DoraemonGPT: Toward Understanding Dynamic Scenes with Large Language Models (Exemplified as A Video Agent)

Zongxin Yang [1] **Guikun Chen** [1] **Xiaodi Li** [1] **Wenguan Wang** [1] **Yi Yang** [1]

Abstract

Recent LLM-driven visual agents mainly focus on solving image-based tasks, which limits their ability to understand dynamic scenes, making it far from real-life applications like guiding students in laboratory experiments and identifying their mistakes. Hence, this paper explores DoraemonGPT, a comprehensive and conceptually elegant system driven by LLMs to understand dynamic scenes. Considering the video modality better reflects the ever-changing nature of real-world scenarios, we exemplify DoraemonGPT as a video agent. Given a video with a question/task, DoraemonGPT begins by converting the input video into a symbolic memory that stores *task-related* attributes. This structured representation allows for *spatial-temporal* querying and reasoning by well-designed sub-task tools, resulting in concise intermediate results. Recognizing that LLMs have limited internal knowledge when it comes to specialized domains (*e.g.*, analyzing the scientific principles underlying experiments), we incorporate plug-and-play tools to assess external knowledge and address tasks across different domains. Moreover, a novel LLM-driven planner based on Monte Carlo Tree Search is introduced to explore the large planning space for scheduling various tools. The planner iteratively finds feasible solutions by backpropagating the result's reward, and multiple solutions can be summarized into an improved final answer. We extensively evaluate DoraemonGPT's effectiveness on three benchmarks and several in-the-wild scenarios. Project page: https://z-x-yang.github.io/doraemon-gpt.

[1]ReLER, CCAI, Zhejiang University, Hangzhou, China. Correspondence to: Yi Yang <yangyics@zju.edu.cn>.

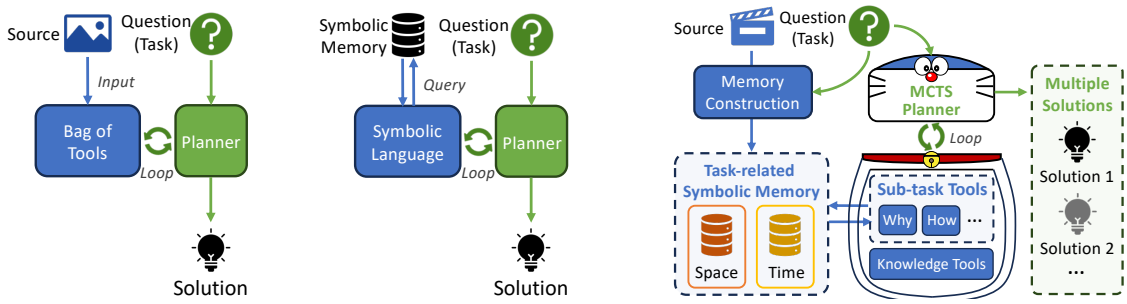
Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

1. Introduction

Based on the advancements in large language models (LLMs) (OpenAI, 2021; Anil et al., 2023; Touvron et al., 2023; OpenAI, 2023; Chiang et al., 2023), recent LLM-driven agents (Surís et al., 2023; Shen et al., 2023; Gupta & Kembhavi, 2023) have demonstrated promise in decomposing a complex image task into manageable subtasks and solving them step-by-step. While static images have been extensively studied, real-world environments are inherently dynamic (Wu et al., 2021) and ever-changing (Smith et al., 2022). Commonly, capturing dynamic scenes is a data-intensive procedure, usually processed by streaming static images into videos. In turn, the *spatial-temporal* reasoning of videos is critical in real-life recognition, semantic description, causal reasoning, *etc*.

Toward understanding dynamic scenes, developing LLM-driven agents to handle videos is of great significance yet involves grand challenges: **i) Spatial-temporal Reasoning**. Reasoning about the relationships of instances is crucial for task decomposing and decision-making. Such relationships may be relevant to space (Santoro et al., 2017), time (Zhou et al., 2018), or their *spatial-temporal* combination. **ii) Larger Planning Space**. Compared to the image modality, high-level semantics about actions and their intentions can typically only be inferred from temporal visual observations (Tapaswi et al., 2016). In other words, the inference of temporal semantics is necessary and will enlarge the search space of decomposing dynamic video tasks. **iii) Limited Internal Knowledge**. LLMs can not encode all the knowledge required for understanding every video due to the ever-changing nature of the real world and/or the lack of learning on proprietary datasets (Peng et al., 2023a).

In light of the preceding discussions, we present DoraemonGPT, an intuitive yet versatile LLM-driven system compatible with various foundation models and applications. Exemplified as a video agent, DoraemonGPT has three desirable abilities: **First**, collecting information regarding the given task before reasoning. In DoraemonGPT, the decomposition of the given dynamic task is decided by the agent-based reasoning of *spatial-temporal* relations, which are inferred from informative attributes, such as instance



(a) Naïve planners (b) *w/* symbolic memory (c) DoraemonGPT (ours)

Figure 1. (a) Naïve LLM-driven planners (*e.g.*, Shen et al. (2023)) decompose a static image task to a sequence of tool executions, while real-world environments are inherently dynamic. (b) Planners with symbolic memory (*e.g.*, Peng et al. (2023a); Hu et al. (2023)) generate symbolic languages to retrieve external knowledge. (c) Given a dynamic video with a task, our DoraemonGPT (§2) decouples spatial-temporal attributes into task-related memories. Instead of generating symbolic languages directly, a bunch of sub-task (symbolic) tools for different spatial-temporal reasoning and other tools (*e.g.*, for retrieving external knowledge) are planned to solve the task. With the MCTS planner, DoraemonGPT can explore large planning spaces, find potential solutions, and deliver an improved final answer.

locations, actions, scene changes, *etc*. However, it is important to note that only task-solving related information is critical, as gathering excessive context tends to hinder the LLMs' capability (Shi et al., 2023). **Second**, exploring better solutions before making decisions. LLM-driven planning (*e.g.*, Yao et al. (2022); Shinn et al. (2023)) decomposes high-level tasks into sub-tasks or action sequences. Considering an action sequence as a root-to-leaf path in a tree containing all possible sequences, the planning can be viewed as finding optimal decisions from a tree-like search space (Yao et al., 2023). Regarding the large planning space for solving tasks in dynamic scenes, prompting LLMs with tree-like search methods (Korf, 1985; Haralick & Elliott, 1980; Browne et al., 2012) offers opportunities for better solutions and even the possibility of considering tasks from different perspectives. **Third**, supporting knowledge extension. Just as humans consult reference books to tackle domain-specific issues, DoraemonGPT is designed to select the most relevant knowledge source from a series of given external knowledge sources (*e.g.*, search engines, textbooks, databases, *etc*.) and then query the information from it during the planning.

More specifically, DoraemonGPT has a structure of ⟨*memory, tool, planner*⟩ (Fig. 1c): **i) Task-related Symbolic Memory (§2.1).** To collect information related to the given video and task, we consider decoupling *spatial-temporal* attributes into two memories: *space-dominant* and *time-dominant*. Before constructing these memories, LLMs are used to determine their relevance to the given task and keep only the useful one(s). Foundation models are then employed to extract *space-dominant* attributes (*e.g.*, instance trajectory, description, *etc*.) or *time-dominant* attributes (*e.g.*, video descriptions, audio speech, *etc*.) and integrate them into a query table, which facilitates LLMs to access information by using symbolic language (*e.g.*, SQL language). **ii) Sub-task (§2.1) and Knowledge (§2.2) Tools.** To compact our planner's context/text length and improve effectiveness, we simplify memory information querying by designing a series of sub-task tools. Each tool focuses on different kinds of *spatial-temporal* reasoning (*e.g.*, *"How..."*, *"Why..."*, *etc*.) by using individual LLM-driven sub-agents with task-specific prompts and examples. Additionally, dedicated knowledge tools can incorporate external sources for tasks requiring domain-specific knowledge. **iii) Monte Carlo Tree Search (MCTS) Planner (§2.3).** To efficiently explore the large planning space, we propose a novel tree-search-like planner. The planner iteratively finds feasible solutions by backpropagating the answer's reward and selecting a highly expandable node to expand a new solution. After summarizing all the results, the planner derives an informative final answer. To design the tree search planner, we equip our DoraemonGPT with MCTS (Coulom, 2006; Kocsis & Szepesvári, 2006; Browne et al., 2012), which has shown effectiveness in finding optimal decisions from a large search space (Vodopivec et al., 2017), especially in the game AI community (Gelly et al., 2006; Chaslot et al., 2008; Silver et al., 2017).

Combining the above designs, DoraemonGPT effectively handles dynamic *spatial-temporal* tasks, supports a comprehensive exploration of multiple potential solutions, and can extend its expertise by leveraging multi-source knowledge. Extensive experiments on three benchmarks (Xiao et al., 2021; Lei et al., 2020; Seo et al., 2020) show that our DoraemonGPT significantly outperforms recent LLM-driven competitors (*e.g.*, ViperGPT Surís et al. (2023)) in causal/temporal/descriptive reasoning and referring video object recognition. Also, our MCTS planner surpasses the naïve searching method and other baselines. Moreover, DoraemonGPT can deal with more complex in-the-wild tasks previously unapplicable or neglected by recent approaches (Surís et al., 2023; Li et al., 2023b).

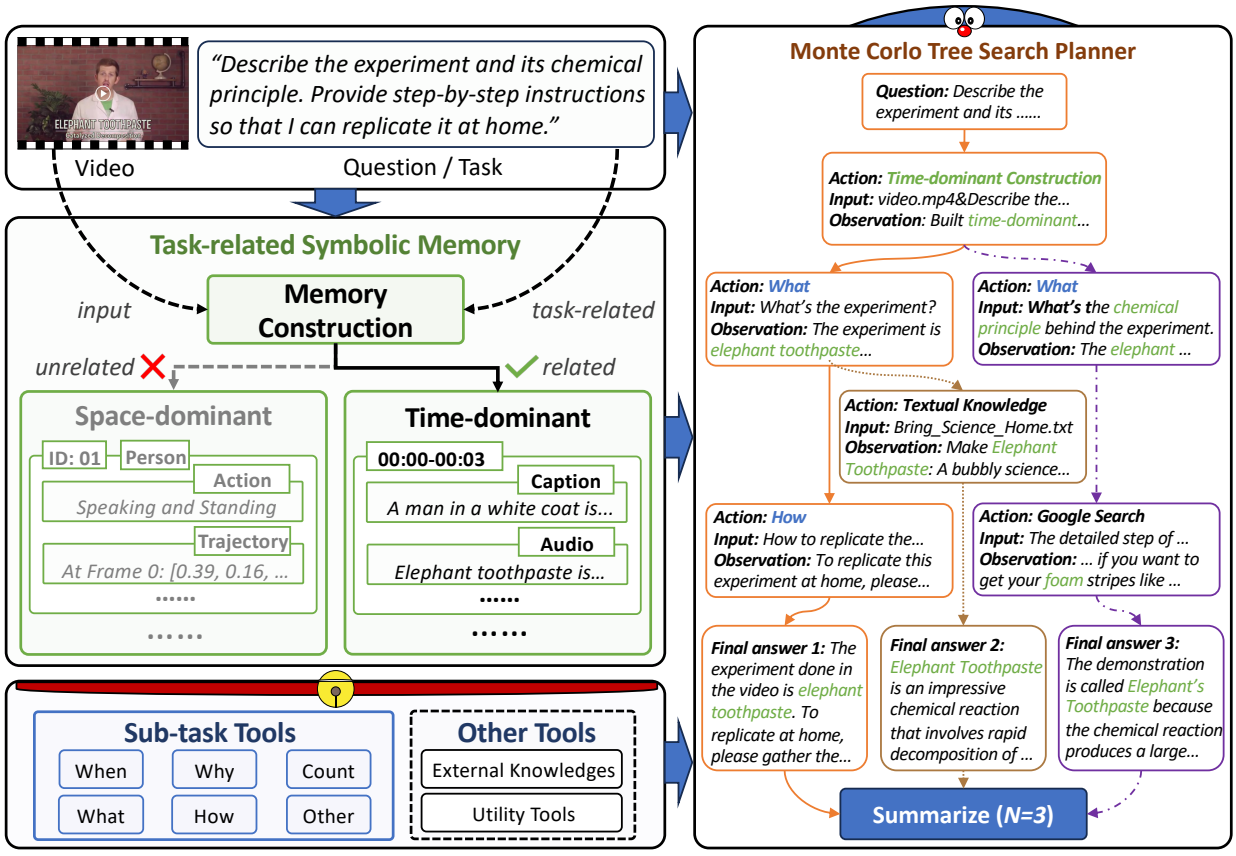


Figure 2. **Overview.** Given a video with a question/task, DoraemonGPT first extracts a Task-related Symbolic Memory (§2.1), which has two types: *space-dominant* memory based on instances and *time-dominant* memory based on time frames/clips. The memory can be queried by LLM-driven sub-task tools for different reasoning. Also, other tools for querying external knowledge (§2.2) or utility tools are supported. For planning, our MCTS Planner (§2.3) decomposes the question into an action sequence by exploring N feasible solutions, which can be further summarized into an informative answer.

2. DoraemonGPT

Overview. As shown in Fig. 2, DoraemonGPT is an LLM-driven agent capable of utilizing various tools to decompose a complex dynamic video task into sub-tasks and solve them. Given a video (V) with a textual task/question (Q), DoraemonGPT first extracts a Task-related Symbolic Memory (§2.1) from V based on the task analysis of Q. Next, employing a Monte Carlo Tree Search (MCTS) Planner (§2.3), DoraemonGPT automatically schedules the tool sets for querying the symbolic memory, accessing external knowledge (§2.2), and calling other utility tools (video inpainting, *etc.*) to solve the question Q. Ultimately, the planner explores the large planning space, returns multiple possible answers, and summarizes an improved answer.

2.1. Task-related Symbolic Memory (TSM)

Videos are complicated dynamic data, including *spatial-temporal* relations. Giveng a question Q for a video V, only a part of related attributes are critical to the solution, regardless of the large amount of irrelevant information. Thus, we propose to extract and store potentially relevant video information regarding Q into TSM before solving Q.

TSM Construction. To construct the TSM, we use a straightforward in-context learning method (Brown et al., 2020) to select the task type of TSM based on the question Q. We place the `task description` of each type of TSK into the context of our LLM-driven planner, which will be prompted to predict a suitable TSM in the format like *"Action: ⟨TSM_type⟩ construction..."*. Then, the API of constructing the corresponding TSM will be called to extract task-related attributes and store them in an SQL table, which can be accessed by symbolic languages, *i.e.*, SQL.

There is no standardized criterion for categorizing video tasks. In DoraemonGPT, we choose the perspective of *spatial-temporal* decoupling, which has been widely applied in video representation learning (Bertasius et al., 2021; Arnab et al., 2021), to design two memory types:

- *Space-dominant* memory is primarily used to address questions related to specific targets (*e.g.*, persons or an-

Table 1. Attributes in *space-* and *time-dominant* memories (§2.1). Each attribute is extracted/predicted based on different foundation models.

Attribute	Used Model	Explanation
Space-dominant Memory		
ID number	-	A unique ID assigned to an instance
Category	YOLOv8 (Jocher et al., 2023)/Grounding DINO (Liu et al., 2023c)	The category of an instance, *e.g.*, person
Trajectory	Deep OC-Sort (Maggiolino et al., 2023)/DeAOT (Yang & Yang, 2022)	An instance's bounding box in each frame
Segmentation	YOLOv8-Seg (Jocher et al., 2023)/DeAOT (Yang & Yang, 2022)	An instance's segmentation mask in each frame
Appearance	BLIP (Li et al., 2022) / BLIP-2 (Li et al., 2023a)	A description of an instance's appearance
Action	InternVideo (Wang et al., 2022)	The action of an instance
Time-dominant Memory		
Timestamp	-	The timestamp of a frame/clip
Audio content	Whisper (Radford et al., 2023)	Speech recognition results of the video
Optical content	OCR (PaddlePaddle, 2023)	Optical character recognition results of the video
Captioning	BLIP (Li et al., 2022)/BLIP-2 (Li et al., 2023a)/InstructBlip (Dai et al., 2023)	Frame-level/clip-level captioning results

imals) or their spatial relations. We use multi-object tracking methods (Maggiolino et al., 2023) to detect and track instances. Each instance has attributes that include unique `ID`, semantic `category`, `trajectory` & `segmetnation` for localization, `appearance` description extracted by (Li et al., 2022; 2023a) and used for text-based grounding, and `action` classification.

- *Time-dominant* memory focuses on constructing temporal-related information of the video. It requires comprehending the content throughout the video. The attributes stored in this memory include `timestamp`, `audio content` by ASR (Radford et al., 2023), `optical content` by OCR (PaddlePaddle, 2023), frame-level `captioning` by BLIPs (Li et al., 2022; 2023a; Dai et al., 2023), clip-level `captioning` by dedupicating similar and continuous frame-level results, etc.

Table 1 provides the attribute types with corresponding extraction models of our TSMs.

Sub-task Tools. Although LLM-driven agents (Hu et al., 2023; Li et al., 2023b) can assess external information by in-context learning of the whole memory or generating symbolic sentences to access the memory, these methods can significantly increase the length of the context, potentially leading to the omission of crucial information in the reasoning process or being influenced by redundant context. Thus, we provide a series of sub-task tools responsible for querying information from our TSMs (Shi et al., 2023; Liu et al., 2023a) by answering sub-task questions. The LLM-driven planner learns the function of each sub-task tool through its in-context `description`, which describes the `sub-task description`, `tool name`, and `tool inputs`. To call the API of a sub-task tool, DoraemonGPT parses the command generated by LLMs, like *"Action: ⟨tool_name⟩. Input: ⟨video_name⟩#⟨sub_question⟩..."*.

To collaborate with the above two kinds of TSMs, we design sub-task tools with different `sub-task description` and for solving different sub-questions, including:

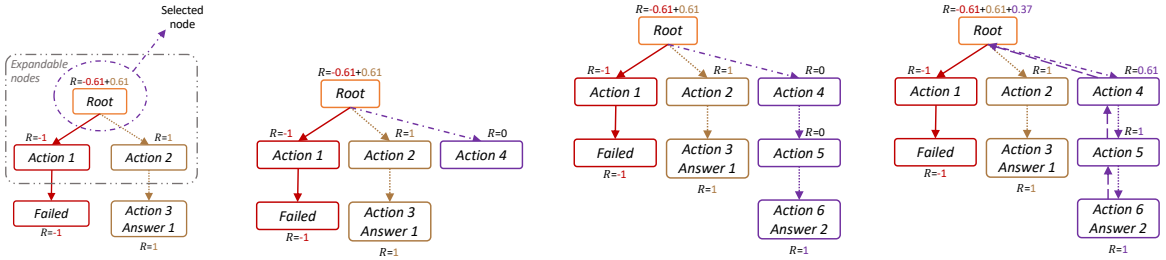
- *When*: related to temporal understanding, *e.g.*, *"When the dog walks past by the sofa?"*
- *Why*: related to causal reasoning, *e.g.*, *"Why did the lady shake the toy?"*
- *What*: describing the required information, *e.g.*, *"What's the name of the experiment?"*
- *How*: what manner, means, or quality of something, *e.g.*, *"How does the baby keep himself safe?"*
- *Count*: counting sth., *e.g.*, *"How many people in the room?"*
- *Other*: questions not in the above tools, *e.g.*, *"Who slides farther at the end?"*

The `API functions` of these tools are built upon LLMs as well. Each sub-task tool function is an individual LLM-driven agent, which can generate SQL to query our TSMs and answer the given sub-task question. Different sub-task agents have different in-context examples regarding their purposes. Note that a sub-question may be suitable for two or more sub-tools (*e.g.*, *"What was the baby doing before playing the toy?"*, related to *what* and *when*), and our MCTS planner (§2.3) is capable of exploring different selections.

2.2. Knowledge Tools and Others

When tackling complex problems, LLM-driven agents sometimes fail to make accurate decisions solely based on video understanding and the implicit knowledge learned by LLMs during training. Thus, DoraemonGPT supports the integration of external knowledge sources that can assist the LLM in comprehending the specialized content within the input video/question. In DoraemonGPT, a knowledge source can be integrated in a plug-and-play manner by using an individual knowledge tool. Similar to the sub-task tools (§2.1), a knowledge tool consists of two parts: **i**) an in-context `knowledge description` to describe the given knowledge source and **ii**) an `API function` to query information from the source by question answering.

We consider three types of `API function` for covering different knowledge: **i**) *symbolic knowledge* refers to information presented in a structured format such as Excel or SQL tables. The `API function` is a symbolic question-answering sub-agent like our sub-task tools (§2.1). **ii**) *textual knowledge* encompasses knowledge expressed through



(a) Node selection (b) Branch Expansion (c) Chain Execution (d) Back-propagation

Figure 3. An illustration of our Monte Carlo Tree Search (MCTS) planner (§2.3). R: the reward of a node. *Root*: the input video and question/task. *Action*: a ReAct (Yao et al., 2022)-style step in the form of ⟨*thought, action, action input, observation*⟩.

natural language text, such as research publications, reference books, *etc*. The `API function` is built based on text embedding and searching (OpenAI, 2022). **iii**) *web knowledge* denotes knowledge searched from the internet. The `API function` is a search engine API, such as Google, Bing, *etc*. Besides the knowledge tools, DoraemonGPT also supports integrating general *utility tools*, commonly used in recent LLM-driven agents (Xi et al., 2023), to help complete specialized vision tasks, *e.g*., video editing and inpainting.

2.3. Monte Carlo Tree Search (MCTS) Planner

Previous LLM-driven planners (Shen et al., 2023; Surís et al., 2023; Gupta & Kembhavi, 2023) decompose the given Q into an action/sub-task sequence and solve it step by step. Such a strategy can be seen as a greedy search method that generates a chain of action nodes until the final answer. Similar to some works (Yao et al., 2024; Zhuang et al., 2024), we consider the large planning space of LLM-driven planning as a tree. Moreover, we consider that a single attempt may not yield the correct result or that better solutions may exist. To efficiently explore the planning space, we propose a novel tree-search-like planner equipped with MCTS (Coulom, 2006; Kocsis & Szepesvári, 2006; Browne et al., 2012), which is practical in searching large trees.

We define the question input Q as the root node v_0, and an action or tool call is a non-root node, then an action sequence can be viewed as a path from the root node to a leaf node. In our MCTS planner, a non-root node is a ReAct (Yao et al., 2022)-style step in the form of ⟨*thought, action, action input, observation*⟩, and a leaf node has a *final answer* in addition. The planner iteratively executes the following four phases for N times and produces N solutions:

Node Selection. Each iteration starts by selecting an expandable node for planning a new solution. For the first iteration, only the root v_0 is selectable. For subsequent iterations, we randomly select a non-leaf node based on their sampling probability, formulated as $P(v_i) = Softmax(R_i)$, where R_i is the reward value of node v_i initialized as 0 and updated in the Reward Back-propagation phase. The node with a higher reward has a greater probability of being selected.

Branch Expansion. A child will be added to the selected expandable node to create a new branch. To leverage LLM for generating a new tool call different from the previous child nodes, we add historical tool actions into the prompt of LLM and instruct it to make a different choice. Such an in-context prompt will be removed in the subsequent chain execution toward a new final answer.

Chain Execution. After expanding a new branch, we use a step-wise LLM-driven planner (Yao et al., 2022) to generate a new solution. The execution process consists of a chain of steps/nodes of tool calls and will terminate upon obtaining the final answer or encountering an execution error.

Reward Back-propagation. After obtaining a leaf/outcome node v_l, we will gradually propagate its reward to its ancestor nodes until v_0. Here, we consider two kinds of reward:

- *Failure*: the planner produces an unexpected result, *e.g*., a failed tool call or incorrect result format. The reward R_{v_l} is set to a negative value (*e.g*., -1) for these cases.
- *Non-failure*: the planner successfully produces a result, but it is not sure whether the result is correct as ground truth. Here R_{v_l} is set to a positive value (*e.g*., 1).

For simplicity, let α be a positive base reward, we set $R_{v_l} = \pm\alpha$ for *Failure* and *Non-failure*, respectively. According to (Liu et al., 2023a), the outcome produced by LLMs is more related to the context at the beginning (the initial prompts) and the end (the final nodes). We believe more rewards should be applied to nodes close to v_l. Thus, the backpropagation function is formulated as $R_{v_i} \leftarrow R_{v_i} + R_{v_l} e^{\beta(1-d(v_i,v_l))}$, where $d(v_i, v_l)$ denotes the node distance between v_i and v_l, and β is a hyperparameter for controlling the decay rate of the reward. The further the node distance, the greater the reward decay ratio, $e^{\beta(1-d(v_i,v_l))}$. Commonly, setting a higher α/β will increase the probability of the expanded node approaching the leaf node (of a non-failure answer).

After all the MCTS iterations, the planner will produce N *non-failure* answers at most, and we can use LLMs to summarize all the answers to generate an informative answer.

Table 2. Quantitative comparison of Video Question Answering (§3.2) and Referring Video Object Segmentation (§3.3). All LLM-driven agents use the same LLM, *i.e.*, GPT-3.5-turbo. †: reimplement using the officially released codes. ‡: we equipped ViperGPT (Surís et al., 2023) with DeAOT (Yang & Yang, 2022; Cheng et al., 2023) as ours to enable object tracking and segmenation.

	Method	Pub.	Acc_C	Acc_T	Acc_D	Avg	Acc_A
Supervised	HME (Fan et al., 2019)	CVPR19	46.2	48.2	58.3	50.9	48.7
	VQA-T (Yang et al., 2021a)	ICCV21	41.7	44.1	60.0	48.6	45.3
	ATP (Buch et al., 2022)	CVPR22	53.1	50.2	66.8	56.7	54.3
	VGT (Xiao et al., 2022)	ECCV22	52.3	55.1	64.1	57.2	55.0
	MIST (Gao et al., 2023b)	CVPR23	54.6	56.6	66.9	59.3	57.2
Agent	†ViperGPT (Surís et al., 2023)	ICCV23	43.2	41.0	62.3	49.4	45.5
	†VideoChat (Li et al., 2023b)	arXiv23	50.2	47.0	65.7	52.5	51.8
	DoraemonGPT (Ours)	ICML24	**54.7**	**50.4**	**70.3**	**54.7**	**55.7**

(a) NExT-QA (Xiao et al., 2021)

	Method	Pub.	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{J}\&\mathcal{F}$
Supervised	CMSA (Ye et al., 2019)	CVPR19	36.9	43.5	40.2
	URVOS (Seo et al., 2020)	ECCV20	47.3	56.0	51.5
	VLT (Ding et al., 2021)	ICCV21	58.9	64.3	61.6
	ReferFormer (Wu et al., 2022a)	CVPR22	58.1	64.1	61.1
	SgMg (Miao et al., 2023)	ICCV23	60.6	66.0	63.3
	OnlineRefer (Wu et al., 2023b)	ICCV23	61.6	67.7	64.8
Age.	‡ViperGPT (Surís et al., 2023)	ICCV23	24.7	28.5	26.6
	DoraemonGPT (Ours)	ICML24	**63.9**	**67.9**	**65.9**

(b) Ref-YouTube-VOS (Seo et al., 2020)

Alternatively, for single-/multiple-choice questions, we can determine the final answer through a voting process.

3. Experiment

3.1. Experimental Setup

Datasets. For validating our algorithm's utility comprehensively, we conduct experiments on three datasets, *i.e.*, NExT-QA (Xiao et al., 2021), TVQA+ (Lei et al., 2020), and Ref-YouTube-VOS (Seo et al., 2020). They were selected to cover the common dynamic scenes with *video question answering* and *referring video object segmentation* tasks.

- **NExT-QA** contains 34,132/4,996 *video-question* pairs for `train`/`val`. Each question is annotated with a question type (causal/temporal/descriptive) and 5 answer candidates. We randomly sample 30 samples per type from `train` (90 questions in total) for ablation studies, and the `val` set is used for method comparison.
- **TVQA+** is an enhanced version of TVQA (Lei et al., 2018) dataset, augmented with 310.8K bounding boxes to link visual concepts in questions and answers to depicted objects in videos. For evaluation, we randomly sample 900 samples from the `val` set as in (Gupta & Kembhavi, 2023), resulting in a total of 900 questions (`s_val`).
- **Ref-YouTube-VOS** is a large-scale *referring video object segmentation* dataset with ∼15,000 referential expressions associated with >3,900 videos, covering diverse scenarios. In our experiments, we use the validation set of Ref-YouTube-VOS (Seo et al., 2020) (including 202 videos and 834 objects with expressions) to validate our effectiveness in pixel-wise *spatial-temporal* segmentation.

Evaluation Metric. For question answering, we adopt the standard metric (Xiao et al., 2021), top-1 accuracy, for evaluation. On NExT-QA, we also report accuracy for causal (Acc_C), temporal (Acc_T), descriptive (Acc_D), Avg (the average of Acc_C, Acc_T, and Acc_D) and Acc_A (the overall accuracy of all questions). For referring object segmentation, we evaluated the performance on the official challenge server of Ref-YouTube-VOS. the metrics are the average of region similarity ($\mathcal{J}$) and contour accuracy ($\mathcal{F}$), collectively referred to as $\mathcal{J}\&\mathcal{F}$.

Implementation Details. We use GPT-3.5-turbo API provided by OpenAI as our LLM. As summarized in Table 1, we use BLIP series (Dai et al., 2023) for captioning, YOLOv8 (Jocher et al., 2023) and Deep OC-Sort (Maggiolino et al., 2023) for object tracking, PaddleOCR (PaddlePaddle, 2023) for OCR, InternVideo (Wang et al., 2022) for action recognition, and Whisper (Radford et al., 2023) for ASR. Our experiments are conducted under the in-context learning (ICL) setting. The external knowledge is not used in quantitative or qualitative comparisons to ensure fairness.

Competitors. We involve several *open-sourced* LLM-driven agents for comparison. ViperGPT (Surís et al., 2023) leverages code generation models to create subroutines from vision-and-language models through a provided API, thus it solves the given task by generating Python code that is later executed. VideoChat (Li et al., 2023b) is an end-to-end chat-centric video understanding system that integrates several foundation models and LLMs to build a chatbot. We do not report the performance of others since they do not release their codes for video tasks or even open-source it.

Hyperparameters. We set $\alpha = 1$ and $\beta = 0.5$ for the reward back-propagation (§2.3). For experiments on VQA, exploring $N = 2$ solutions has better acc.-cost trade-offs.

3.2. Zero-shot Video Question Answering

NExT-QA. Table 2a presents comparisons of our DoraemonGPT against several top-leading supervised VQA models and LLM-driven systems. As can be seen, DoraemonGPT achieves competitive performance compared to recently proposed supervised models. In particular, it shows a more promising improvement in descriptive questions, even outperforming the previous SOTA, MIST (Gao et al., 2023b) (**70.3** *vs* 66.9). The main reason is that our task-related symbolic memory can provide enough information for reason-

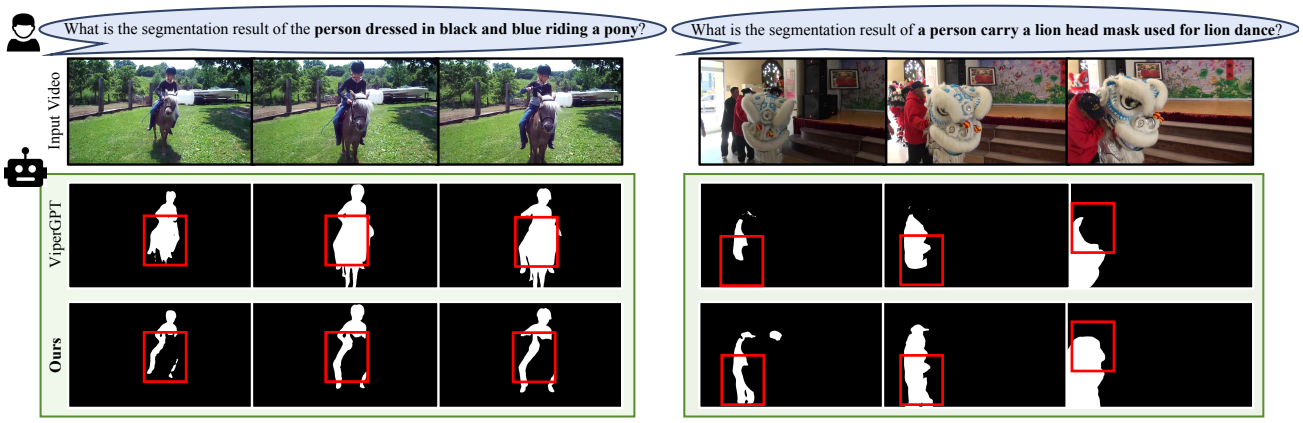


Figure 4. Qualitative comparison of Referring Video Object Segmentation (§3.3) on Ref-YouTube-VOS (Seo et al., 2020).

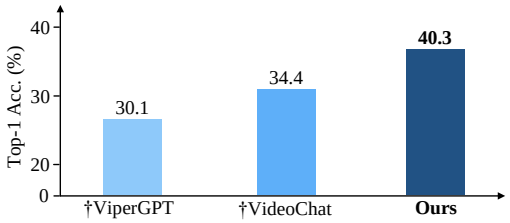


Figure 5. Comparison on TVQA+ (Lei et al., 2020) (§3.2).

ing. In terms of temporal questions, supervised models are slightly superior to ours, mainly due to their well-designed architectures that have learned underlying patterns. In addition, DoraemonGPT outperforms recent concurrent works, *i.e.*, ViperGPT and VideoChat. Concretely, it outperforms ViperGPT by **11.5/9.4/8.0/10.2** (Acc_C/Acc_T/Acc_D/Acc_A) and VideoChat by **4.5/3.4/4.6/3.9** across the four question types. These results show the efficacy of our MCTS planner based on TSM. Inference examples are in A.3.

TVQA+. The results in Fig. 5 confirm again the superiority of our method. DoraemonGPT outperforms ViperGPT and VideoChat by **10.2**% and **5.9**%, respectively. ViperGPT's performance (30.1%) is lower compared to VideoChat and DoraemonGPT, which are specifically designed for dynamic videos. This is consistent with the findings on NExT-QA.

3.3. Zero-shot Referring Object Segmentation

Ref-YouTube-VOS. We further evaluate DoraemonGPT against state-of-the-art supervised referring video object segmentation models and LLM-driven agents, as summarized in Table 2b. Benefiting from our task-related symbolic memory, the DoraemonGPT (without learning on Ref-YouTube-VOS) can effectively ground video instances with textual description and remarkably surpasses recent supervised models (**65.9%** *v.s.* 64.8%), *e.g.*, OnlineRefer (Wu et al., 2023b). By contrast, the agent competitor, ViperGPT, only achieves 26.6% due to a lack of well-designed video information memory, failing to ground the referred object or track it accurately in the video.

As depicted in Fig. 4, our DoraemonGPT demonstrates a higher level of accuracy in identifying, tracking, and segmenting the referred objects mentioned by the users. In contrast, ViperGPT encounters failure cases where the recognized objects do not completely match the semantic and descriptive aspects. Our superiority in referring video object segmentation further demonstrates the necessity of building a symbolic video memory.

3.4. In-the-wild Example

DoraemonGPT exhibits a versatile skill set, *e.g.*, checking experimental operations, video understanding, and video editing. It adeptly navigates complex questions by exploring multiple reasoning paths and leveraging external sources for comprehensive answers. Please see A.2 for more details.

3.5. Diagnostic Experiment

To gain more insights into DoraemonGPT, we conduct a set of ablative experiments on NExT-QA.

Task-related Symbolic Memory. First, we investigate the essential components in DoraemonGPT, *i.e.*, symbolic memories (§2.1) for *space-dominant* (SDM) and *time-dominant* (TDM) information. The results are summarized in Table 3a. Two crucial conclusions can be drawn. **First**, TDM is more preferred for temporal questions, while SDM can provide relevant information for descriptive questions. **Second**, our complete system achieves the best performance by combining our SDM and TDM, confirming the necessity of dynamically querying two types of symbolic memory.

Multiple Solutions by MCTS Planner. We will next study the influence of the number of answer candidates during the exploration process of our MCTS planner. When $N = 1$,

Table 3. A set of ablative experiments (§3.5) about the MCTS planner on NExT-QA (Xiao et al., 2021) `s_train`.

TDM	SDM	Acc_C	Acc_T	Acc_D	Acc_A
✓		63.3	26.7	53.3	47.8
	✓	53.3	23.3	46.7	41.1
✓	✓	96.7	46.7	53.3	65.7

(a) Essential components

Models	Acc_C	Acc_T	Acc_D	Acc_A
BLIP-2	51.4	45.5	63.3	51.2
InstructBlip	54.7	50.4	70.3	55.7

(b) Captioning models

N	Acc_C	Acc_T	Acc_D	Acc_A
1	63.3	20.0	46.7	43.3
2	80.0	43.3	46.7	56.7
3	86.7	43.3	53.3	61.1
4	96.7	46.7	53.3	65.7
5	86.7	43.3	50.0	60.0

(c) Number of answer candidates

α	β	Acc_C	Acc_T	Acc_D	Acc_A
0.5	1.0	86.7	23.3	50.0	53.3
1.0	0.5	96.7	46.7	53.3	65.7
0.5	2.0	86.7	26.7	50.0	54.4
2.0	0.5	83.3	46.7	50.0	60.0
2.0	2	80.0	46.7	50.0	58.9

(d) Reward and decay rate ($N = 4$)

Strategy	Acc_C	Acc_T	Acc_D	Acc_A
DFS	66.7	36.7	50.0	51.1
Root	73.3	16.7	46.7	45.6
Uniform	67.7	26.7	50.0	47.8
MCTS	96.7	46.7	53.3	65.7

(e) Exploring strategy ($N = 4$)

the planner is equivalent to a greedy search, explores only a chain of nodes, and returns a single answer – the first node in LLM's thinking that can be terminated without further exploration. As shown in Table 3c, gradually increasing N from 1 to 4 leads to better performance (i.e., 43.3 $\rightarrow$ 65.7). This supports our hypothesis that one single answer is far from enough to handle the larger planning space for dynamic modalities and proves the efficacy of our MCTS planner. Since the questions in NExT-QA (Xiao et al., 2021) are single-choice, exploring more answers does not always result in positive returns. We stop using $N > 5$ as the required number of API calls exceeds our budget.

Back-propogation in MCTS Planner. Then, we ablate the effect of the base reward α and decay rate β (§2.3) that controls the exploring procedure of our MCTS planner. The results in Table 3d are stable regardless of the combination of α and β used. Hence, we set the slightly better one, $\alpha = 1$ and $\beta = 0.5$, as our default setting. We leave some special combinations in the next part (*e.g.*, our MCTS planner becomes the depth-first search (*DFS*) when setting $\beta = 10^8$ and $R_{v_l} = 1$ for both *Failure* and *Non-failure* cases).

Exploring Strategies used by Planner. Last, to verify the advantage of our MCTS planner, we compare MCTS with several standard exploring strategies, *i.e.*, depth-first search (*DFS*), *Root*, which always selects the root node, and *Uniform*, which samples a node with equal probability. As shown in Table 3e, we observe that their performance is suboptimal due to the inability to leverage the value/reward of the outcome leaf nodes and accordingly adjust their searching strategy. Compared to these naïve strategies, our MCTS planner adaptively samples a node with the guidance of reward back-propagation, which is more effective in a large solution space. These results further validate the superiority of the proposed MCTS planner.

Additionally, compared to *Uniform*, *DFS* notably exhibits superior performance on temporal questions yet demonstrates comparable or even inferior performance on descriptive and causal questions. We hypothesize that this discrepancy arises because temporal questions often include cues (e.g., at the beginning) indicating the specific period within the video where the correct answer can be found. *DFS* can exploit these cues, whereas *Uniform* cannot.

Impact of Captioning Models. We further conduct experiments regarding one of the most important models, i.e., BLIP series, which is the essential tool for DoraemonGPT to perceive and recognize visual input. As seen in Table 3b, the model with instruction tuning, i.e., InstructBLIP, demonstrates better performance. This suggests that DoraemonGPT can benefit from the development of stronger foundation models. More discussion of foundation models' impact on DoraemonGPT is supplied in A.4.

4. Related Work

Multimodal Understanding. Various efforts were made to create multimodal systems tailored for specific tasks (Lu et al., 2019; Marino et al., 2019; 2021; Bain et al., 2021; Lei et al., 2021; Grauman et al., 2022; Lin et al., 2022; Li et al., 2023d;c; Lei et al., 2024; Wong et al., 2022; Chen et al., 2023a; Hu et al., 2020; Yang et al., 2021b). While these systems showed impressive performance in their respective domains, their applicability to broader, real-world scenarios was limited due to the lack of generalizability. Recent years have witnessed remarkable progress in *general* multimodal systems, due to the fast evolution of data volumes and computational resources. Specifically, Frozen (Tsimpoukelli et al., 2021) is a typical example; it demonstrates a feasible way to empower LLMs with the ability to handle visual inputs. In the past few years, numerous efforts have been devoted to building large multimodal models (OpenAI, 2023; Driess et al., 2023; Zhu et al., 2023a; Li et al., 2022). Considering the training cost, several attempts (Li et al., 2023a; Yu et al., 2023) try to build zero-shot systems for various tasks. An alternative strategy (Xi et al., 2023), which will be detailed later, involves combining multiple models or APIs to tackle compositional multimodal reasoning tasks. Our DoraemonGPT shares a similar spirit of decomposing complex tasks into simpler ones, but it is designed to solve complicated tasks for dynamic modalities in real-world scenarios.

LLM-driven Modular Systems. Deconstructing complex tasks and merging the results from multiple intermediate steps is an innate human ability that drives the scientific and industrial communities (Newell et al., 1972; Wang & Chiew, 2010). Benefiting from the impressive emergent capabilities

of LLMs, VisProg (Gupta & Kembhavi, 2023) pioneers the idea of addressing complex vision tasks through the decomposition of questions into manageable subtasks. Along this line, tremendous progress has been achieved, which can be divided into two categories according to reasoning styles: i) Reasoning with fixed paths (Gupta & Kembhavi, 2023; Wu et al., 2023a; Surís et al., 2023; Lu et al., 2023; Shen et al., 2023; Liang et al., 2023). They transform the given task into an ordered sequence of subtasks, each addressed by a specific module. For example, ViperGPT(Surís et al., 2023) treats the solving process as a Python program with manually designed APIs. Similarly, HuggingGPT (Shen et al., 2023) models task dependencies between numerous foundation models. ii) Reasoning with dynamic paths(Nakano et al., 2021; Schick et al., 2023; Yao et al., 2022; Yang et al., 2023; Gao et al., 2023a). Considering the intermediate results may not meet expectations, a promising avenue is to perform planning and executing concurrently. Such an interactive paradigm (Yao et al., 2022) provides a flexible and error-tolerant manner compared to those with fixed paths. Additionally, there are many agents focussing on other domains, *e.g.*, planning in the open-world environments (Wang et al., 2023c; Yuan et al., 2023; Park et al., 2023), tool usage (Ruan et al., 2023; Qin et al., 2023), reinforcement learning (Shinn et al., 2023; Xu et al., 2023). This work focuses on computer vision only.

Despite impressive, existing LLM-driven modular systems mainly focus on developing specific strategies to solve composition tasks for *static* modalities, ignoring the fundamental gaps between static and dynamic modalities, which is a pivotal aspect towards achieving artificial general intelligence (AGI) (Goertzel, 2014). These works, to some extent, can be seen as a subset of our system. Though exceptions exist (Surís et al., 2023; Li et al., 2023b; Wang et al., 2023a; Gao et al., 2023a), in general, they are scattered, lacking systematic study, *e.g.*, simply treating the video as a sequence of images (Surís et al., 2023) or building a chatbot based on pre-extracted information of the given video (Li et al., 2023b; Wang et al., 2023a). In sharp contrast, we take the video and task as a whole, resulting in a compact, *task-related* memory. The reasoning paths of our system are powered by the MCTS planner. In addition to facilitating the answer searching, the MCTS planner has the potential to find multiple possible candidates. This is crucial for questions with open-ended answers.

LLMs with External Memory. How to effectively design prompt templates, known as prompt engineering, is of importance for accurate LLM responses (Zhou et al., 2022; Wang et al., 2023b). One of the areas in the spotlight is memory-augmented LLMs (Wu et al., 2022c;b; Zhong et al., 2022; Lewis et al., 2020; Guu et al., 2020; Izacard et al., 2022; Khattab et al., 2022; Park et al., 2023; Cheng et al., 2022; Sun et al., 2023; Hu et al., 2023). In general, training-free memories can be divided into: i) Textual memory (Zhu et al., 2023b; Park et al., 2023). In this kind of memory, the long contexts LLMs cannot handle (*e.g.*, books) are stored as embeddings, which can be further retrieved by computing similarity. A typical example is the document question answering shown in LangChain[1]. ii) Symbolic memory. It models memory as structured representations with corresponding symbolic languages, e.g., codes for programming language (Cheng et al., 2022), execution commands for Excel[2], and structured query language (SQL) for databases (Sun et al., 2023; Hu et al., 2023). Different from techniques (Dao et al., 2022; Dao, 2023; Ratner et al., 2023; Hao et al., 2022; Chen et al., 2023b; Mu et al., 2023; Mohtashami & Jaggi, 2023; Peng et al., 2023b) that directly extend the context window of LLMs, memory-augmented LLMs use retrieval-based approaches to bypass the limitation of context length. This is more favored because (i) it is a plug-and-play module without any fine-tuning or architectural modifications, and (ii) concurrent works (Shi et al., 2023; Liu et al., 2023a) suggest that LLMs may be distracted or lost while encountering irrelevant or long contexts. By absorbing their ideas of memory organization, we construct a request-related database, which stores instance-aware and instance-agnostic information in individual tables. To retrieve the relevant information, we explicitly define several sub-task tools based on prompt templates and SQL, with respect to different purposes. With a broader view, our multi-source knowledge, which is a complementary module to provide reliable guidance in specific domains, can also be considered as a hybrid of external memories.

5. Conclusion

Regarding real-world scenarios' dynamic and ever-changing nature, we present DoraemonGPT, an LLM-driven agent for solving dynamic video tasks. Compared to existing LLM-driven visual modular systems, DoraemonGPT has merits in: **i**) conceptually elegant system designed by delving into the dynamic modalities in our lives; **ii**) compact task-related symbolic memory by decoupling, extracting, and storing *spatial-temporal* attributes; **iii**) effective and decomposed memory querying through symbolic sub-task tools; **iv**) plug-and-play knowledge tools for accessing domain-specific knowledge; **v**) automated exploration of large planning space using the MCTS planner, providing multiple solutions with an informative final answer; and **vi**) answer diversity that provides multiple potential candidates by fully exploring the solution space. Extensive experiments confirm the versatility and effectiveness of DoraemonGPT.

[1]https://docs.langchain.com/docs/use-cases/qa-docs
[2]https://chatexcel.com/

Acknowledgments

This work was supported by the National Science and Technology Major Project (No. 2023ZD0121300), the National Natural Science Foundation of China (No. 62372405), and the CCF-Tencent Open Fund.

Impact Statement

DoraemonGPT leverages LLMs to address real-world dynamic tasks, excelling in video-based reasoning with potential applications in autonomous vehicles, surveillance, and interactive robotics. Despite its promise, several ethical concerns must be addressed. **i)** The system could be misused for video manipulation or generating misleading content, necessitating robust safeguards against malicious activities. **ii)** Biases in training data could perpetuate discriminatory behavior, highlighting the need for fairness and bias mitigation. **iii)** Reliance on external knowledge sources underscores the importance of adhering to data access and usage regulations to avoid legal issues. **iv)** DoraemonGPT's methodology extends the application of LLM-driven agents beyond vision, offering transformative impacts in various fields. For instance, the MCTS planner enhances exploration strategies in large solution spaces, while its symbolic memory aids interactive planning scenarios with precise guidance, relevant for diverse applications, *e.g.*, embodied intelligence (Liu et al., 2024; 2023b; Dong et al., 2024).

References

Anil, R., Dai, A. M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., and Schmid, C. Vivit: A video vision transformer. In *ICCV*, pp. 6836–6846, 2021.

Bain, M., Nagrani, A., Varol, G., and Zisserman, A. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *ICCV*, pp. 1728–1738, 2021.

Bertasius, G., Wang, H., and Torresani, L. Is space-time attention all you need for video understanding? In *ICML*, volume 2, pp. 4, 2021.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *NeurIPS*, 33:1877–1901, 2020.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Buch, S., Eyzaguirre, C., Gaidon, A., Wu, J., Fei-Fei, L., and Niebles, J. C. Revisiting the" video" in video-language understanding. In *CVPR*, pp. 2917–2927, 2022.

Chaslot, G. M. B., Winands, M. H., and van Den Herik, H. J. Parallel monte-carlo tree search. In *Computers and Games: 6th International Conference, CG 2008, Beijing, China, September 29-October 1, 2008. Proceedings 6*, pp. 60–71, 2008.

Chen, J., Gao, D., Lin, K. Q., and Shou, M. Z. Affordance grounding from demonstration video to target image. In *CVPR*, pp. 6799–6808, 2023a.

Chen, S., Wong, S., Chen, L., and Tian, Y. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023b.

Cheng, Y., Li, L., Xu, Y., Li, X., Yang, Z., Wang, W., and Yang, Y. Segment and track anything. *arXiv preprint arXiv:2305.06558*, 2023.

Cheng, Z., Xie, T., Shi, P., Li, C., Nadkarni, R., Hu, Y., Xiong, C., Radev, D., Ostendorf, M., Zettlemoyer, L., et al. Binding language models in symbolic languages. *arXiv preprint arXiv:2210.02875*, 2022.

Chiang, W.-L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J. E., et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2023.

Coulom, R. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pp. 72–83, 2006.

Dai, W., Li, J., Li, D., Tiong, A. M. H., Zhao, J., Wang, W., Li, B., Fung, P., and Hoi, S. C. H. Instructblip: Towards general-purpose vision-language models with instruction tuning. *ArXiv*, 2023.

Dao, T. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. *NeurIPS*, 35:16344–16359, 2022.

Ding, H., Liu, C., Wang, S., and Jiang, X. Vision-language transformer and query generation for referring segmentation. In *ICCV*, pp. 16321–16330, 2021.

Dong, Y., Zhu, X., Pan, Z., Zhu, L., and Yang, Y. Villageragent: A graph-based multi-agent framework for coordinatingcomplex task dependencies in minecraft. In *ACL*, 2024.

Driess, D., Xia, F., Sajjadi, M. S., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Fan, C., Zhang, X., Zhang, S., Wang, W., Zhang, C., and Huang, H. Heterogeneous memory enhanced multimodal attention model for video question answering. In *CVPR*, pp. 1999–2007, 2019.

Gao, D., Ji, L., Zhou, L., Lin, K. Q., Chen, J., Fan, Z., and Shou, M. Z. Assistgpt: A general multi-modal assistant that can plan, execute, inspect, and learn. *arXiv preprint arXiv:2306.08640*, 2023a.

Gao, D., Zhou, L., Ji, L., Zhu, L., Yang, Y., and Shou, M. Z. Mist: Multi-modal iterative spatial-temporal transformer for long-form video question answering. In *CVPR*, pp. 14773–14783, 2023b.

Gelly, S., Wang, Y., Munos, R., and Teytaud, O. *Modification of UCT with patterns in Monte-Carlo Go*. PhD thesis, INRIA, 2006.

Goertzel, B. Artificial general intelligence: concept, state of the art, and future prospects. *Journal of Artificial General Intelligence*, 5(1):1, 2014.

Grauman, K., Westbury, A., Byrne, E., Chavis, Z., Furnari, A., Girdhar, R., Hamburger, J., Jiang, H., Liu, M., Liu, X., et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *CVPR*, pp. 18995–19012, 2022.

Gupta, A., Dollar, P., and Girshick, R. Lvis: A dataset for large vocabulary instance segmentation. In *CVPR*, pp. 5356–5364, 2019.

Gupta, T. and Kembhavi, A. Visual programming: Compositional visual reasoning without training. In *CVPR*, pp. 14953–14962, 2023.

Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M. Retrieval augmented language model pre-training. In *ICML*, pp. 3929–3938, 2020.

Hao, Y., Sun, Y., Dong, L., Han, Z., Gu, Y., and Wei, F. Structured prompting: Scaling in-context learning to 1,000 examples. *arXiv preprint arXiv:2212.06713*, 2022.

Haralick, R. M. and Elliott, G. L. Increasing tree search efficiency for constraint satisfaction problems. *Artificial intelligence*, 14(3):263–313, 1980.

Hu, C., Fu, J., Du, C., Luo, S., Zhao, J., and Zhao, H. Chatdb: Augmenting llms with databases as their symbolic memory. *arXiv preprint arXiv:2306.03901*, 2023.

Hu, R., Singh, A., Darrell, T., and Rohrbach, M. Iterative answer prediction with pointer-augmented multimodal transformers for textvqa. In *CVPR*, pp. 9992–10002, 2020.

Izacard, G., Lewis, P., Lomeli, M., Hosseini, L., Petroni, F., Schick, T., Dwivedi-Yu, J., Joulin, A., Riedel, S., and Grave, E. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*, 2022.

Jocher, G., Chaurasia, A., and etc. Yolo by ultralytics. https://github.com/ultralytics/ultralytics, 2023.

Khattab, O., Santhanam, K., Li, X. L., Hall, D., Liang, P., Potts, C., and Zaharia, M. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp. *arXiv preprint arXiv:2212.14024*, 2022.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

Kocsis, L. and Szepesvári, C. Bandit based monte-carlo planning. In *ECML*, pp. 282–293, 2006.

Korf, R. E. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial intelligence*, 27(1):97–109, 1985.

Lei, J., Yu, L., Bansal, M., and Berg, T. L. Tvqa: Localized, compositional video question answering. In *EMNLP*, 2018.

Lei, J., Yu, L., Berg, T. L., and Bansal, M. Tvqa+: Spatio-temporal grounding for video question answering. In *ACL*, 2020.

Lei, J., Li, L., Zhou, L., Gan, Z., Berg, T. L., Bansal, M., and Liu, J. Less is more: Clipbert for video-and-language learning via sparse sampling. In *CVPR*, pp. 7331–7341, 2021.

Lei, J., Li, L., Wang, C., Xiao, J., and Chen, L. Seeing beyond classes: Zero-shot grounded situation recognition via language explainer. *arXiv preprint arXiv:2404.15785*, 2024.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *NeurIPS*, volume 33, pp. 9459–9474, 2020.

Li, J., Li, D., Xiong, C., and Hoi, S. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, pp. 12888–12900, 2022.

Li, J., Li, D., Savarese, S., and Hoi, S. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023a.

Li, K., He, Y., Wang, Y., Li, Y., Wang, W., Luo, P., Wang, Y., Wang, L., and Qiao, Y. Videochat: Chat-centric video understanding. *arXiv preprint arXiv:2305.06355*, 2023b.

Li, L., Chen, G., Xiao, J., and Chen, L. Compositional zero-shot learning via progressive language-based observations. *arXiv preprint arXiv:2311.14749*, 2023c.

Li, L., Xiao, J., Chen, G., Shao, J., Zhuang, Y., and Chen, L. Zero-shot visual relation detection via composite visual cues from large language models. In *NeurIPS*, 2023d.

Liang, Y., Wu, C., Song, T., Wu, W., Xia, Y., Liu, Y., Ou, Y., Lu, S., Ji, L., Mao, S., et al. Taskmatrix. ai: Completing tasks by connecting foundation models with millions of apis. *arXiv preprint arXiv:2303.16434*, 2023.

Lin, K. Q., Wang, J., Soldan, M., Wray, M., Yan, R., XU, E. Z., Gao, D., Tu, R.-C., Zhao, W., Kong, W., et al. Egocentric video-language pretraining. *NeurIPS*, 35:7575–7586, 2022.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755. Springer, 2014.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023a.

Liu, R., Wang, X., Wang, W., and Yang, Y. Bird's-eye-view scene graph for vision-language navigation. In *ICCV*, pp. 10968–10980, 2023b.

Liu, R., Wang, W., and Yang, Y. Volumetric environment representation for vision-language navigation. In *CVPR*, 2024.

Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Li, C., Yang, J., Su, H., Zhu, J., et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023c.

Lu, J., Batra, D., Parikh, D., and Lee, S. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *NeurIPS*, 32, 2019.

Lu, P., Peng, B., Cheng, H., Galley, M., Chang, K.-W., Wu, Y. N., Zhu, S.-C., and Gao, J. Chameleon: Plug-and-play compositional reasoning with large language models. *arXiv preprint arXiv:2304.09842*, 2023.

Maggiolino, G., Ahmad, A., Cao, J., and Kitani, K. Deep oc-sort: Multi-pedestrian tracking by adaptive re-identification. *arXiv preprint arXiv:2302.11813*, 2023.

Marino, K., Rastegari, M., Farhadi, A., and Mottaghi, R. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *CVPR*, pp. 3195–3204, 2019.

Marino, K., Chen, X., Parikh, D., Gupta, A., and Rohrbach, M. Krisp: Integrating implicit and symbolic knowledge for open-domain knowledge-based vqa. In *CVPR*, pp. 14111–14121, 2021.

Miao, B., Bennamoun, M., Gao, Y., and Mian, A. Spectrum-guided multi-granularity referring video object segmentation. In *ICCV*, pp. 920–930, 2023.

Mohtashami, A. and Jaggi, M. Landmark attention: Random-access infinite context length for transformers. *arXiv preprint arXiv:2305.16300*, 2023.

Mu, J., Li, X. L., and Goodman, N. Learning to compress prompts with gist tokens. *arXiv preprint arXiv:2304.08467*, 2023.

Nakano, R., Hilton, J., Balaji, S., Wu, J., Ouyang, L., Kim, C., Hesse, C., Jain, S., Kosaraju, V., Saunders, W., et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Newell, A., Simon, H. A., et al. *Human problem solving*, volume 104. 1972.

OpenAI. Introducing chatgpt. *OpenAI Blog*, 2021.

OpenAI. New and improved embedding model. https://openai.com/blog/new-and-improved-embedding-model, 2022.

OpenAI. Gpt-4 technical report. *OpenAI Blog*, 2023.

PaddlePaddle. Paddleocr. https://github.com/PaddlePaddle/PaddleOCR, 2023.

Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–22, 2023.

Peng, B., Galley, M., He, P., Cheng, H., Xie, Y., Hu, Y., Huang, Q., Liden, L., Yu, Z., Chen, W., et al. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*, 2023a.

Peng, B., Quesnelle, J., Fan, H., and Shippole, E. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023b.

Qin, Y., Hu, S., Lin, Y., Chen, W., Ding, N., Cui, G., Zeng, Z., Huang, Y., Xiao, C., Han, C., et al. Tool learning with foundation models. *arXiv preprint arXiv:2304.08354*, 2023.

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., and Sutskever, I. Robust speech recognition via large-scale weak supervision. In *ICML*, pp. 28492–28518, 2023.

Ratner, N., Levine, Y., Belinkov, Y., Ram, O., Magar, I., Abend, O., Karpas, E., Shashua, A., Leyton-Brown, K., and Shoham, Y. Parallel context windows for large language models. In *ACL*, pp. 6383–6402, 2023.

Ruan, J., Chen, Y., Zhang, B., Xu, Z., Bao, T., Du, G., Shi, S., Mao, H., Zeng, X., and Zhao, R. Tptu: Task planning and tool usage of large language model-based ai agents. *arXiv preprint arXiv:2308.03427*, 2023.

Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. A simple neural network module for relational reasoning. *NeurIPS*, 30, 2017.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

Seo, S., Lee, J.-Y., and Han, B. Urvos: Unified referring video object segmentation network with a large-scale benchmark. In *ECCV*, pp. 208–223. Springer, 2020.

Shen, Y., Song, K., Tan, X., Li, D., Lu, W., and Zhuang, Y. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*, 2023.

Shi, F., Chen, X., Misra, K., Scales, N., Dohan, D., Chi, E. H., Schärli, N., and Zhou, D. Large language models can be easily distracted by irrelevant context. In *ICML*, pp. 31210–31227, 2023.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K. R., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*, 2023.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Smith, L., Kew, J. C., Peng, X. B., Ha, S., Tan, J., and Levine, S. Legged robots that keep on learning: Fine-tuning locomotion policies in the real world. In *ICRA*, pp. 1593–1599, 2022.

Sun, R., Arik, S. O., Nakhost, H., Dai, H., Sinha, R., Yin, P., and Pfister, T. Sql-palm: Improved large language modeladaptation for text-to-sql. *arXiv preprint arXiv:2306.00739*, 2023.

Surís, D., Menon, S., and Vondrick, C. Vipergpt: Visual inference via python execution for reasoning. In *ICCV*, 2023.

Tapaswi, M., Zhu, Y., Stiefelhagen, R., Torralba, A., Urtasun, R., and Fidler, S. Movieqa: Understanding stories in movies through question-answering. In *CVPR*, pp. 4631–4640, 2016.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Tsimpoukelli, M., Menick, J. L., Cabi, S., Eslami, S., Vinyals, O., and Hill, F. Multimodal few-shot learning with frozen language models. volume 34, pp. 200–212, 2021.

Vodopivec, T., Samothrakis, S., and Ster, B. On monte carlo tree search and reinforcement learning. *Journal of Artificial Intelligence Research*, 60:881–936, 2017.

Wang, J., Chen, D., Luo, C., Dai, X., Yuan, L., Wu, Z., and Jiang, Y.-G. Chatvideo: A tracklet-centric multimodal and versatile video understanding system. *arXiv preprint arXiv:2304.14407*, 2023a.

Wang, J., Liu, Z., Zhao, L., Wu, Z., Ma, C., Yu, S., Dai, H., Yang, Q., Liu, Y., Zhang, S., et al. Review of large vision models and visual prompt engineering. *arXiv preprint arXiv:2307.00855*, 2023b.

Wang, Y. and Chiew, V. On the cognitive process of human problem solving. *Cognitive systems research*, 11(1):81–92, 2010.

Wang, Y., Li, K., Li, Y., He, Y., Huang, B., Zhao, Z., Zhang, H., Xu, J., Liu, Y., Wang, Z., et al. Internvideo: General video foundation models via generative and discriminative learning. *arXiv preprint arXiv:2212.03191*, 2022.

Wang, Z., Cai, S., Liu, A., Ma, X., and Liang, Y. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. In *NeurIPS*, 2023c.

Wong, B., Chen, J., Wu, Y., Lei, S. W., Mao, D., Gao, D., and Shou, M. Z. Assistq: Affordance-centric question-driven task completion for egocentric assistant. In *ECCV*, pp. 485–501, 2022.

Wu, B., Yu, S., Chen, Z., Tenenbaum, J. B., and Gan, C. Star: A benchmark for situated reasoning in real-world videos. In *NeurIPS*, 2021.

Wu, C., Yin, S., Qi, W., Wang, X., Tang, Z., and Duan, N. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*, 2023a.

Wu, D., Wang, T., Zhang, Y., Zhang, X., and Shen, J. Onlinerefer: A simple online baseline for referring video object segmentation. In *ICCV*, pp. 2761–2770, 2023b.

Wu, J., Jiang, Y., Sun, P., Yuan, Z., and Luo, P. Language as queries for referring video object segmentation. In *CVPR*, pp. 4974–4984, 2022a.

Wu, Y., Rabe, M. N., Hutchins, D., and Szegedy, C. Memorizing transformers. *arXiv preprint arXiv:2203.08913*, 2022b.

Wu, Y., Zhao, Y., Hu, B., Minervini, P., Stenetorp, P., and Riedel, S. An efficient memory-augmented transformer for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2210.16773*, 2022c.

Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., Zhang, M., Wang, J., Jin, S., Zhou, E., et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.

Xiao, J., Shang, X., Yao, A., and Chua, T.-S. Next-qa: Next phase of question-answering to explaining temporal actions. In *CVPR*, pp. 9777–9786, 2021.

Xiao, J., Zhou, P., Chua, T.-S., and Yan, S. Video graph transformer for video question answering. In *ECCV*, pp. 39–58, 2022.

Xu, Z., Yu, C., Fang, F., Wang, Y., and Wu, Y. Language agents with reinforcement learning for strategic play in the werewolf game. *arXiv preprint arXiv:2310.18940*, 2023.

Yang, A., Miech, A., Sivic, J., Laptev, I., and Schmid, C. Just ask: Learning to answer questions from millions of narrated videos. In *ICCV*, pp. 1686–1697, 2021a.

Yang, Z. and Yang, Y. Decoupling features in hierarchical propagation for video object segmentation. *Advances in Neural Information Processing Systems*, 35:36324–36336, 2022.

Yang, Z., Lu, Y., Wang, J., Yin, X., Florencio, D., Wang, L., Zhang, C., Zhang, L., and Luo, J. Tap: Text-aware pre-training for text-vqa and text-caption. In *CVPR*, pp. 8751–8761, 2021b.

Yang, Z., Li, L., Wang, J., Lin, K., Azarnasab, E., Ahmed, F., Liu, Z., Liu, C., Zeng, M., and Wang, L. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*, 2023.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. In *NeurIPS*, volume 36, 2024.

Ye, L., Rochan, M., Liu, Z., and Wang, Y. Cross-modal self-attention network for referring image segmentation. In *CVPR*, pp. 10502–10511, 2019.

Yu, S., Cho, J., Yadav, P., and Bansal, M. Self-chained image-language model for video localization and question answering. In *NeurIPS*, 2023.

Yuan, H., Zhang, C., Wang, H., Xie, F., Cai, P., Dong, H., and Lu, Z. Plan4mc: Skill reinforcement learning and planning for open-world minecraft tasks. *arXiv preprint arXiv:2303.16563*, 2023.

Zhong, Z., Lei, T., and Chen, D. Training language models with memory augmentation. *arXiv preprint arXiv:2205.12674*, 2022.

Zhou, B., Andonian, A., Oliva, A., and Torralba, A. Temporal relational reasoning in videos. In *ECCV*, pp. 803–818, 2018.

Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., and Ba, J. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022.

Zhu, D., Chen, J., Shen, X., Li, X., and Elhoseiny, M. Minigpt-4: Enhancing vision-language understanding

with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023a.

Zhu, X., Chen, Y., Tian, H., Tao, C., Su, W., Yang, C., Huang, G., Li, B., Lu, L., Wang, X., et al. Ghost in the minecraft: Generally capable agents for open-world enviroments via large language models with text-based knowledge and memory. *arXiv preprint arXiv:2305.17144*, 2023b.

Zhuang, Y., Chen, X., Yu, T., Mitra, S., Bursztyn, V., Rossi, R. A., Sarkhel, S., and Zhang, C. Toolchain*: Efficient action space navigation in large language models with a* search. In *ICLR*, 2024.

```
"""
Regarding a given video from {video_filename}, answer the following questions as best you
    can. You have access to the following tools:
{tool_descriptions}
Use the following format:
Question: the input question you must answer
Thought: you should always think about what to do
Action: the action to take, should be one of [{tool_names}]
Action Input: the input to the action
Observation: the result of the action
... (this Thought/Action/Action Input/Observation can repeat N times)
Thought: I now know the final answer
Final Answer: the final answer to the original input question
Begin!
Question: {input_question}
{ancestor_history}
Thought: {expansion_prompt} {agent_scratchpad}
"""
```

Figure 6. The in-context prompt of the MCTS planner (§A.1).

A. Appendix

The appendix is organized as follows:

- §A.1 offers more implementation details of the MCTS planner.
- §A.2 introduces several in-the-wild examples.
- §A.3 provides inference results on NExT-QA (Xiao et al., 2021) dataset.
- §A.4 discusses used foundation models.
- §A.5 analyzes time of inference and efficiency of token usage.
- §A.6 provides typical failure cases.
- §A.7 discusses our limitations.

A.1. Implementation Details of MCTS Planner

Fig. 6 shows the in-context prompt used in the LLMs of our MCTS planner. By changing the placeholders in the form like {*placeholder*}, the prompt can be adapted to complete **branch expansion** or *chain execution*. The meaning of each placeholder in the prompt is listed below:

- {*video_filename*}: the file path of the input video.
- {*input_question*}: the given question/task regarding the given video.
- {*tool_names*}: the names of tools that can be called by the planner, including sub-task tools, knowledge tools, and utility tools.
- {*tool_descriptions*}: the descriptions of all the callable tools' functions and input format. For example, the description of our *What* sub-task tool is *"Useful when you need to describe the content of a video.....The input to this tool must be a string for the video path and a string for the question. For example: inputs is ./videos/xxx.mp4#What's in the video?"*.
- {*agent_scratchpad*}: the place to put the intermediary output during executing a ReAct (Yao et al., 2022) step.
- {*ancestor_history*}: the place to put the history of all the ancestor nodes. For example, when selecting a non-root node for *branch expansion*, the action history (which is a string in the form of ⟨*thought, action, action input, observation*⟩ for each node) of all the ancestor nodes of this non-root node will be put in {*ancestor_history*}.

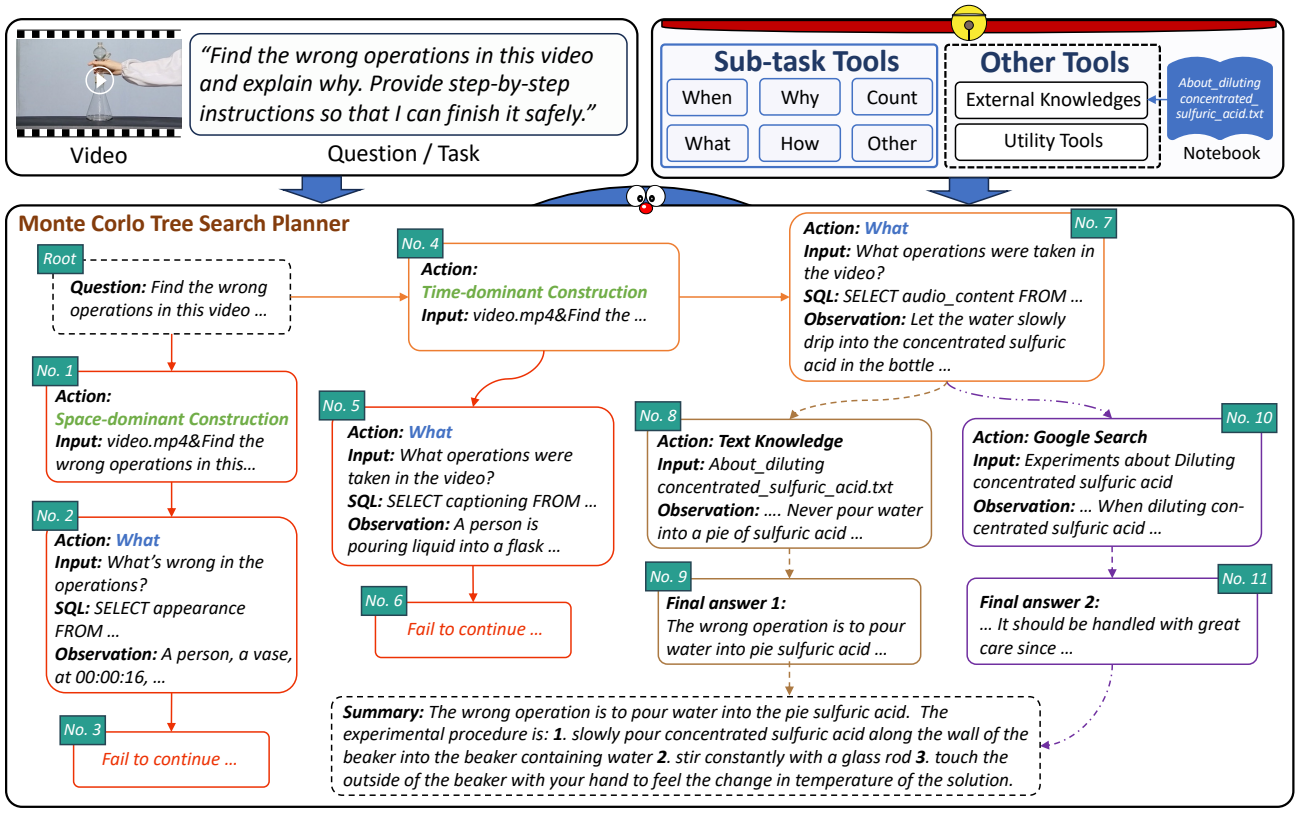


Figure 7. An in-the-wild example of DoraemonGPT (§3.4). Given a video input and a question, our system automatically explores the solution space powered by MCTS planner and various tools. This figure demonstrates both the utilized tools, and the result of intermediate steps during the exploration. Ta-king advantage of the tree-like exploration paths, DoraemonGPT can not only summarize collected answers into a better one, but also has the potential to generate multiple potential solutions.

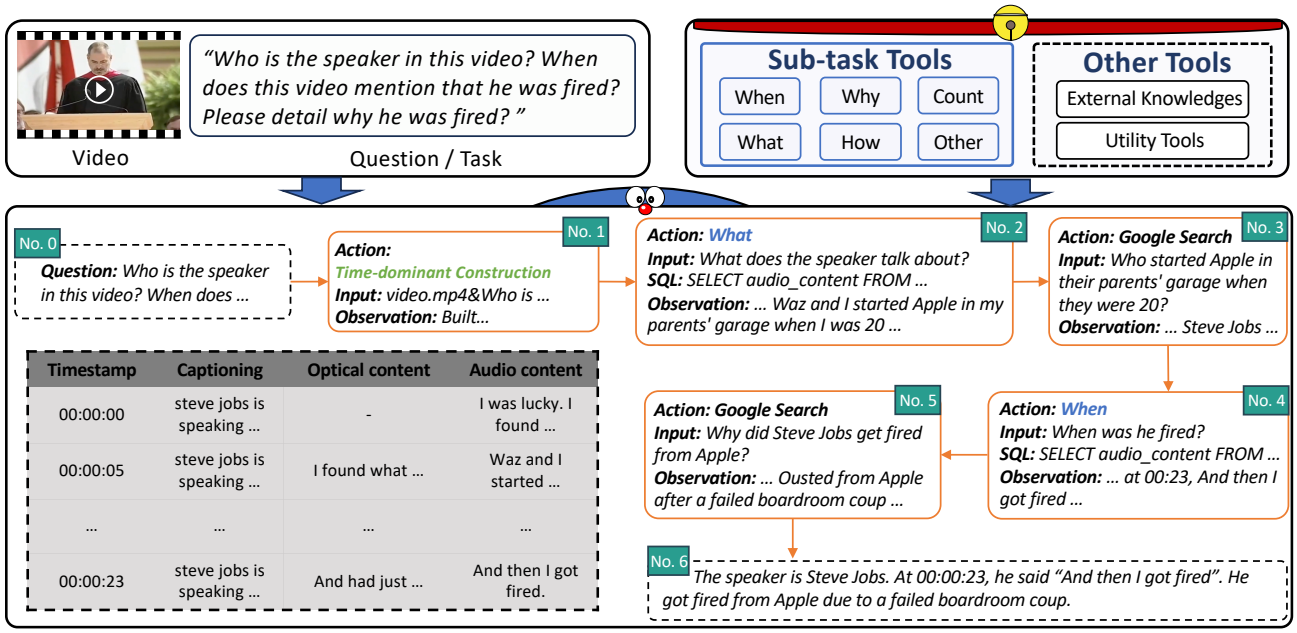
- {*expansion_prompt*}: the place to put the history of all the child nodes for expanding a node, *e.g.*, *"I have thought about the next action before, such as.....I want to think out a different action."*. Only useful in the *branch expansion* phase, set to an empty string for *chain execution*.

The construction of a task-related knowledge base is also based on in-context learning. Concretely, we provide DoraemonGPT with in-context descriptions, usage, and examples of the two types of task-related symbolic memory, supplemented by the input question. The prompt is similar to Fig. 6, but the {*tool_descriptions*} is replaced with a {*memory_descriptions*}. As such, DoraemonGPT can autonomously construct one or both of the task-related knowledge bases. The attributes in each task-related symbolic memory are manually designed.

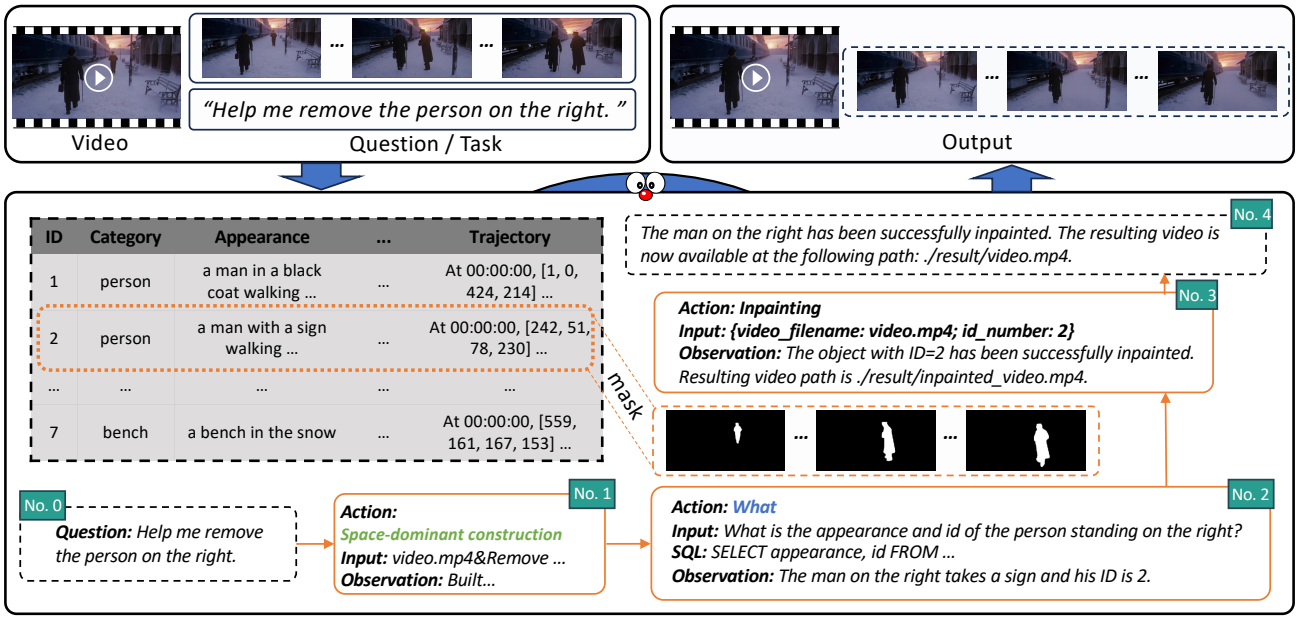
A.2. In-the-wild Examples

In Fig. 7, we visualize the reasoning paths of an in-the-wild example. As depicted, DoraemonGPT is asked to check the experimental operations shown in the video and tell the user how to finish it step by step. Specifically, DoraemonGPT first makes two failed attempts at the beginning, *i.e.*, the queried SQL table or symbolic memory does not contain the information related to the sub-question. Regarding the relevant parts, DoraemonGPT understands the experimental operations shown in the video after expanding a new tree branch by querying a different sub-question. Then, there are two alternative ways to get the final answer, *i.e.*, through the textual notebook provided by users or through a search engine. These paths generate two relevant but different final answers, which can be further summarized into a better, more comprehensive answer. Such an exploration process shows our system's ability to handle questions more complex than those constructed in previous studies.

In Fig. 8a, we visualize the reasoning path of a standard video understanding task. As depicted, DoraemonGPT is asked to identify the speaker and analyze information about the dismissal. After several calls to various tools, DoraemonGPT got the right answers. Here, we also visualize the *time-dominant* symbolic memory, which is the pivotal part of data processing in



(a) Video understanding.



(b) Video editing.

Figure 8. In-the-wild examples of DoraemonGPT (§A.2). In the video editing example, the segmentation mask is also visualized.

DoraemonGPT. Combining it with the well-defined symbolic language (SQL) promises transparency and efficiency.

In addition, we demonstrate an example of video editing by integrating a video inpainting tool. In Fig. 8b, DoraemonGPT is asked to recognize the right person and remove it from the video. To accomplish this, DoraemonGPT constructs the *space-dominant* memory that encompasses the segmentation results for each object within the scene. After recognizing the right person, the inpainting tool is successfully called with an input of the unique ID number assigned to the man on the right, which successfully generates the desired video output.

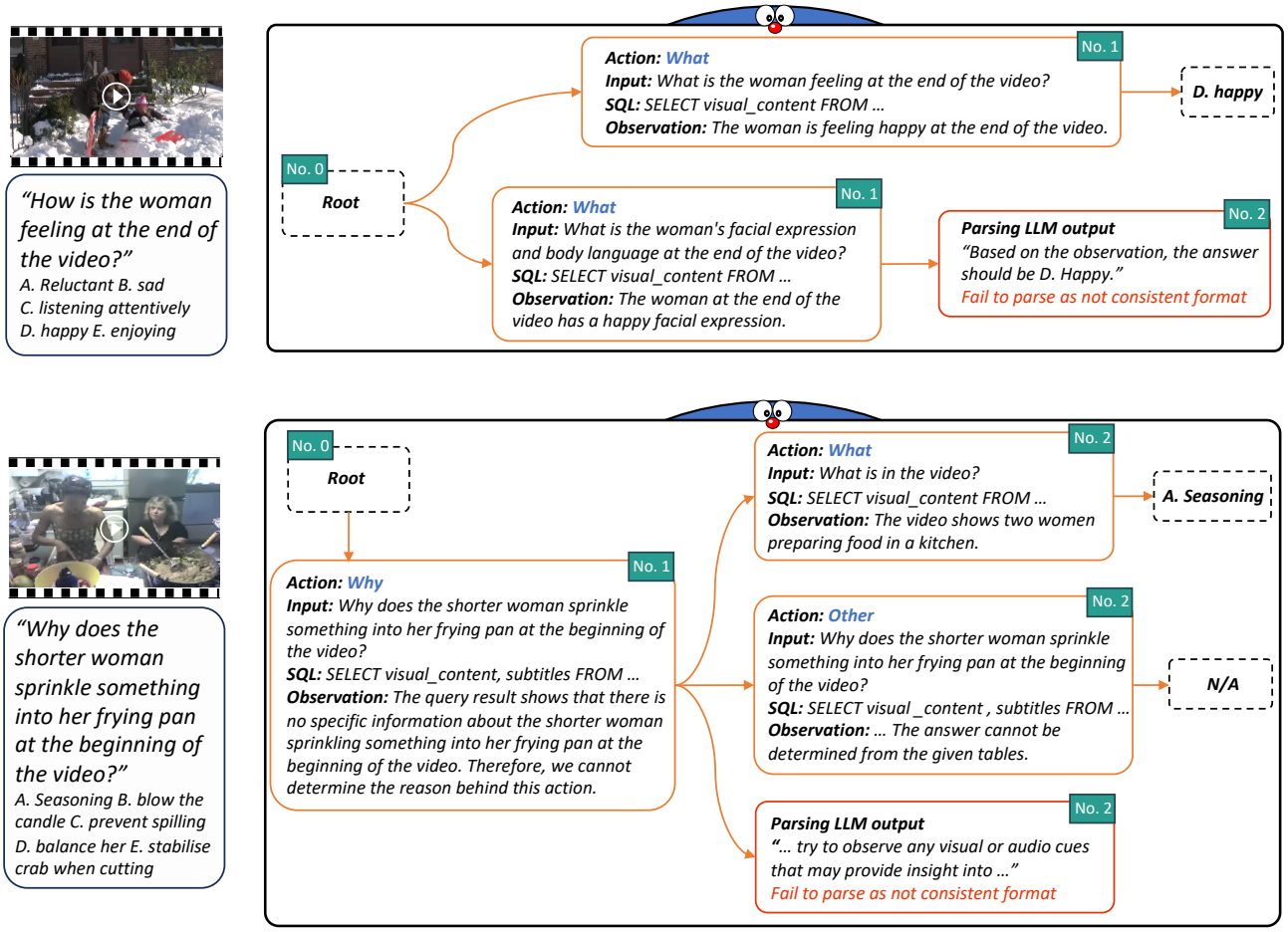


Figure 9. Inference results on NExT-QA (Xiao et al., 2021) (§A.3).

A.3. Inference Results on NExT-QA

Fig. 9 depicts inference results of DoraemonGPT on NExT-QA (Xiao et al., 2021) dataset. From the top part, we have the following findings: (i) A simple question can be finished within a sub-task tool, *e.g.*, using only the What tool can get the correct answer. (ii) The output of LLM that is not formatted may result in an error case, which is very common in current LLM-driven agents. Similar examples can be observed in the bottom part of the same figure.

As shown in the bottom part of Fig. 9, it's quite possible to pick the wrong tool in the early stages of exploration. Our system is able to explore the planning space with multiple branches further. Interestingly, LLM sometimes considers current information insufficient to make a choice. This is tolerated as our system will eventually vote or summarize all candidate answers.

A.4. Discussion on the Impact of Foundation Models

DoraemonGPT leverages foundation models to extract space-dominant and time-dominant information from videos. Hence, the performance of DoraemonGPT is influenced by the quality of these models as well as its own limitations. This impact can be further summarized as follows:

A.4.1. In space-dominant memory

Detection (YOLOv8 (Jocher et al., 2023)): The object categories (COCO (Lin et al., 2014), 80 common categories) are limited by the model, which hinders DoraemonGPT from obtaining information about objects outside these categories. However, YOLOv8 (Jocher et al., 2023) can be replaced with a detection model that supports a wider range of categories

(such as one trained on LVIS (Gupta et al., 2019), with 1000+ categories).

Tracking (Deep OC-sort (Maggiolino et al., 2023)): The current multi-object tracking model is prone to errors in extremely complex scenes (such as those with numerous occluded or similar objects), which affects DoraemonGPT's ability to locate instances in complex videos accurately.

Segmentation (YOLOv8-seg (Jocher et al., 2023)): The segmentation results may not perfectly align with instances' edges, and incomplete segmentation masks can impact the precision of AIGC tools such as video editing (e.g., inpainting).

Referring Object Detection (Grounding DINO (Liu et al., 2023c)): DoraemonGPT harnesses the power of Grounding Dino to detect objects regarding their textual descriptions. For converting the detected bounding boxes into segmentation masks, SAM (Kirillov et al., 2023) is used. However, when facing complex user descriptions that involve multiple features (such as color, location, and size combined) or scenarios with multiple objects having similar semantics, Grounding Dino may encounter challenges in accurately detecting the desired object.

Tracking and Segmentation (DeAoT (Yang & Yang, 2022)): Since the referring detection results of Grounding DINO is image-based instead of video-based, DoraemonGPT employs DeAoT to track the object from the most confident frame to the entire video sequence and segment object masks. By leveraging DeAoT, DoraemonGPT enhances its capabilities in precisely tracking and segmenting objects of interest.

Appearance description (BLIP (Li et al., 2022)/BLIP-2 (Li et al., 2023a)): The textual descriptions cannot accurately capture all the details of an instance (such as intricate clothing details on a human body), which affects DoraemonGPT's handling of tasks related to detailed descriptions.

Action recognition (InternVideo (Wang et al., 2022)): The accuracy is limited by the capabilities of the model, which in turn affects DoraemonGPT's ability to handle action-related inquiries.

A.4.2. IN TIME-DOMINANT MEMORY

Speech recognition (Whisper (Radford et al., 2023)): Current methods can accurately convert audio to text. However, in multi-party conversation scenarios, the methods still cannot accurately perform voiceprint recognition for multiple speakers and accurately separate the results of different speakers. Additionally, it is challenging to match multiple voice prints with the visual IDs of the speakers. This limitation restricts DoraemonGPT's ability to infer and deduce speakers' identities in complex multi-party conversation scenarios, relying solely on the inherent capabilities of LLMs.

Optical character recognition (OCR (PaddlePaddle, 2023)): OCR technology can accurately recognize subtitles and well-structured text. However, it still struggles to handle occluded text and artistic fonts robustly.

Captioning (BLIP (Li et al., 2022)/BLIP-2 (Li et al., 2023a)/InstructBLIP (Dai et al., 2023)): It cannot guarantee that the textual descriptions can accurately cover all the details in the scene, which can affect DoraemonGPT's ability to handle tasks related to detailed descriptions.

Additionally, the domain of the training set for foundation models also affects DoraemonGPT. For instance, currently, visual foundation models trained on real and common scenarios still struggle with extreme lighting conditions or non-realistic scenes (such as simulations or animations).

A.5. Evaluation on the Inference Time and Token Usage Efficiency

For efficiency comparison, we thoroughly analyze the efficiency of DoraemonGPT in comparison with the baselines, ViperGPT and VideoChat. The tables 4 above provide a detailed analysis of the time required for each foundation model used in memory building. When processing videos at a rate of 1 fps, it takes approximately 1 second (or 0.42/0.47s for space/time-dominant memory) to process a 10s video clip using an NVIDIA-A40 GPU. The actual processing time increases linearly with video length.

In comparison, VideoChat creates a time-stamped memory and takes around 2 seconds to process a 10s video at 1 fps. On the other hand, ViperGPT does not construct a memory but generates a code to invoke foundation models. However, there is a 6.7% chance on NExT-QA that ViperGPT fails to generate an executable code, and it's difficult to fairly compare the average time of calling foundation models in ViperGPT.

Due to the influence of simultaneous requests and network delay on ChatGPT's online server, it's impossible to record

Table 4. Token Efficiency (Averaged on the NExT-QA (Xiao et al., 2021) s_val).

Method	Prompt tokens	Node tokens	Steps per Solution	Tokens per Answer	NExT-QA Acc_A
ViperGPT (Surís et al., 2023)	4127	-	-	4127	45.5
VideoChat (Li et al., 2023b)	722	-	-	**722**	51.8
DoraemonGPT	**617**	34.6	2.3	1498	**55.7**

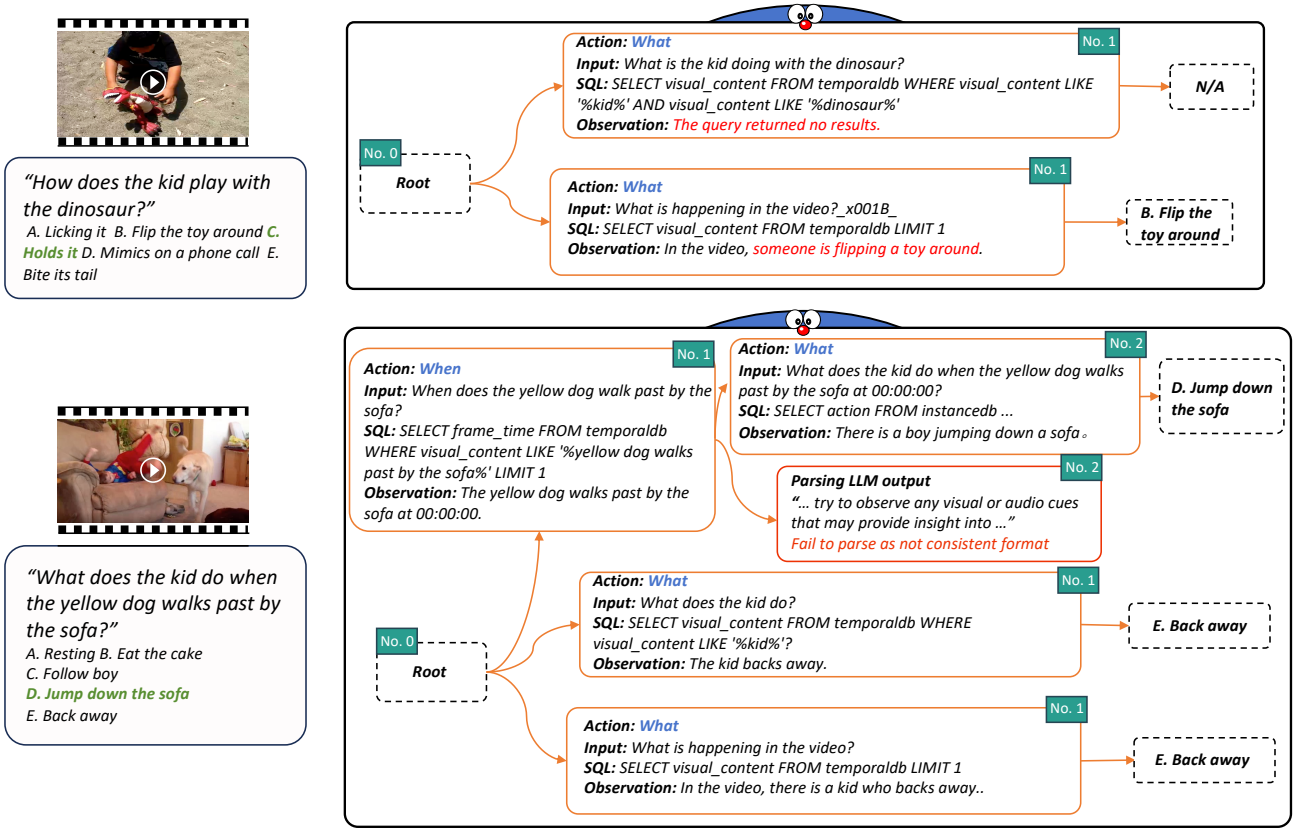


Figure 10. Typical failure cases on NExT-QA (Xiao et al., 2021) (§A.6).

ChatGPT's run-time fairly. Thus, a more equitable efficiency comparison when calling ChatGPT is to record the number of tokens used. As shown in the table above, DoraemonGPT's prompt design is more efficient (617 tokens), which is less than VideoChat's approach of directly incorporating video memory into the prompt (722 tokens) and significantly less than ViperGPT's approach of including a large code definition in the prompt (4127 tokens). Additionally, even though the introduction of our MCTS planner divides the task into multiple nodes/steps, DoraemonGPT still requires far fewer tokens on average to obtain an answer compared to ViperGPT (1498 tokens vs 4127 tokens). Furthermore, DoraemonGPT significantly outperforms VideoChat (55.7 vs 51.8) on the challenging NExT-QA dataset.

A.6. Typical Failure Cases

Here we list two typical failure cases: **i**) The foundation model cannot extract useful information for reasoning (see the top half of Fig. 10). **ii**) The correct answer is identified in the early exploration and the subsequent exploration results in wrong answers (see the bottom half of Fig. 10).

A.7. Limitations

Despite its comprehensive and conceptually elegant system, DoraemonGPT has some limitations for future studies. First, although TSM is a simple and effective way to decouple and handle spatial-temporal reasoning and DoraemonGPT has shown effectiveness with two task-related memory types (*space-dominant* and *time-dominant*), we believe that by further

subdividing the types of tasks, we can introduce more nuanced categories of memory (*e.g.*, human-centric memory) to construct task-related information with greater task-relevance. However, at present, the design of memory types is still a heuristic and manually driven process, lacking an automated design method. Second, the establishment of memory relies on the available foundation models (*e.g.*, BLIP-2 (Li et al., 2023a)). In other words, the performance of foundation models directly influences memory reliability. Incorrect model predictions will introduce noise into the memory, thereby reducing its reliability and affecting the accuracy of decision-making. Additionally, foundation models may struggle to effectively extract the required video attributes in real-world scenarios that are difficult to generalize (*e.g.*, low light, blurriness, occlusions, *etc.*). Third, the accuracy of planning in DoraemonGPT is limited by the capabilities of LLMs. When using a small-scale or insufficiently trained LLM, the likelihood of DoraemonGPT exploring reasonable solutions may be significantly reduced. Last, while the MCTS planner significantly improves the decision-making ability of DoraemonGPT, it also introduces additional computational costs. This means that DoraemonGPT may only be available on high-end computing systems or online LLM services (OpenAI, 2021), limiting its use in real-time, resource-constrained scenarios.