic embedding space using an off-the-shelf embedding model.
3. Apply Bayesian Optimization (BO) over the prompt candidates:
   a. Progressively estimates the approximate performance landscape and retrieves the prompt that has the best performance estimate from this landscape.
4. Yields approximately 4-5% improvements.

Prompt candidate A
Prompt candidate B
…



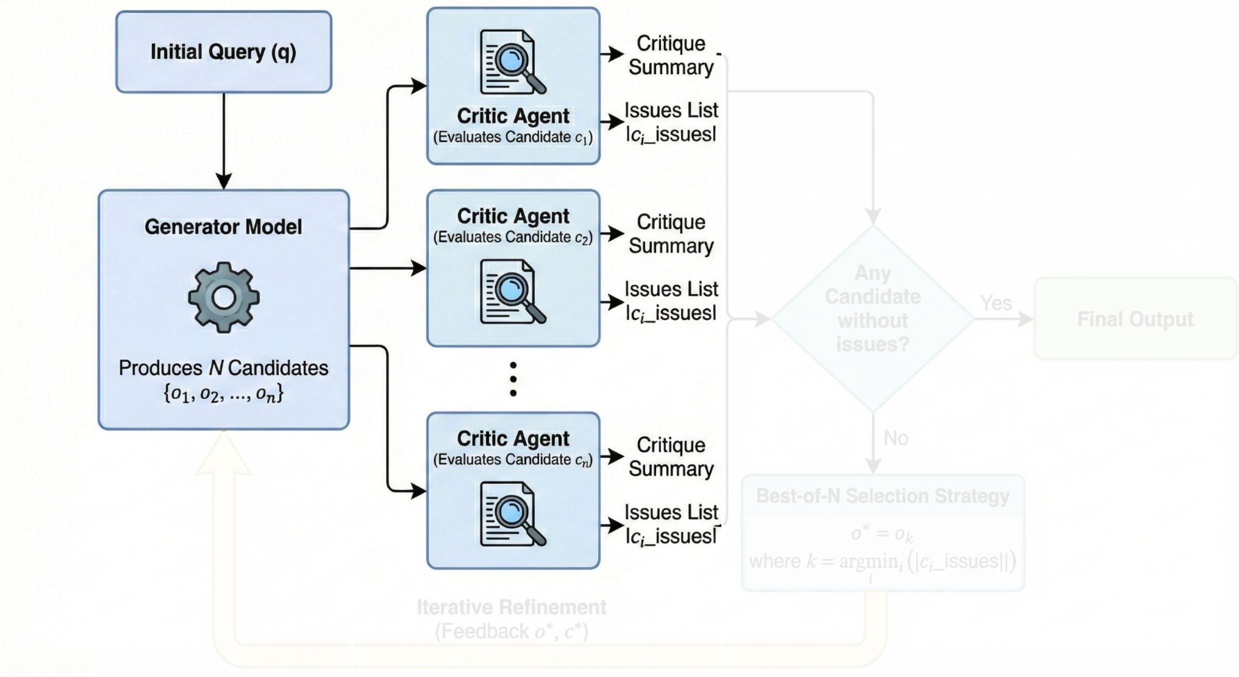
Critic Best-of-N (Critic BoN) Framework



1. Generation Phase:

Given the initial query, the generator model produces *N* candidate outputs.

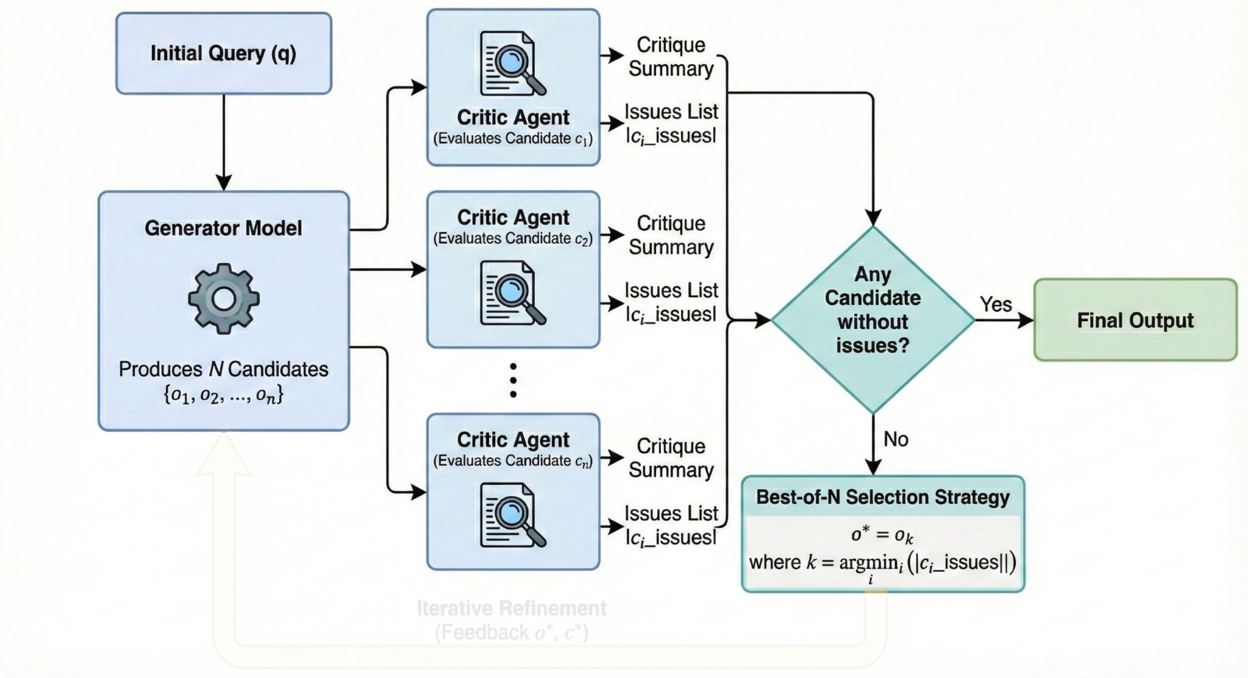
Critic Best-of-N (Critic BoN) Framework



2. Critic Phase:

A Critic Agent checks every candidate against a predefined checklist and outputs any identified issues.

Critic Best-of-N (Critic BoN) Framework

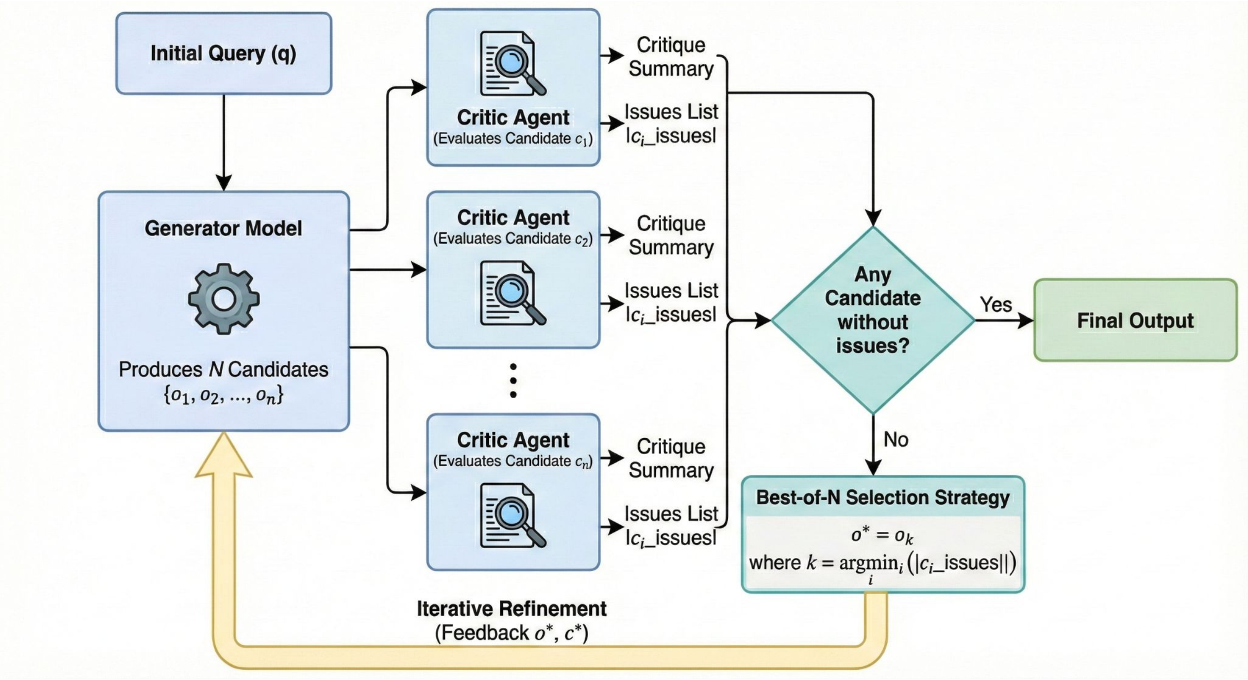


3. Decision Logic:

(a) If a candidate has zero issues, it is accepted as the final output.

(b) If all candidates contain errors, the framework defaults to the Best-of-N strategy.

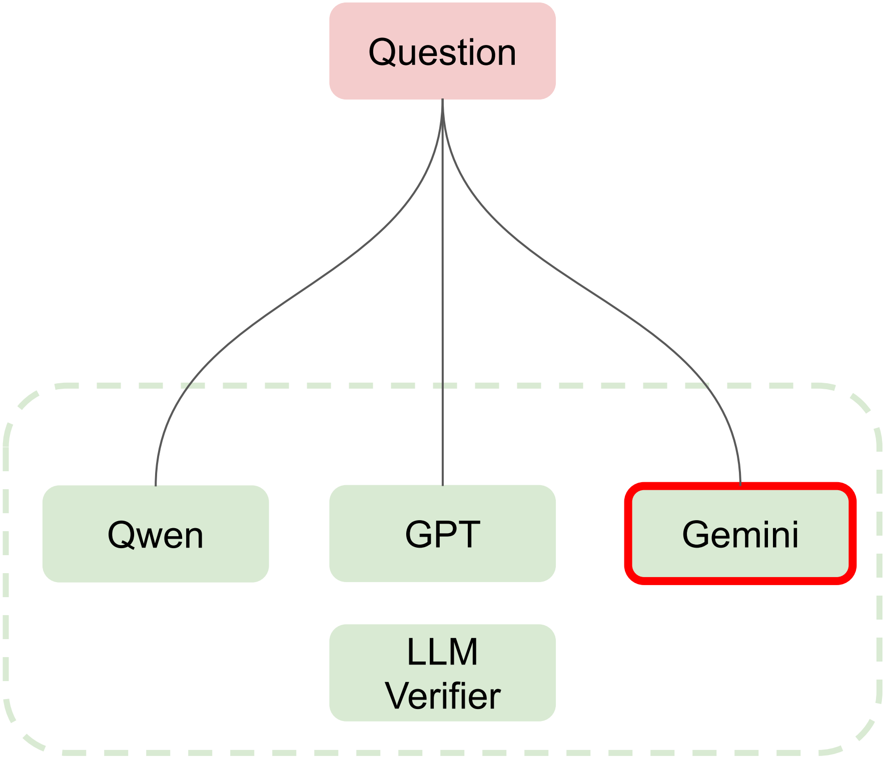
Critic Best-of-N (Critic BoN) Framework



4. Refinement Loop:

The selected candidate and its corresponding critique are provided to the generator model, which then iteratively refines its output based on this feedback.

Multi-Model Best-of-N at Test Time



- Baseline BoN: Sample N answers from one LLM and let a verifier pick the best.
- Multi-Model BoN: For each question, query three heterogeneous LLMs (Qwen, GPT, Gemini) and treat their outputs as N=3 candidates.
- Verifier selection: An LLM verifier scores candidates on JSON format, constraint satisfaction, and task quality, then selects the best answer.
- Benefit: Better robustness and accuracy from model diversity, with no extra training.

Thank you for your time!