DynaAct: Large Language Model Reasoning with Dynamic Action Spaces

Xueliang Zhao[1,2], Wei Wu[2], Jian Guan[2], Qintong Li[1], Lingpeng Kong[1]
[1] The University of Hong Kong [2] Ant Group

Source Code:
https://github.com/zhaoxlpku/DynaAct
Scan QR ->

Introduction

Limitations of Current TTS Frameworks

- ✓ **Static Action Spaces:** Most methods rely on fixed or hand-crafted actions, limiting adaptability across tasks and domains.
- ✓ **Costly Search:** Large, unfiltered action spaces cause **slow and inefficient exploration** during test-time reasoning.
- ✓ **Weak Action Utility:** Selected actions often fail to trigger **key reasoning steps**, yielding limited gains on hard benchmarks.
- ✓ **Scaling Bottleneck:** Enlarging the action space typically increases computation exponentially, restricting real-world use.

Our Contributions: DynaAct

- ✓ **Dynamic Action Construction:** Builds **context-aware, compact** action spaces via a submodular objective balancing **utility + diversity**.
- ✓ **Efficient High-Utility Reasoning:** Reduces millions of candidate actions to a small, high-impact set—**faster search, stronger reasoning**.
- ✓ **State-of-the-Art Performance:** Outperforms RAP, rStar, and few-shot/fine-tuned baselines on **MMLU, GPQA, GSM8K, and MATH-500**.
- ✓ **Plug-and-Play Design:** Backbone-agnostic and compatible with any search method (i.e., MCTS), enabling future scaling.

Method

Proxy Action Space Estimation

- ✓ Extract observation-style actions from diverse reasoning traces to form a **large proxy action pool $\mathcal{A}$**.
- ✓ Train a lightweight **embedding model** using Q-learning over demonstrations to encode action utility.

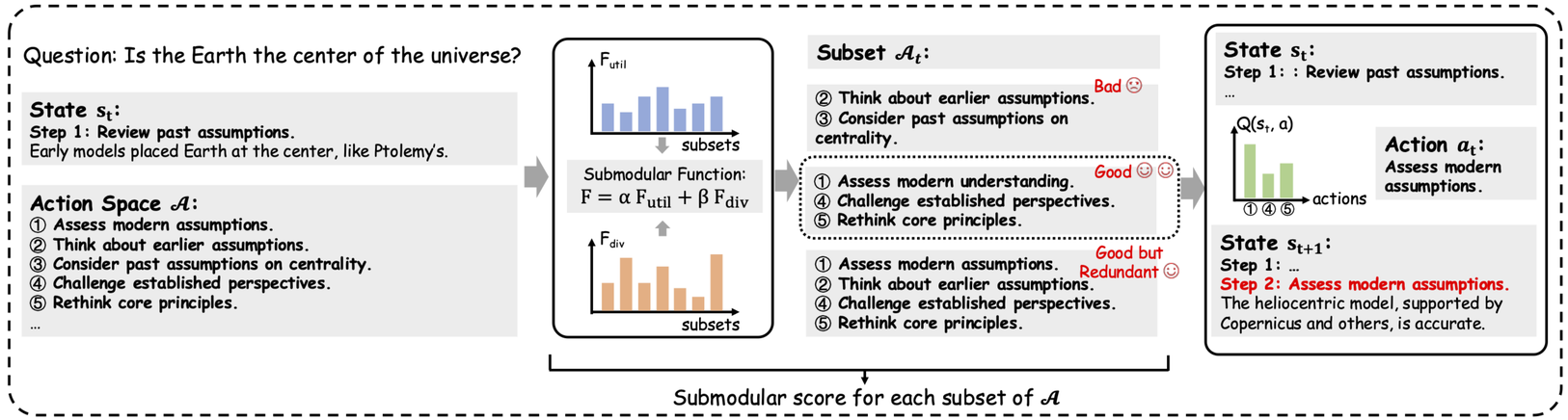
Submodular Action Selection

- ✓ Define a submodular score $F = \alpha f_{\text{util}} + \beta f_{\text{div}}$, jointly measuring relevance to the current state and diversity.
- ✓ Apply a **greedy maximization** procedure to select a compact candidate set $\mathcal{A}_t$ of size m for each step.
- ✓ Ensures **utility-focused yet non-redundant** actions with linear-time selection.

Reasoning with Dynamic Action Spaces

- ✓ Evaluate actions in $\mathcal{A}_t$ via Monte-Carlo Tree Search (MCTS) to estimate $Q(s_t, a)$.

➢ Overview of the proposed method



➢ Experiments

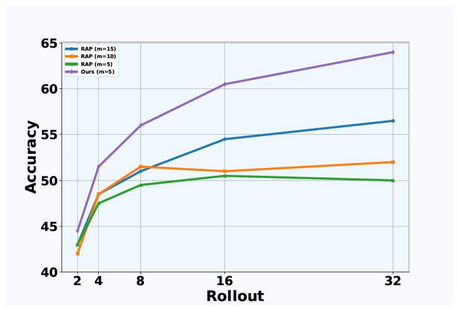
➢Main Results

Model	General		Reasoning		Math	
	MMLU	MMLU-Pro	GPQA	ARC-C	GSM8K	MATH-500
Zero-shot CoT	68.87	43.45	31.82	81.06	76.12	45.40
SC@maj16	69.66	49.36	34.34	80.63	86.66	52.00
RAP	69.46	48.70	38.89	85.41	87.79	51.60
rStar	68.61	48.81	36.87	86.43	87.11	54.20
DynaAct	**70.22**	**51.40**	**39.39**	**88.31**	**89.16**	**61.00**

➢Ablation Study

Model	ARC-C	MATH-500
DynaAct (full)	**88.31**	**61.00**
- util	87.63	53.40
- div	86.52	53.80
- q-learning	87.80	55.80
- submodular	85.15	52.00

➢Compactness Study



➢Reasoning Performance Across Difficulty

	Level 3	Level 4	Level 5
rStar	72.38	50.78	15.67
DynaAct	**76.19**	**58.59**	**31.34**
- util	68.57	52.34	17.16
- q-learning	71.43	53.13	20.90

➢Action Diversity Analysis

Model	Diversity	Accuracy
Ours	0.73	31.34
- div	0.49	24.63

➢Resource Consumption Analysis

Method	Raw Time (↓)	Accuracy (↑)
Zero-shot CoT	1.68s	45.40
SC@maj16	26.88s	52.00
RAP	64.51s	51.60
rStar	54.72s	54.20
DynaAct	57.60s	**61.00**

For inquiries, please contact:
xlzhao22@connect.hku.hk