Craft Jarvis

November 13, 2023

JARVIS-1: Open-world Multi-task Agents with Memory-Augmented Multimodal Language Models

Zihao Wang[1,2], Shaofei Cai[1,2], Anji Liu[3], Yonggang Jin[4], Jinbing Hou[4], Bowei Zhang[5], Haowei Lin[1,2], Zhaofeng He[4], Zilong Zheng[6], Yaodong Yang[1], Xiaojian Ma[6†], Yitao Liang[1†]

Presented by:

John Tan Chong Min

Verdict:
- Cool use of images for multimodal planning
- Needs better planning
- Needs better learning for controller
- Needs better memory encoding and retrieval

Previously on Voyager…

- Curriculum generator, skill learning, skill reuse

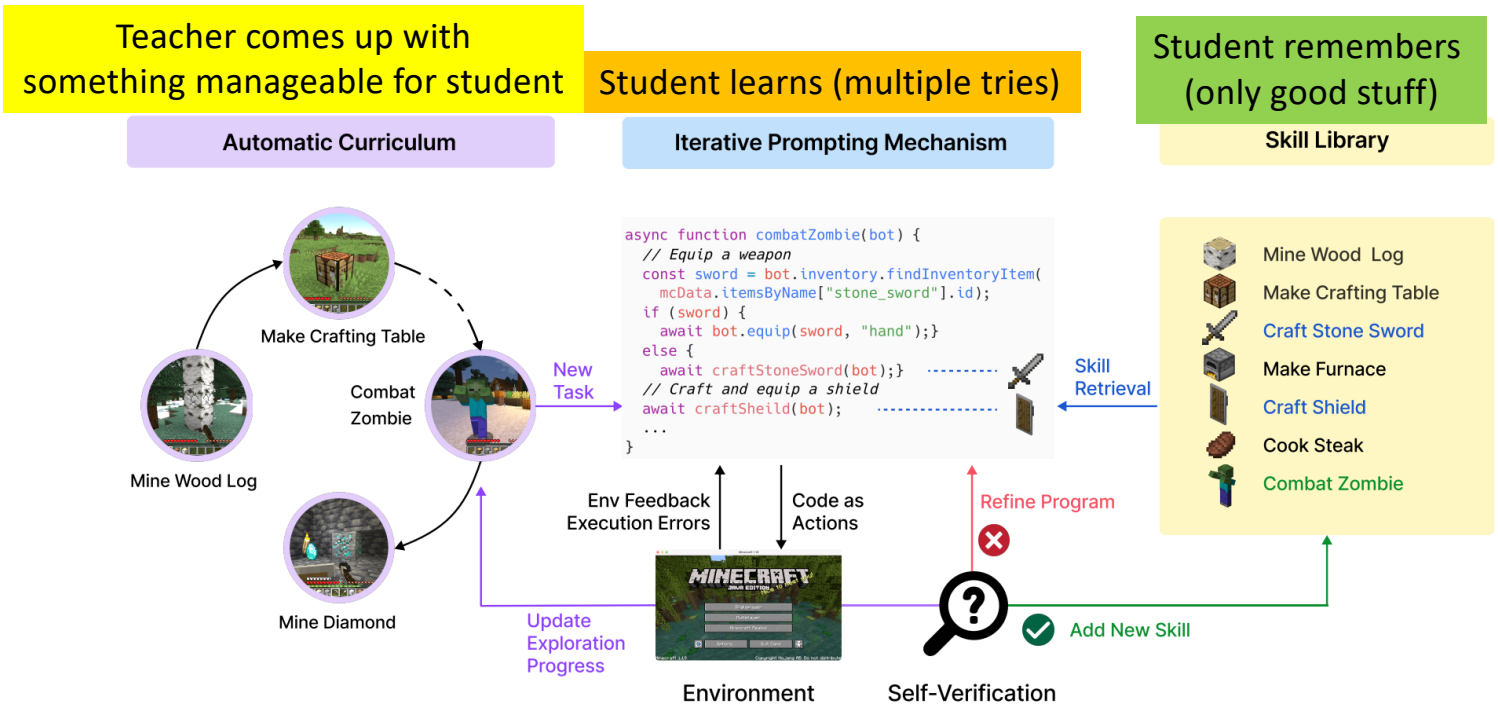
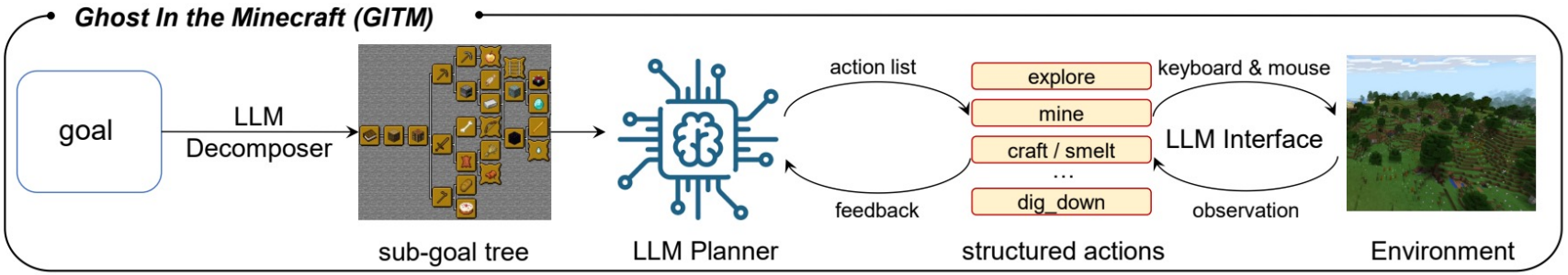


Figure 2: VOYAGER consists of three key components: an automatic curriculum for open-ended exploration, a skill library for increasingly complex behaviors, and an iterative prompting mechanism that uses code as action space.

Voyager. Wang et al. 2023.

Previously on Ghost in the MineCraft…

- Decompose into sub-goals, plan based on memory



Action Implementation. The observation of the action contains LiDAR rays with an interval of 5 degrees in the horizon and vertical direction for locating objects, and voxels with 10 units radius only for navigation, inventory, life status, and agent location status (X-ray cheating is carefully avoided).

Ghost in the MineCraft. Zhu et al. 2023.

Previously on Ghost in the MineCraft…

- Close to 50% diamond pickaxe success rate

- Better than 12.5% in this paper, but… this is using vision rather than LiDAR rays (which may help simplify the environment)

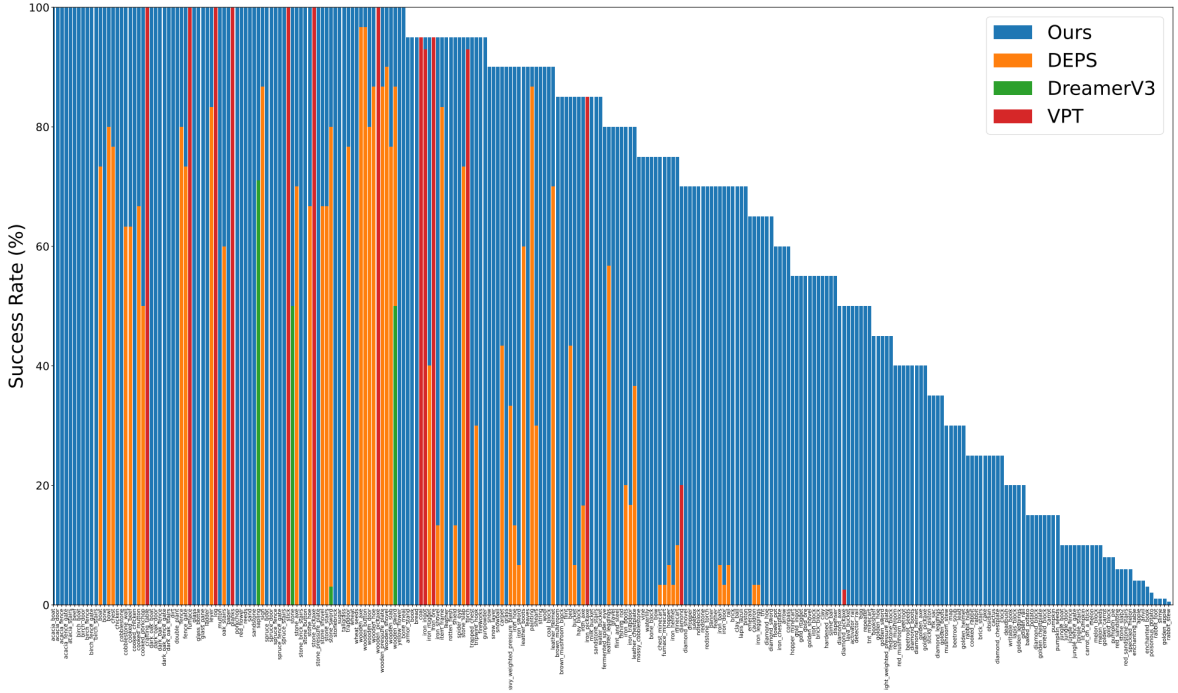


Figure 5: **Success rate for all items in the entire Minecraft Overworld Technology Tree**. The x axis lists all item names. We overlay the results from our GITM and the best results from baselines.

Ghost in the MineCraft. Zhu et al. 2023.

JARVIS-1

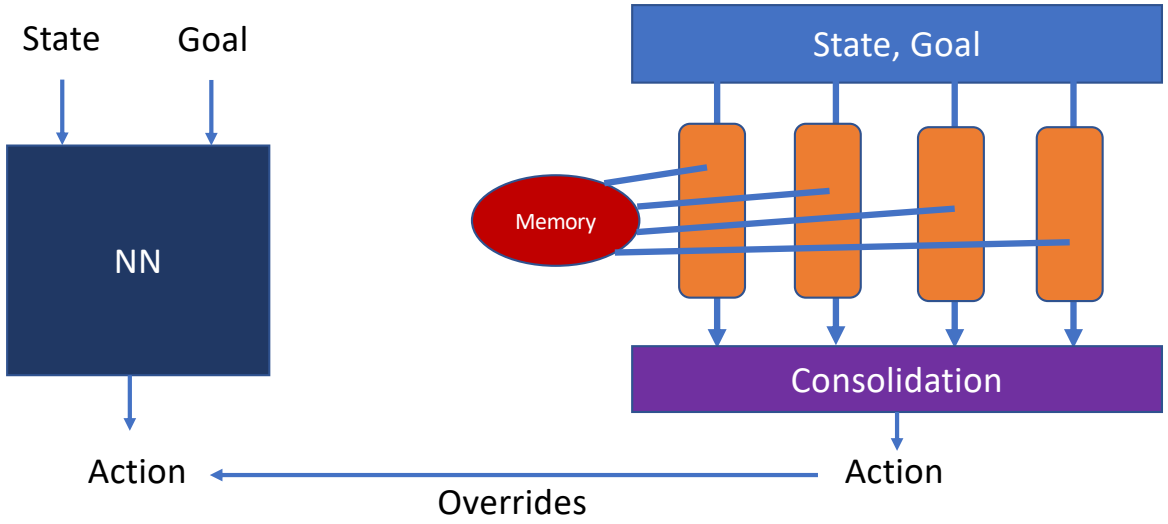
- JARVIS-1 built on top of **pre-trained multimodal language models**, which map visual observations and textual instructions to plans
- Plans will be ultimately dispatched to the **goal-conditioned** controllers

- JARVIS-1 has a **multimodal memory**, which facilitates planning using both pre-trained knowledge and its actual game survival experiences.
- JARVIS-1 is able to **self-improve** following a life-long learning paradigm thanks to multimodal memory

- **No structured functions needed to interact with MineCraft, only native keyboard and mouse interactions (huge improvement compared to Voyager and GiTM!)**

Two Networks – Fast and Slow

- Memory is important for fast adaptation before neural networks learn



Neural Networks: Fast retrieval, slow learning

Memory: Slow retrieval, fast learning
(World Model planning as Memory Retrieval)

Learning, Fast and Slow: A Goal-Directed Memory-Based Approach for Dynamic Environments. John and Motani. 2023.

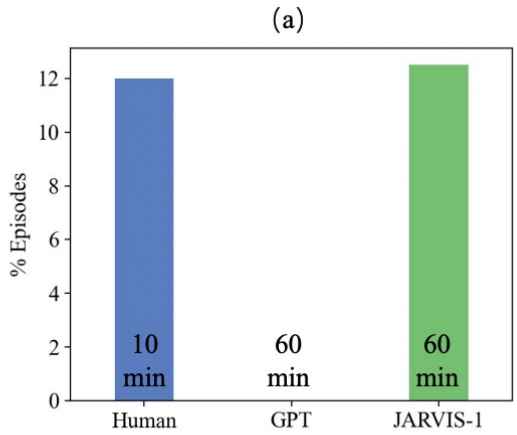
Unlocking Entire Technology Tree

- More than Voyager
- Similar to Ghost in the MineCraft

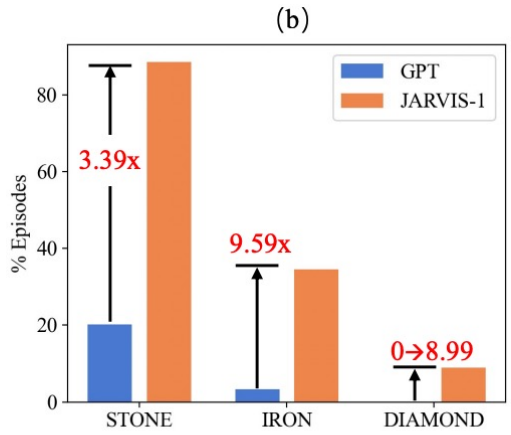
Situation-aware Planning is important!

- A GPT-based planner that produces plans only at the beginning without looking at the current situation, the agent failed to complete the task as opposed to human players and JARVIS-1, which perform situation-aware planning from time to time

- *Note: This is not new. Voyager and GiTM also do this by feedbacking error message from game back into LLM.*



Success Rate on ObtainDiamond Task



Success Rate on Crafting Stone, Iron, Diamond Tasks

Self-Improvement via memory

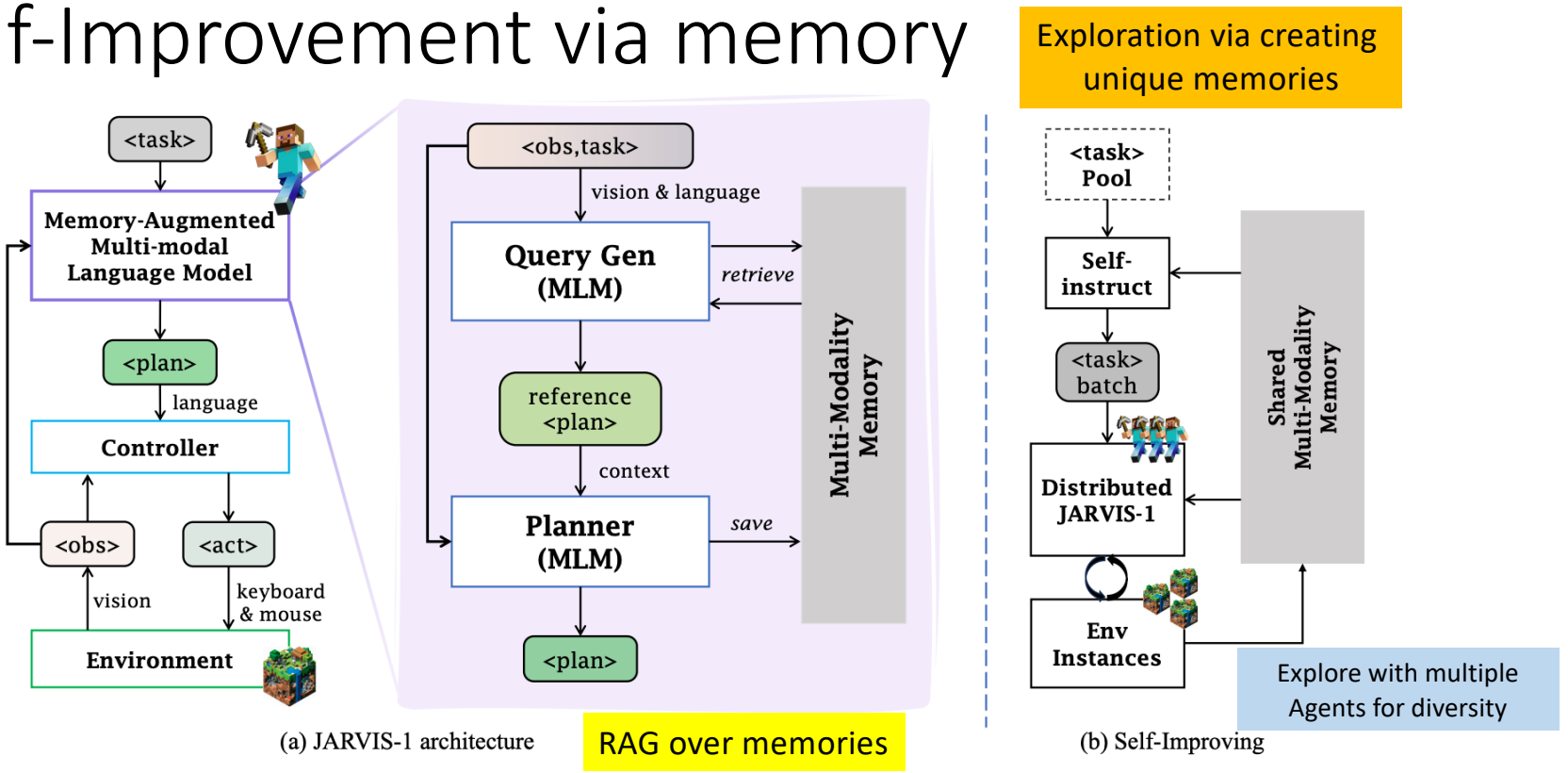


Figure 3: **Architecture of JARVIS-1 and its self-improving mechanism**. (a) JARVIS-1 comprises a memory-augmented multimodal language model (MLM) that produces plans and a low-level action controller. JARVIS-1 also utilizes a multimodal memory to store and obtain experiences as references for planning. (b) JARVIS-1 can strengthen its own planning skills through exploration with its own proposed tasks (*self-instruct*) and a growing memory that helps with better planning on tasks that has been (partially) visited before.

Observational Space

- No LiDAR or X-ray vision, just normal RGB pixels from perspective of a human playing the game!

Table 5: The observation space we use in Minecraft.

Sources	Shape	Description
pov	(640, 360, 3)	Ego-centric RGB frames.
player_pos	(5,)	The coordinates of (x,y,z), pitch, and yaw of the agent.
location_stats	(9,)	The environmental information of the agent's current position, including `biome_id`, `sea_level`, `can_see_sky`, `is_raining` etc.
inventory	(36,)	The items in the current inventory of the agent, including the `type` and corresponding `quantity` of each item in each slot. If there is no item, it will be displayed as `air`.
equipped_items	(6,)	The current equipment of the agent, including `mainhand`, `offhand`, `chest`, `feet`, `head`, and `legs` slots. Each slot contains `type`, `damage`, and `max_damage` information.
event_info	(5,)	The events that occur in the current step of the game, including `pick_up` (picking up items), `break_item` (breaking items), `craft_item` (crafting items using a crafting table or crafting grid), `mine_block` (mining blocks by suitable tools), and `kill_entity` (killing game mobs).

Multimodal to Unimodal: Image Observation to Text (Constrained)

- As opposed to letting the MLM caption the scene directly, first extract keywords of Minecraft items (e.g., "acacia tree", "sheep") from Minecraft wiki and utilise GPT to generate sentences that describe these observations
- For example, a generated sentence could be "I can see sheep in the acacia plains"
- Additional situation details including biome and inventory status are also converted into text using templates
- Finally, prompt the MLM again (the language part only) into a plan given the task instruction and all the aforementioned textual situation descriptions
- Constrained LLM text generation is better than full end-to-end
  - Limitations of image modality captioning

Breaking into sub-tasks, use memory to solve

- Similar to GiTM
- But all sub-goals retrieved at the same time is bad for retrieval!

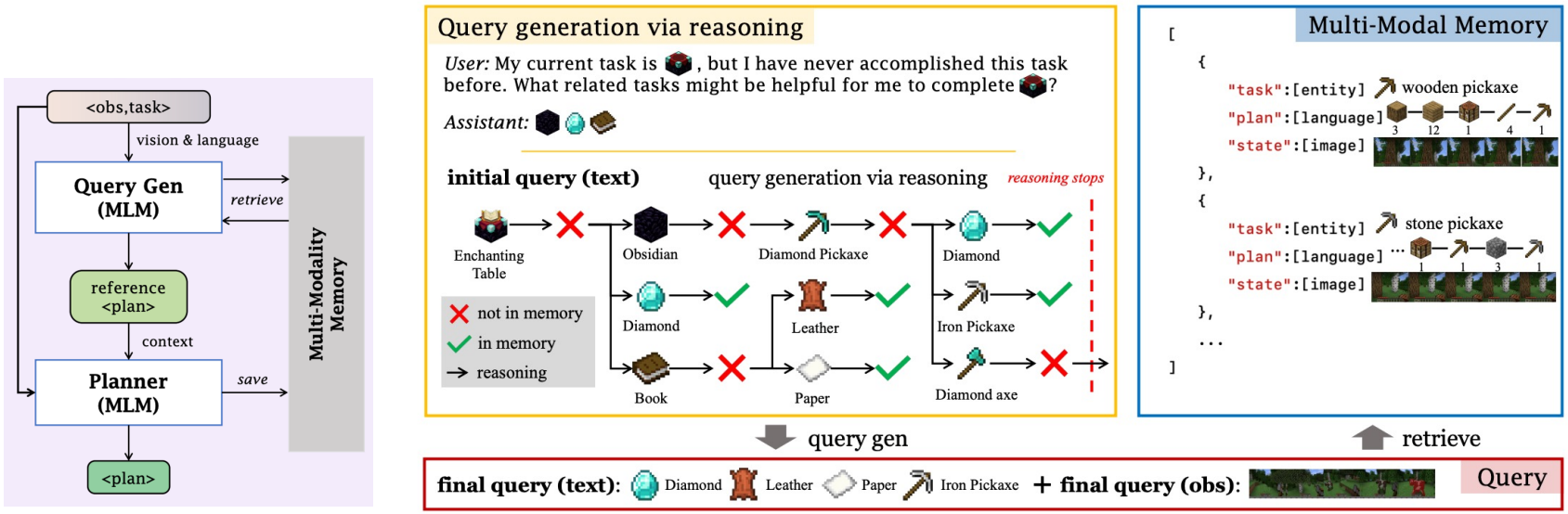
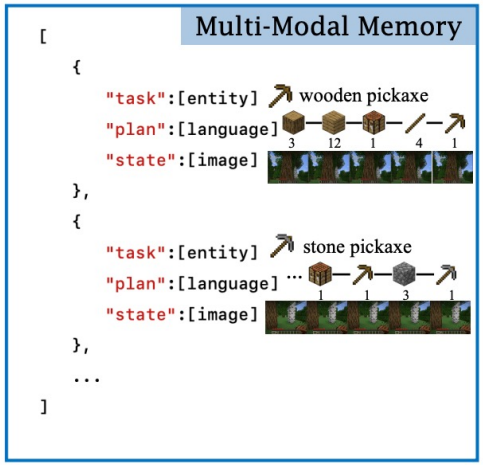


Figure 5: **Query generation in JARVIS-1**. Given the current observation and the task, JARVIS-1 will first think backward and figure out the needed intermediate sub-goals. The reasoning will be bounded by a limited depth. The sub-goal that is present in the memory will join the current visual observation to form the final query. Entries that match the text query will be ranked by the perceiving distance of their states to the obs query and only the top entry of each sub-goal will be retrieved.

Storing the memory

- Records every successful trajectory experience:
  - **Task goals** that the agent needs to execute
  - **Plan** executed by the agent
  - **State** (visual observation and symbolic information returned from the environment) when the agent completes the task

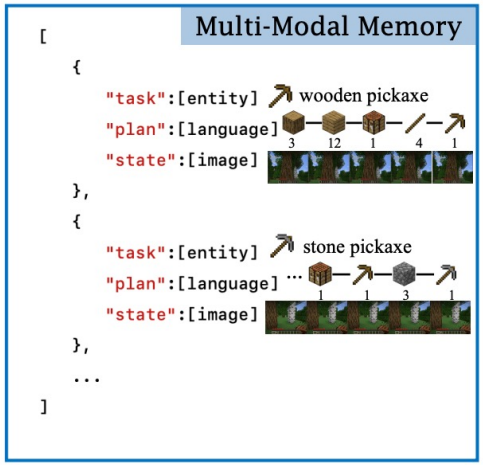
- **My take: Inefficient!**
  - Why not also make any end point a goal?
  - Why not make any point along the trajectory a start point?
  - Different environmental states along trajectory are not stored – missing out on lots of learning experiences
  - We should store transitions instead of full plans, so we can mix and match (but maybe MineCraft is too structured)



Retrieving the memory

- Use CLIP (Text) embeddings between text query and “task” memory to get memories above a certain similarity score
- Then, use CLIP (Image) embeddings between query image and “state” image memory to choose the top-k memory
- Retrieve the plan (value) from the memory
- **TLDR: Sort by task relevance first, then sort by current state**

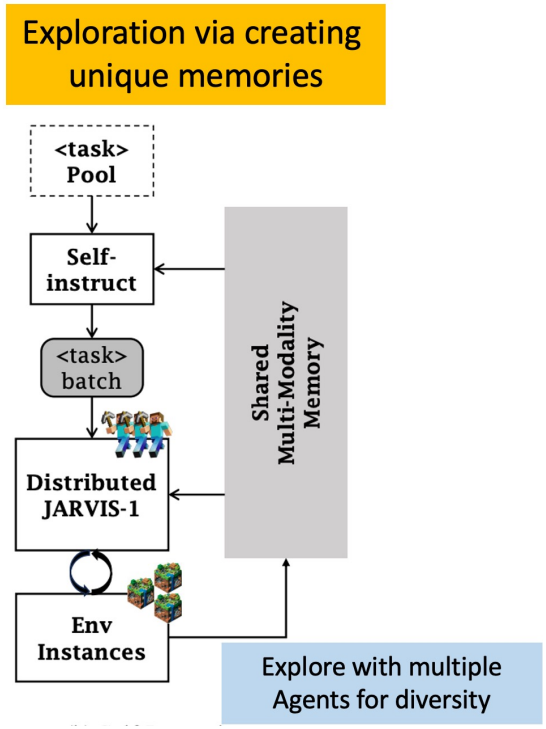
Key + Value



Query

Generating the memories

- Bootstrapping off **task-oriented exploration**:
  - Choose set of tasks from task pool
  - Use a population-based approach to gather experience performing the tasks in various environments
  - Shared memory means all agents will learn the plan from solved tasks (similar to GiTM; different from Voyager learning the code for task)
- **Lifelong learning**: Keep updating memory as agent performs tasks in environment
- Questions to ponder
  - Should each agent be more varied with unique memory?
  - Should we override memories with better ones?



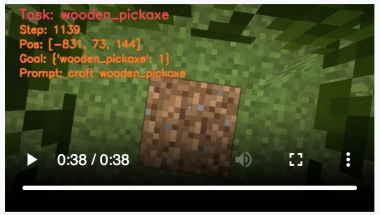
Goal-directed learning with diversity: Executes Goal in various Biomes

Intruction-Following in Diverse Biomes

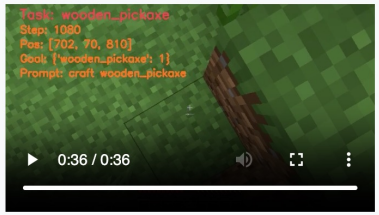
JARVIS-1 can execute human instructions in diverse environments. We illustrate executions in different biomes below.

Pick up a Minecraft:wooden_pickaxe

Execution in Plains:

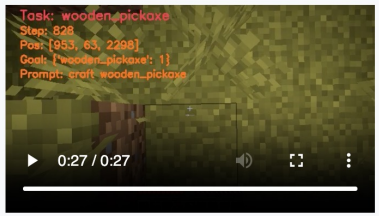


Execution in Birch Forest:



Execution in Jungle:

Execution in Savanna:



Improving via Reflection and Error Feedback

- Reflecting on plan via self-check is new for MineCraft agents and is similar to Reflexion

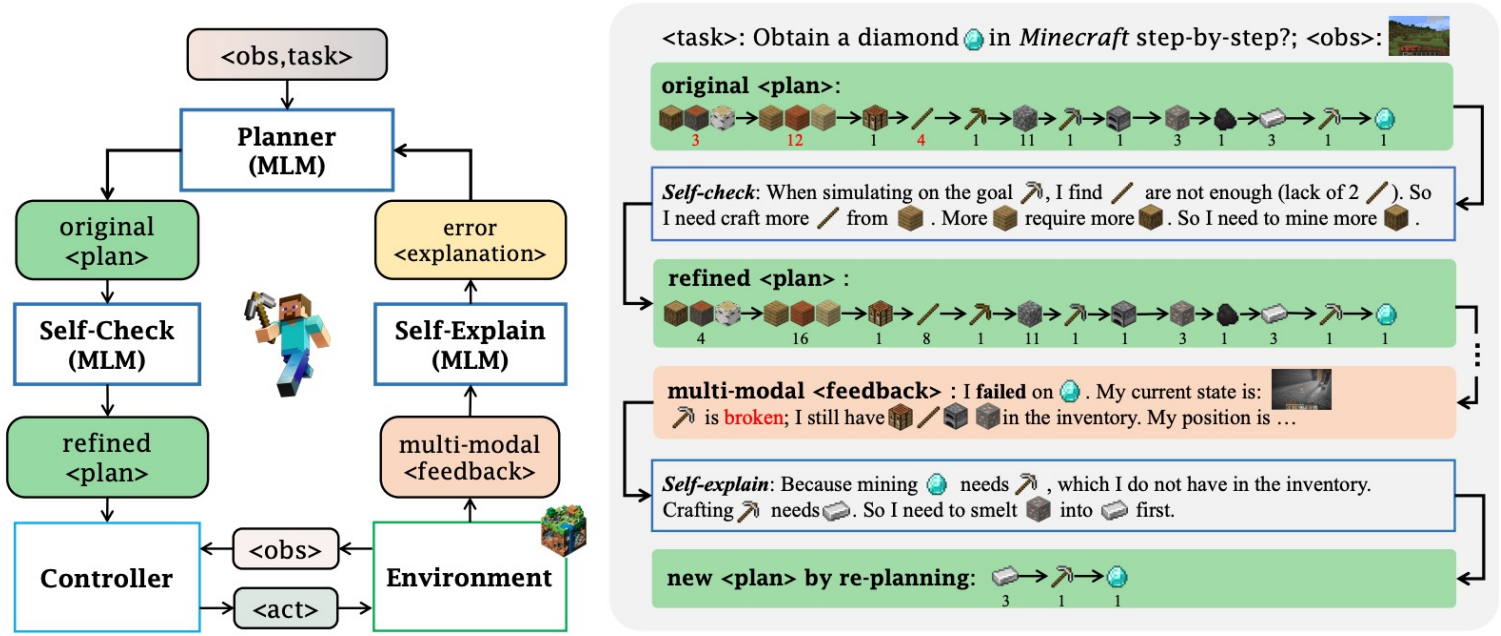


Figure 4: **Interactive planning in JARVIS-1**. After receiving the current task instruction and observation, JARVIS-1 will produce an initial plan, which will go through *self-check* to get possible bugs (marked in red) fixed. Further, in case any error (also marked in red) occurs during the execution of the refined plan, JARVIS-1 will try to reason about the next move from the environmental feedback via *self-explain*. Interleaving self-check and self-explain significantly boosts the correctness and robustness of JARVIS-1 planning.

Performance

- In the experiments, JARVIS-1 exhibits nearly perfect performances across over 200 varying tasks from the Minecraft Universe Benchmark, ranging from entry to intermediate levels.

Table 1: Characteristics of 11 task groups encompassing over 200 minecraft tasks.

Group	Task Num.	Max. Steps	Initial Inventory	Biome	Language Instruction
Wood	34	12k	null	Plains/Forest	Pick up a `wooden_pickaxe`.
Wood-Variants	43	12k	null	Savanna/Jungle/Taiga	Pick up a `acacia_boat`.
Stone	10	12k	iron_axe	Plains/Forest	Craft a `furnace` given an iron axe.
Iron	22	12k	iron_axe	Plains/Forest	Smelt and craft an `iron_door` given an iron axe.
Gold	9	36k	iron_axe	Plains/Forest	Smelt and craft an `golden_axe` given an iron axe.
Diamond	7	36k	iron_axe	Plains/Forest	Dig down to mine diamond and craft `diamond_pickaxe`.
Redstone	7	36k	iron_axe	Plains/Forest	Mine redstone and make `dropper` given an iron axe.
Blocks	15	12-36k	iron_axe	Plains/Forest	Dig down to mine `lapis_lazuli` block.
Armor	17	12-36k	iron_axe	Plains/Forest	Craft `diamond_boots` given an iron axe and equip it.
Decoration	17	12k	iron_axe	Flower Forest	Obtain the `bed` and dye it `red`.
Food	9	12k	iron_axe	Plains	Kill sheep to obtain `mutton` and cook it.

Results (Abridged)

Table 2: Results of JARVIS-1 and baselines on Minecraft. The detailed task instructions, settings and results can be found in the Appendix.

Group	Task	GPT	ReAct	Inner Monologue	DEPS	JARVIS-1
Wood		26.67	45.00	36.67	75.00	**91.55**
	AVG	27.30±14.86	40.31±13.30	60.15±19.41	80.23±17.32	**88.84±16.82**
Wood Var		6.67	36.67	30.00	36.67	**60.47**
	AVG	24.39±11.08	38.13±12.81	53.39±12.86	68.75±12.32	**76.78±12.27**
Stone		20.00	20.00	66.67	75.00	**94.20**
	AVG	20.21±12.32	39.00±12.15	52.86±16.90	69.27±7.78	**88.69±4.87**
Iron		0.00	0.00	3.33	20.00	**33.82**
		3.33	6.67	0.00	20.00	**38.10**
	AVG	3.27±2.85	4.61±3.63	5.20±5.17	16.92±4.69	**34.63±10.61**
Gold		0.00	2.00	2.00	6.00	**14.49**
	AVG	0.00±0.00	0.45±0.60	0.59±0.64	2.20±1.55	**6.85±4.71**
Diamond		0.00	0.00	1.00	2.00	**9.20**
		0.00	0.00	0.00	2.50	**6.22**
	AVG	0.00±0.00	0.35±0.48	0.96±0.67	2.42±1.01	**8.99±2.68**

- Better than GPT and ReAct
- May not be as good as Voyager and GiTM (but hard to compare as no structured code used)

Memory is important

Table 4: **Success rates for memory ablation on Minecraft tasks.** **R** is reasoning process, **T** and **M** represent retrieve with text embedding and multi-modal embedding, respectively.

Memory	Retrieval					
-	-	0.85	0.00	0.05	0.00	0.05
✓	T	0.85	0.05	0.10	0.00	0.10
✓	T+R	0.95	0.25	0.30	0.05	0.20
✓	M+R	0.94	0.34	0.40	0.09	0.24

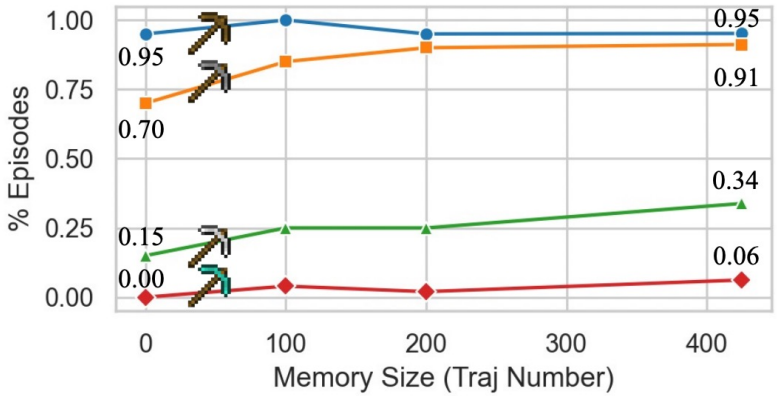
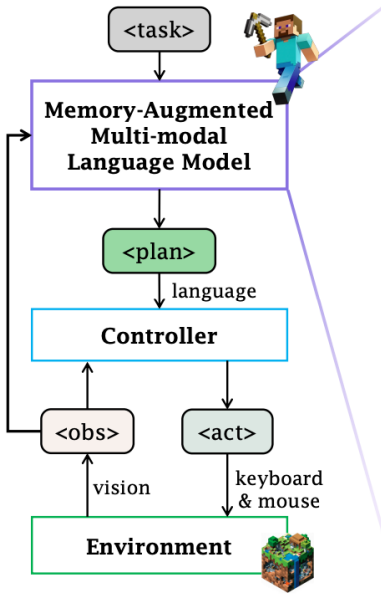


Figure 6: **Success rate by memory size for different items**. We evaluated the performance of JARVIS-1 at different memory sizes (representing different learning stages) by measuring the success rate (% Episodes) of completing key items on the Minecraft technology tree. As the learning progressed, we observed an improvement in completion rates for all items, with an increasing number of successful trajectories being included in memory. After 4 epochs of learning, JARVIS-1 had accumulated a total of 425 successful trajectories in its memory.

Controller is the weakest link!



- The bottleneck for JARVIS-1 in tasks involving diamonds often lies with the Controller's inability to perfectly execute short horizon text instructions generated by LLM
- Controller is goal-conditioned, maps goal and current state to low-level actions
- **How is controller trained? Not in paper!!!**
- Will be interesting to see JARVIS-1's performance given a perfect controller, such as those developed using structured means like Voyager and GiTM

Questions to Ponder (1/2)

- If we wanted to fine-tune the multimodal LLM with the agent's experience, as is what is proposed in Fast & Slow, how can we do it?

- In the task pool for self-instruct, there is still a ”ground truth” list of tasks to choose from. How can we do learn without this?

- Right now, the image processing is done by converting it to text, then passing it through the LLM. Are there ways to process it directly in the image domain? Will they be better than using text?

Questions to Ponder (2/2)

- Should we share all experiences with all agents? Or only a select few? How about diversity between agents by having different memory in each one?

- Should we do lookahead planning to determine which memory trajectory to use?

- Could memory soup fit into this by encoding long and short-term horizon memory?