SPROUT: BUILDING DYNAMIC MEMORY WHILE REASONING FOR AGENTIC VIDEO UNDERSTANDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Long video understanding relies on video memory to overcome the context limits of multimodal large language models. Existing methods follow a build-then-reasoning pipeline: memory is built *offline* for the entire video, then reasoned over as a *static* source. In practice a long video is shared by several questions, and this pipeline is costly at both ends: with few questions, building memory for the whole video costs far more than answering them; with many questions, the memory is never updated, so what is learned while answering questions is lost to the next question. To alleviate these, we introduce **Sprout**, an agentic framework that builds memory *while* reasoning: a temporal tree that sprouts detailed nodes as questions are answered. The agent *watches* the video segment by segment at a low frame rate, stopping when the current question can be answered, *remembers* each segment as a coarse node of the tree, and *revisits* key intervals at a higher frame rate to refine the tree with the recovered details. Once a segment is recorded as text, its video input is removed from the context history, while the original video remains reachable through the video tools. The memory tree and prior question–answer records persist across questions, so the memory is online and dynamic: built from the first question onward and updated by every question thereafter. We find that replacing accumulated video inputs with textual memory substantially reduces context usage while maintaining accuracy, with slight improvements in some settings. Across benchmarks on three models, Sprout achieves competitive or improved accuracy relative to representative offline memory methods, with no upfront construction stage and lower context cost per question.

1 INTRODUCTION

Advances in multimodal large language models (MLLMs) (Qwen Team, 2026; Google DeepMind, 2026; Keye Team, 2026) have extended video understanding from short clips to long recordings spanning hours, such as lectures, films, and days of egocentric activity (Fu et al., 2025; Wang et al., 2025a; Yang et al., 2025). Answering questions about such videos requires locating brief moments scattered across time and connecting the evidence among them. As duration grows, however, feeding the video directly into the model becomes infeasible: the number of visual tokens scales linearly with length and quickly exceeds what a context window can hold. Sampling more sparsely fits the budget but discards the brief moments that questions often hinge on, while sampling more densely preserves them at a cost the context cannot absorb. Long video understanding therefore relies on *video memory*: a representation of the video maintained outside the context, from which question-relevant content is retrieved when needed.

Existing methods (QwenLM, 2026; Choi et al., 2026) construct video memory through a *build-then-reasoning* pipeline (Figure 1a): they first build memory *offline*, then reason over the resulting *static* memory with an agentic framework. In the construction stage, the model processes the entire video without knowing future questions and stores it as persistent memory; In the reasoning stage, it retrieves relevant evidence from that memory to answer a given question. Prior work within this pipeline mainly differs in how the offline memory is organized. Hierarchical memories (Yang et al., 2025; Tian et al., 2025) recursively summarize short segments into multi-scale representations, enabling top-down retrieval from coarse summaries to fine segments. Graph-based memories (Long et al., 2026a; Yeo et al., 2026; Chen et al., 2026; QwenLM, 2026) model relations among events, entities, and time, answering questions through navigation and traversal over the resulting structure.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

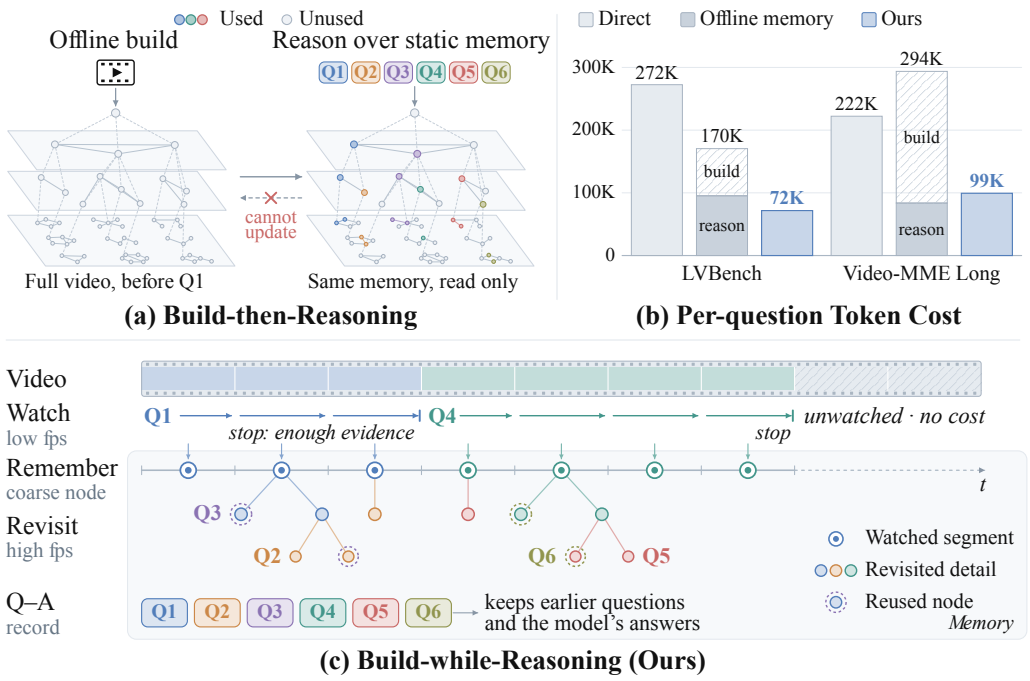


Figure 1: **(a)** Offline memory is first built for the whole video before the first question; Q1–Q6 then reason over the same memory without updating it. Nodes used by a question take that question's color, and the rest are never touched. **(b)** Per-question tokens with Gemini 3.8 Flash as the model. *Direct* feeds the video and question directly to the model. *Offline memory* is Qwen-MM-Plugins: its construction cost, amortized over the benchmark's questions. Our method has no construction stage. **(c)** Sprout builds memory while reasoning. The agent *watches* the video segment by segment at a low frame rate and stops once the question has enough evidence, so the tail of the video is never watched and costs nothing. Each watched segment is *remembered* as a coarse textual node written into memory; *revisits* at a higher frame rate add child nodes beneath the intervals a question needs.

Flat episodic memories (Choi et al., 2026) keep construction minimal by indexing clip captions with multiple retrieval keys and deferring semantic composition to inference time. Despite these differences, all share two properties. First, memory is built offline over the entire video before answering, requiring costly conversion of fine-grained details into memory. Second, memory is static: it is only read during inference and not updated, so the model's observations and conclusions cannot persist across questions. In practice, a long video may be used to answer anywhere from a single question to a dozen or more. This leaves offline, static memory limited at both extremes: 1) upfront construction is wasteful when few questions are asked, since it must record every segment regardless of what is eventually asked; 2) a non-updatable memory prevents useful information from accumulating when many questions are asked, forcing each question to recover the same evidence from scratch.

To address these limitations, we propose a new framework whose memory is built *while* reasoning rather than before it, following two principles. First, the memory is built online from what the model has watched: rather than recording every segment in advance, it grows as the model watches for each question, so no cost is paid for video that no question needs. Second, the memory is refined from coarse to fine: it starts as brief summaries of what happens in each segment, and whenever the model looks closer at a segment to answer a question, it adds the details it finds.

Based on the build-while-reasoning framework, we introduce **Sprout**, an agentic framework whose memory is a temporal tree that sprouts new nodes as questions are answered (Figure 1c), built through three actions: *watch*, *remember*, and *revisit*. *Watch.* Given a question, the agent streams the video segment by segment at a low frame rate, starting from the beginning of videos or where the memory left off, and stops once it judges that the segments the question needs have been found.

Remember. As each segment is watched, the agent writes it into memory as a coarse node with a time interval and an event summary, forming the first level of a temporal tree. Once a segment is recorded, its video input is removed from the context history, allowing the agent to reason without that video chunk and greatly reducing context pressure. *Revisit.* When a retrieved node identifies a relevant video interval but lacks the needed detail, the agent revisits it at a higher frame rate, saves the recovered details as child nodes, and may search memory for related earlier information. The revisited frames are released from the context history in the same way.

Together, the three actions follow the two principles above. Watching and remembering build the memory online from the video the model has actually watched, and keep text rather than video in the context. Revisiting refines coarse nodes into finer ones while the model reasons. As a result, our memory is *online* and *dynamic*: it is built online after the first question arrives, even when the full video is available in advance, and it keeps growing as each question adds nodes, refines existing ones, and leaves its answer behind.

We evaluate Sprout with three models (GPT-5 (OpenAI, 2025), Qwen3.8-Max, and Gemini 3.8 Flash) on LVBench (Wang et al., 2025a), Video-MME Long (Fu et al., 2025), and Video-Holmes (Cheng et al., 2026). Its accuracy is competitive with or better than representative offline memory methods, while it pays no construction cost before the first question. It also uses fewer tokens (Figure 1b): with Gemini 3.8 Flash, it spends 71.7K and 99.3K tokens per question on LVBench and Video-MME Long, 74% and 55% fewer than direct inference.

In summary, our contributions are threefold:

- We identify two limitations of the build-then-reasoning pipeline: offline construction is wasteful for few-question videos, and static memory cannot accumulate insights from earlier reasoning.
- We propose Sprout, an agentic framework that builds memory while reasoning through three actions: it watches the video at a low frame rate, remembers each segment as a coarse node of a temporal tree, and revisits key intervals at a higher frame rate to add finer nodes.
- We further find that removing video frames from the context once they are recorded as text keeps the context small without reducing accuracy. Experiments with three models on benchmarks show that Sprout is more effective than an offline memory method.

2 Related Work

Agentic Multimodal Reasoning. Rather than answering from what is already in the context, agentic multimodal methods call external tools during reasoning to gather the evidence they need. MM-REACT routes a language model to vision experts through a textual interface (Yang et al., 2023), while ViperGPT composes visual and language modules into executable programs (Surís et al., 2023). For video, VideoAgent iteratively gathers relevant visual information under the control of an LLM (Wang et al., 2024). DrVideo converts video into a coarse textual document and augments selected entries with additional visual information as its agent searches for missing evidence (Ma et al., 2025). More recently, VideoSeek provides overview, skim, and focus tools, allowing an agent to adapt its inspection granularity while reasoning over accumulated conversation history (Lin et al., 2026a). VideoSearcher instead trains the tool-invocation policy itself with reinforcement learning, unifying temporal localization, spatial focusing, and external multimodal search for open-world video questions (Gao et al., 2026). These methods show that tool use during reasoning can improve multimodal performance. More broadly, reliable multimodal reasoning rests on grounding outputs in explicit visual evidence: models comprehend relations among objects (Li et al., 2026), keep interleaved image–text content coherent (Chen et al., 2025a), suppress hallucinations unsupported by the image (Chen et al., 2025b), and ground reward judgments in fine-grained, region-level visual evidence (Long et al., 2026b; Yang et al., 2026b;a). Sprout connects this tool use to persistent memory management: each inspection updates a temporal node, its video inputs can then leave the context history, and the recorded information remains available to subsequent questions.

Efficient Long Video Understanding. Long video understanding requires balancing temporal coverage, visual detail, and input cost, as reflected in Video-MME, LVBench, and EgoLife (Fu et al., 2025; Wang et al., 2025a; Yang et al., 2025). One direction expands the amount of visual information a model can process: LongVILA combines context extension, long-video instruction tuning, and sequence parallelism to support longer video inputs (Chen et al., 2025c), and Keye-VL-2.0

adapts sparse attention to a multimodal backbone to process hour-level videos within a 256K-token context (Keye Team, 2026). Another direction reduces the representation size: LongVU adaptively compresses spatial and temporal visual tokens to fit long videos within a limited context (Shen et al., 2025a). At inference time, VideoTree allocates representation detail according to query relevance through adaptive expansion of a video tree (Wang et al., 2025b). Together, these approaches improve the trade-off between coverage and detail through model capacity, compression, or evidence selection. Our focus is how observation density and retained context evolve over an answering session. Sprout first builds coarse temporal nodes from low-frame-rate observations, then revisits relevant intervals at higher frame rates and writes the recovered details back to those nodes. Textual memory carries these observations into later reasoning steps, allowing the corresponding visual inputs to be released from the context history while the source video remains accessible.

Video Memory and Retrieval. Long-video methods keep a memory outside the context and answer by retrieving from it. This memory follows a build-then-reasoning pipeline: it is built before any question, in one pass or incrementally as the video arrives, and it stays *static* while questions are answered. Streaming methods such as MA-LMM and M3-Agent are the incremental case: their memory grows as frames arrive, but it is still formed without questions and only read at answer time (He et al., 2024; Long et al., 2026a). Within this pipeline, methods differ in how the memory is organized. Hierarchical memories summarize segments at multiple temporal scales and answer by top-down retrieval (Yang et al., 2025; Tian et al., 2025). Graph-based memories link events, entities, and time, then answer by search or traversal: WorldMM, MemDreamer, and Qwen-MM-Plugins retrieve over multi-scale or hierarchical graphs (Yeo et al., 2026; Chen et al., 2026; QwenLM, 2026); Vgent and EGAgent add verification or entity search on the graph (Shen et al., 2025b; Rege et al., 2026); HippoMM consolidates short-term observations into longer-term memory (Lin et al., 2026b). MERIT keeps the memory flat, indexing clip captions under multiple keys and assembling evidence at inference time (Choi et al., 2026). Caption-based methods such as LLoVi, Video-RAG, and iRAG let the question decide which text is fetched or refined (Zhang et al., 2024; Luo et al., 2025; Arefeen et al., 2024). Sprout builds the memory while reasoning, adding and refining nodes only where questions lead and keeping those nodes and earlier answers for later questions.

3 METHOD

In this section, we present Sprout, an agentic framework that builds video memory while reasoning. Unlike build-then-reasoning methods, which construct memory over the entire video before any question arrives, Sprout starts from an empty memory and builds it only while answering questions. Given a video V of duration T and questions $q_1, \ldots, q_K$ about it, the model answers each question over multiple rounds, calling tools (Appendix A) that perform three actions: *watch* attaches the next unwatched chunk of the video at a low frame rate, *remember* writes it into memory as text, and *revisit* watches a recorded interval again at a higher frame rate to add details. The memory is shared by all questions about the video, so later questions can reuse what earlier ones have built. Figure 2 illustrates this process on three questions.

3.1 DYNAMIC TEMPORAL MEMORY

The memory has three parts: a temporal tree of observations, a watch record of how far the video has been watched, and a question history of earlier questions and their answers.

Temporal tree. The tree is organized by time. Its root spans the whole video $[0, T]$. Every other node covers an interval $[s_i, e_i] \subseteq [0, T]$ and stores an event summary, observed facts, and key entities for that interval; the children of a node split its interval into shorter pieces in temporal order. Because each node keeps its interval, the video behind any node can be watched again. The tree grows with the three actions: watching and remembering add coarse nodes under the root in temporal order, and revisiting adds children under a node where a question needs more detail. At any moment, the tree is therefore deep only where questions have looked closely, and it ends where the unwatched video begins. Nodes store only what was observed in the video; questions and answers are kept in the question history, so that the nodes remain useful to unrelated later questions.

Access. The model does not see the tree directly. Its context holds the current question, the watch record, and a compact list of the question history; the content of the tree is reached only through

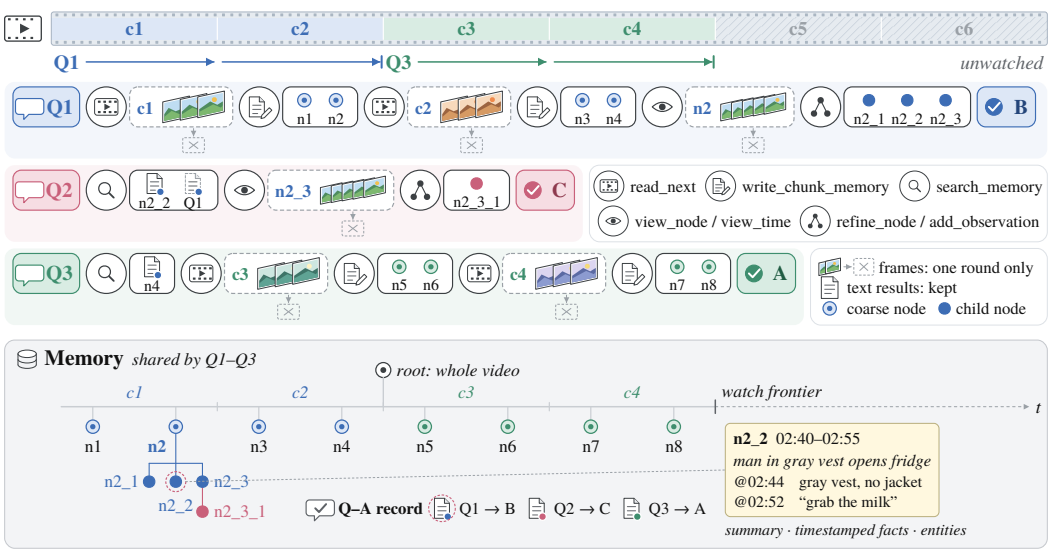


Figure 2: **Pipeline of Sprout.** Three questions about one video are answered in order, building one memory. Each row shows the rounds of one question: tool calls (circles) and their results (boxes). Video frames stay in the context for one round only; text results are kept. **Q1** watches the first two chunks, remembers them as coarse nodes, and revisits n_2 at a higher frame rate to add finer child nodes. **Q2** reuses the memory: search returns a node and Q1's answer, and a short revisit adds one more detail, without watching new video. **Q3** finds that the memory does not cover what it asks, so it watches two more chunks. The last two chunks are never watched. The memory tree (bottom) is drawn on the same time axis as the video; node colors show which question added each node.

search (Section 3.4). All writes go through tools that check the new nodes against the current tree, so that at every level the tree stays in chronological order within the watched video.

3.2 Watch: Coarse Temporal Exploration

The agent watches new video with `read_next`, either when the memory is empty, as at the first question, or when the memory does not yet cover what the question asks about. The tool takes no arguments: the runtime selects the next unwatched chunk from the watch record and attaches its frames at a low rate f_c, together with the speaker-labelled transcript of the chunk when audio is available. At this rate, the agent learns what happens and when at a small visual cost.

How far to watch is decided by the current question. After each chunk is remembered (Section 3.3), the agent has three options: watch the next chunk, revisit an interval that looks relevant, or answer. Watching can therefore stop before the end of the video, and dense revisits can already happen during the first question. Everything watched stays in memory for later questions.

3.3 Remember: Recording Observations and Releasing Frames

Each watched chunk is written into memory right away. After `read_next`, the chunk is *pending*: the agent must store it with `write_chunk_memory` in the next round before it can watch another chunk. The call passes a list of segments, each with an interval, a summary, and a set of facts. The runtime appends them as coarse nodes under the root and advances the watch record to the end of the chunk. The agent is instructed to write about 4–8 segments per chunk with 5–10 objective facts each, and to record only what is in the video, never the question, its options, or its reasoning.

Releasing frames. Once the model has responded to a chunk, its frames are removed from the context history before the next model call; the textual tool results and the model's responses are kept. Frames therefore never pile up in the context: each chunk costs one call with frames and one text-only call to store it. Frames from revisits are removed in the same way.

To see the saving, consider t observations with $v_1, \ldots, v_t$ visual tokens and a textual context of length B_t. Keeping all frames and releasing them after use give context lengths

$$L_t^{\text{retain}} = B_t + \sum_{j=1}^{t} v_j, \qquad L_t^{\text{release}} = B_t + v_t. \tag{1}$$

Each observation is still paid for once, but only the latest one stays in the context.

3.4 Revisit: Retrieval, Dense Inspection, and Write-Back

Revisiting lets the agent return to video it has already watched. It has three steps: retrieve relevant nodes from the memory, watch their intervals at a higher frame rate, and write new details back.

Retrieval. Since the model does not see the tree, it searches the memory with `search_memory`, which returns text only. A query is a set of literal, case-insensitive alternatives (e.g., `"vest | waistcoat"`), a time window, or both; a window alone returns everything stored in that span (Appendix B compares this literal matching with embedding retrieval). The result lists the matching facts with their timestamps under their nodes, followed by entries of the question history whose question mentions the query, so earlier answers come back together with the observations behind them. The agent is instructed to search before answering detail questions, and the number of searches per question is capped.

Dense inspection. When a retrieved node points to the right event but lacks the detail the question needs, the agent watches that part of the video again. `view_node` attaches the whole interval of a node, and `view_time` attaches an explicit interval inside the watched video, which the runtime clamps to the node that contains it. Both tools take a *focus* describing what to look for and a frame rate $f_r > f_c$ chosen by the agent, so dense frames are spent only on the chosen interval. This can reveal brief events or intermediate actions missed at the low rate.

Write-back. Each inspection tool is paired with a write-back tool that accepts only the interval just viewed. `refine_node` splits the node just viewed with `view_node` into contiguous children in temporal order that cover its interval. This is allowed only for a leaf, so an internal node can serve as evidence but its subtree is never rewritten. `add_observation` stores a summary and facts for the interval just viewed with `view_time` as a child of the node. In both cases, the parent keeps its description, so coarse context and new detail are kept side by side, and the revisited frames are removed as in Section 3.3. If the evidence is still not enough, the agent can search again, revisit other nodes, or watch more video. A question ends when the model calls `answer` or reaches the maximum number of rounds, in which case it must answer with the evidence it has.

3.5 Accumulation Across Questions

After a question is answered, the tree and the watch record are kept for the next question. The `answer` call also appends the question, its options, and the predicted answer to the question history; later questions see these entries in compact form and can retrieve them with `search_memory`. A new question therefore starts from everything earlier questions have built: it can reuse existing nodes, refine them, or extend the tree into unwatched video. Since all watching and writing happens while questions are answered, the memory covers only what the questions have needed, and each question pays only for the video and detail it adds.

4 Experiments

4.1 Experimental Setup

Benchmarks and metric. We evaluate on three complementary benchmarks: LVBench (Wang et al., 2025a) contains 1,549 questions over 103 hour-scale videos; the Long subset of Video-MME (Fu et al., 2025) contains 900 questions over 300 videos lasting 30–60 minutes; and Video-Holmes (Cheng et al., 2026) contains 1,837 questions over 270 short suspense videos, emphasizing reasoning across scattered visual clues. We also evaluate on EgoLifeQA (Yang et al., 2025), using the 500 questions of one participant (Jake) over a 44.3-hour recording spanning seven days. With

Table 1: **Accuracy (%) on three video benchmarks.** Rows are grouped into direct inference, agentic and memory methods, and Sprout; Video-MME(L) is the Long subset of Video-MME.

Method	Model	LVBench	Video-MME(L)	Video-Holmes
Direct	GPT-5	60.4	74.3	44.1
Direct	Qwen3.8-Max	81.8	80.4	67.3
Direct	Gemini 3.8 Flash	87.1	90.1	72.0
WorldMM (Yeo et al., 2026)	GPT-5	61.9	76.6	–
MERIT (Choi et al., 2026)	GPT-5	71.8	77.7	–
VideoSeek (Lin et al., 2026a)	GPT-5	68.4	81.2	47.3
Qwen-MM-Plugins	GPT-5	73.3	79.2	48.4
Qwen-MM-Plugins	Qwen3.8-Max	85.6	85.8	68.5
Qwen-MM-Plugins	Gemini 3.8 Flash	85.5	89.4	69.2
Sprout (ours)	GPT-5	73.4	80.4	56.7
Sprout (ours)	Qwen3.8-Max	86.9	86.0	71.6
Sprout (ours)	Gemini 3.8 Flash	88.2	91.2	76.8

Table 2: **Accuracy (%) on EgoLifeQA.** Ent., Event, Habit, Rel., and Task denote the EntityLog, EventRecall, HabitInsight, RelationMap, and TaskMaster categories; Overall is the accuracy over all questions, not the mean of the categories.

Method	Model	Ent.	Event	Habit	Rel.	Task	Overall
Direct	Gemini 2.5 Pro	43.2	40.5	41.0	55.2	52.4	46.4
Ego-R1 (Tian et al., 2025)	3B	51.2	53.2	63.9	50.4	50.8	53.0
M3-Agent (Long et al., 2026a)	7B	44.4	54.8	62.3	56.8	54.0	53.5
WorldMM (Yeo et al., 2026)	Qwen3-VL-8B	49.6	56.4	63.9	58.4	58.7	56.4
MERIT (Choi et al., 2026)	Gemini 2.5 Pro	60.8	61.1	65.6	65.6	73.0	64.2
Qwen-MM-Plugins	Qwen3.8-Max	72.0	77.0	80.3	72.8	82.5	75.8
Qwen-MM-Plugins	Gemini 3.8 Flash	76.0	77.0	77.0	75.2	87.3	77.6
Sprout (ours)	Qwen3.8-Max	79.2	77.0	80.3	84.0	84.1	80.6
Sprout (ours)	Gemini 3.8 Flash	76.0	75.4	85.2	86.4	81.0	80.2

hundreds of questions about the same recording, it tests how well the memory is reused. Table 1 reports the first three benchmarks and Table 2 reports EgoLifeQA.

Implementation. We run Sprout with three models: GPT-5, Qwen3.8-Max, and Gemini 3.8 Flash. The memory starts empty for each video and is kept across its questions. Questions arrive one at a time; the agent sees its own earlier answers, but not the ground truth or future questions. On LVBench and Video-MME Long, `read_next` attaches 10-minute chunks at $f_c = 0.1$ fps; GPT-5 uses 5-minute chunks at the same rate to fit the image limit of its endpoint. On Video-Holmes, chunks are 30 seconds long at $f_c = 1$ fps. Revisits choose $f_r \in \{1, 2, 4, 8\}$ fps and attach at most 768, 48, and 2,048 frames per view for Qwen3.8-Max, GPT-5, and Gemini, respectively. For comparison, Direct gives Qwen3.8-Max up to 768 sampled frames on Video-MME Long and 1 fps on Video-Holmes, and gives Gemini up to 3,072 frames. Each question may use at most 32 *rounds*. An ASR transcript is attached to each chunk on Video-MME Long, but not on LVBench or Video-Holmes. As described in Section 3.1, the model does not see the tree: its context shows only the watch record and the question history, and the tree is reached through `search_memory`. We call this setting the *progress exposure*. For Qwen-MM-Plugins, the memory is built once by Qwen3.7-Plus and shared by all three models.

4.2 Main Results

Long videos. Table 1 reports results on LVBench and Video-MME Long. Memory built offline does not always help: with Gemini 3.8 Flash, Qwen-MM-Plugins even falls below Direct. In contrast, Sprout matches or exceeds both with every model, although it builds no memory before the first question. With Gemini 3.8 Flash, it achieves new state-of-the-art results on both benchmarks. The

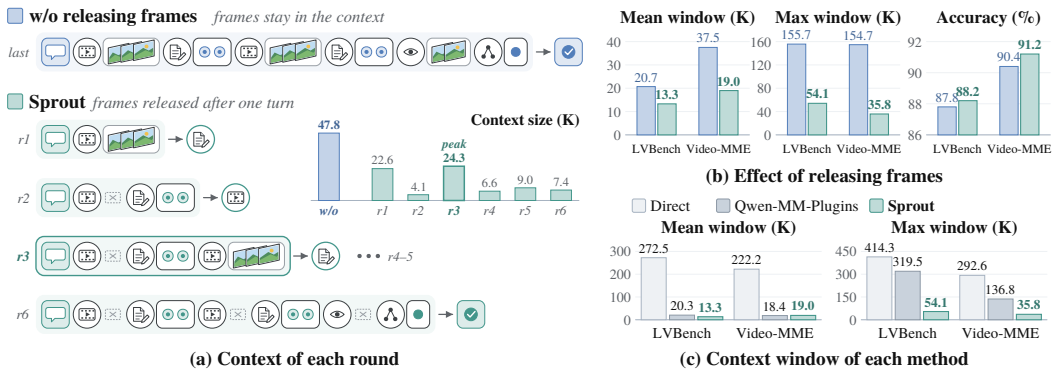


Figure 3: **Effect of releasing frames** with Gemini 3.8 Flash. (a) The context of one LVBench question in each round, followed by the model output of that round; the bar chart gives the measured context size of each round. Without releasing frames, every frame stays, so the context only grows and its last round reaches 47.8K tokens. Sprout releases frames once the model has written them as text, so its context rises and falls: the rounds that watch a chunk (r1, r3) have the most tokens, and even the peak (24.3K) is about half of the variant's last round. (b) Context window and accuracy of the two variants. (c) Context window of Direct, Qwen-MM-Plugins, and Sprout. The context window is the peak context size over all rounds of a question (the single request for Direct), in thousands of tokens (K); we report its mean and maximum over all questions.

gain over Direct is largest with GPT-5 (13.0 points on LVBench) and small with Gemini 3.8 Flash, so the memory helps most when the model cannot use a long context well.

Short videos. The advantage is larger on Video-Holmes, where Sprout achieves the best results with all three models. Offline memory again falls below Direct with Gemini 3.8 Flash, while Sprout exceeds Qwen-MM-Plugins by up to 8.3 points. These results show that Sprout is a general method that applies to both long and short videos.

Day-long video. Table 2 reports results on EgoLifeQA, whose videos are much longer than those above. With both Qwen3.8-Max and Gemini 3.8 Flash, Sprout outperforms Qwen-MM-Plugins overall, and its Qwen3.8-Max run achieves the best result in the table. This shows that Sprout also performs well on ultra-long videos.

4.3 Releasing Frames and Context Window

As described in Section 3.3, Sprout releases frames from the context once they have been recorded as text. We now examine how this affects accuracy and the context window, with Gemini 3.8 Flash as the model. Since Sprout answers a question over multiple rounds, its context size changes from round to round (Figure 3a): a round that carries frames uses more tokens than the rounds after it, so the last round is not necessarily the one with the most tokens. We therefore take the peak context size over all rounds of a question as its context window, and report the mean and maximum of this value over all questions.

Releasing frames. Figure 3b shows that releasing frames does not hurt accuracy. We compare Sprout with a variant *w/o releasing frames*, which keeps every frame in the context until the question ends. Keeping frames enlarges the context window by up to 4.3×. Yet accuracy does not improve. The variant revisits less often because old frames remain visible, but what these frames provide is already recorded in memory or can be recovered by a revisit.

Context window. Releasing frames also keeps the context small throughout a question, as shown in Figure 3c. Direct places the whole sampled video in a single request of more than 200K tokens. Qwen-MM-Plugins keeps all retrieved text and frames across rounds, so its context grows as a question proceeds, and its largest question approaches the size of Direct. In contrast, Sprout watches each video segment only once and then removes it from the context history. Its mean window is similar to that of Qwen-MM-Plugins, but its largest window is several times smaller and stays

Table 3: **Token cost per question** with Gemini 3.8 Flash, in thousands of tokens (K) on LVBench / Video-MME(L), the Long subset of Video-MME. *Construction* builds the memory offline, amortized over the 1,549 and 900 questions of the two benchmarks; *Answering* sums input and output over all rounds of a question; *Total* is their sum, with the lowest in bold.

Method	Construction	Answering	Total
Direct	–	272.5 / 222.2	272.5 / 222.2
Qwen-MM-Plugins	75.2 / 210.0	95.3 / 83.7	170.5 / 293.7
Sprout (ours)	–	71.7 / 99.3	**71.7 / 99.3**

Table 4: **Effect of the question history** with Gemini 3.8 Flash; accuracy (%), with Video-MME(L) the Long subset of Video-MME. **Direct** answers each question independently. **Causal** answers the questions of a video in order: the whole video is attached to the first question, and each later question sees all earlier questions, the model's reasoning, and its answers as text.

Method	LVBench	Video-MME(L)	Video-Holmes
Direct	87.1	90.1	72.0
Causal	87.0 (−0.1)	90.9 (+0.8)	75.9 (+3.9)

below 55K tokens on both benchmarks. Therefore, recording each observation as text and releasing its frames keeps the context small without losing accuracy.

4.4 Cost Analysis

We next measure the total tokens a question uses over all its rounds, with Gemini 3.8 Flash (Table 3); for Qwen-MM-Plugins, we add its offline construction cost, spread over the questions of each benchmark. Sprout uses 74% and 55% fewer tokens than Direct on LVBench and Video-MME Long, since it watches each segment only once at a low frame rate and revisits only the intervals a question needs. Qwen-MM-Plugins answers with about as many tokens as Sprout, but it pays for construction no matter how many questions a video receives: on Video-MME Long, with only 3 questions per video, construction alone costs 210K tokens per question and pushes the total above Direct. In contrast, Sprout builds memory only as questions arrive, so its total cost is the lowest on both benchmarks.

4.5 Effect of the Question History

Sprout keeps the questions of a video and its own answers in the question history, which later questions can see. To test whether this history helps on its own, we separate it from the memory tree (Table 4): Direct answers each question independently, while Causal answers the questions of a video in order and shows each later question the earlier questions, reasoning, and answers as text. Although these earlier answers may be wrong, Causal does not hurt accuracy. On Video-Holmes, where questions about one video build on the same clues, accuracy improves by 3.9 points; on LVBench and Video-MME Long, where questions are largely unrelated, the differences stay within 1 point. Earlier answers thus help when questions share content and do no harm otherwise, so keeping the question history is worth its small text cost.

5 Conclusion

We presented Sprout, an agentic framework that builds video memory while reasoning instead of before it. The agent watches the video in coarse chunks, remembers each chunk as a node of a temporal tree, and revisits the intervals a question needs at a higher frame rate, writing the recovered details back into the tree. Frames are released once they are recorded as text, and the tree and the question–answer records persist across questions, so the memory is built from the first question onward and refined by every question thereafter. Across three benchmarks and three models, Sprout matches or exceeds offline memory methods without an upfront construction stage, keeps the context window bounded by a single observation, and spends fewer tokens per question on long videos. Our experiments also show that releasing frames after use costs no accuracy and that earlier answers help later questions when they share content. Future work includes richer retrieval over the tree, such as multi-key indexing, and mechanisms for merging or forgetting nodes, which the current memory, like existing offline memories, does not have.

AI USE STATEMENT

We used generative AI tools to aid and polish the writing of the manuscript, to draft and edit the figures and tables, and to retrieve, survey, and organize related work. We have reviewed all AI-assisted work. The related-work compilation was checked by the authors against the original papers, and all AI-assisted text, figures, and tables were revised and verified by the authors.

REFERENCES

Md Adnan Arefeen, Biplob Debnath, Md Yusuf Sarwar Uddin, and Srimat Chakradhar. iRAG: Advancing RAG for videos with an incremental approach. In *CIKM*, pp. 4341–4348, 2024.

Cong Chen, Guo Gan, Kaixiang Ji, ZhaoYang Zhang, Zhen Yang, Guangming Yao, Hao Chen, Jingdong Chen, Yi Yuan, and Chunhua Shen. MemDreamer: Decoupling perception and reasoning for long video understanding via hierarchical graph memory and agentic retrieval mechanism. *arXiv preprint arXiv:2606.07512*, 2026.

Wei Chen, Lin Li, Yongqi Yang, Bin Wen, Fan Yang, Tingting Gao, Yu Wu, and Long Chen. CoMM: A coherent interleaved image-text dataset for multimodal understanding and generation. In *CVPR*, pp. 8073–8082, 2025a.

Wei Chen, Xin Yan, Bin Wen, Fan Yang, Tingting Gao, Di Zhang, and Long Chen. Decoupling contrastive decoding: Robust hallucination mitigation in multimodal large language models. In *NeurIPS*, 2025b.

Yukang Chen, Fuzhao Xue, Dacheng Li, Qinghao Hu, Ligeng Zhu, Xiuyu Li, Yunhao Fang, Haotian Tang, Shang Yang, Zhijian Liu, Ethan He, Hongxu Yin, Pavlo Molchanov, Jan Kautz, Linxi Fan, Yuke Zhu, Yao Lu, and Song Han. LongVILA: Scaling long-context visual language models for long videos. In *ICLR*, 2025c.

Junhao Cheng, Yuying Ge, Teng Wang, Yixiao Ge, Jing Liao, and Ying Shan. Video-Holmes: Can MLLM think like Holmes for complex video reasoning? In *ECCV*, 2026.

Yeeun Choi, Youngbeom Yoo, Joon-Young Lee, Hyolim Kang, and Seon Joo Kim. Keep it simple: Multi-key episodic memory retrieval for ultra-long video understanding. In *ECCV*, 2026.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, Peixian Chen, Yanwei Li, Shaohui Lin, Sirui Zhao, Ke Li, Tong Xu, Xiawu Zheng, Enhong Chen, Caifeng Shan, Ran He, and Xing Sun. Video-MME: The first-ever comprehensive evaluation benchmark of multi-modal LLMs in video analysis. In *CVPR*, pp. 24108–24118, 2025.

Zhenkun Gao, Yicheng Bao, Jinlong Peng, Xueheng Li, Theo Huang, Bangwei Liu, Kunquan Li, Zhenye Gan, Tao Hu, Chengjun Xie, Mingqian Yang, Xuanhua He, Zhizhong Zhang, Xin Tan, Chengjie Wang, and Yuan Xie. VideoSearcher: Empowering video deep research with multi-tool agentic reasoning via reinforcement learning. *arXiv preprint arXiv:2607.02927*, 2026.

Google DeepMind. Gemini 3.8 Flash. Model card, 2026. URL https://deepmind.google/models/model-cards/gemini-3-8-flash/.

Bo He, Hengduo Li, Young Kyun Jang, Menglin Jia, Xuefei Cao, Ashish Shah, Abhinav Shrivastava, and Ser-Nam Lim. MA-LMM: Memory-augmented large multimodal model for long-term video understanding. In *CVPR*, pp. 13504–13514, 2024.

Keye Team. Kwai Keye-VL-2.0 technical report. *arXiv preprint arXiv:2606.10651*, 2026.

Lin Li, Wei Chen, Jiahui Li, Kwang-Ting Cheng, and Long Chen. Relation-R1: Progressively cognitive chain-of-thought guided reinforcement learning for unified relation comprehension. In *AAAI*, 2026.

Jingyang Lin, Jialian Wu, Jiang Liu, Ximeng Sun, Ze Wang, Xiaodong Yu, Jiebo Luo, Zicheng Liu, and Emad Barsoum. VideoSeek: Long-horizon video agent with tool-guided seeking. In *CVPR*, 2026a.

Yueqian Lin, Jingyang Zhang, Qinsi Wang, Hancheng Ye, Yuzhe Fu, Yudong Liu, Hai Helen Li, and Yiran Chen. HippoMM: Hippocampal-inspired multimodal memory for long audiovisual event understanding. In *CVPR Findings*, pp. 5968–5977, 2026b.

Lin Long, Yichen He, Wentao Ye, Yiyuan Pan, Yuan Lin, Hang Li, Junbo Zhao, and Wei Li. Seeing, listening, remembering, and reasoning: A multimodal agent with long-term memory. In *ICLR*, 2026a.

Yancheng Long, Yankai Yang, Hongyang Wei, Wei Chen, Tianke Zhang, Haonan Fan, Changyi Liu, Kaiyu Jiang, Jiankang Chen, Kaiyu Tang, Bin Wen, Fan Yang, Tingting Gao, Han Li, and Shuo Yang. SpatialReward: Bridging the perception gap in online RL for image editing via explicit spatial reasoning. In *ICML*, 2026b.

Yongdong Luo, Xiawu Zheng, Xiao Yang, Guilin Li, Haojia Lin, Jinfa Huang, Jiayi Ji, Fei Chao, Jiebo Luo, and Rongrong Ji. Video-RAG: Visually-aligned retrieval-augmented long video comprehension. In *NeurIPS*, 2025.

Ziyu Ma, Chenhui Gou, Hengcan Shi, Bin Sun, Shutao Li, Hamid Rezatofighi, and Jianfei Cai. DrVideo: Document retrieval based long video understanding. In *CVPR*, pp. 18936–18946, 2025.

OpenAI. GPT-5 system card. System card, 2025. URL https://openai.com/index/gpt-5-system-card/.

Qwen Team. Qwen3.8. Official blog, 2026.

QwenLM. Qwen-MM-Plugins: Video memory. Software and documentation, 2026. Accessed September 17, 2026.

Aniket Rege, Arka Sadhu, Yuliang Li, Kejie Li, Ramya Korlakai Vinayak, Yuning Chai, Yong Jae Lee, and Hyo Jin Kim. Agentic very long video understanding. In *ACL*, 2026.

Xiaoqian Shen, Yunyang Xiong, Changsheng Zhao, Lemeng Wu, Jun Chen, Chenchen Zhu, Zechun Liu, Fanyi Xiao, Balakrishnan Varadarajan, Florian Bordes, Zhuang Liu, Hu Xu, Hyunwoo J. Kim, Bilge Soran, Raghuraman Krishnamoorthi, Mohamed Elhoseiny, and Vikas Chandra. LongVU: Spatiotemporal adaptive compression for long video-language understanding. In *ICML*, volume 267, pp. 54582–54599. PMLR, 2025a.

Xiaoqian Shen, Wenxuan Zhang, Jun Chen, and Mohamed Elhoseiny. Vgent: Graph-based retrieval-reasoning-augmented generation for long video understanding. In *NeurIPS*, 2025b.

Dídac Surís, Sachit Menon, and Carl Vondrick. ViperGPT: Visual inference via Python execution for reasoning. In *ICCV*, pp. 11888–11898, 2023.

Shulin Tian, Ruiqi Wang, Hongming Guo, Penghao Wu, Yuhao Dong, Xiuying Wang, Jingkang Yang, Hao Zhang, Hongyuan Zhu, and Ziwei Liu. Ego-R1: Chain-of-tool-thought for ultra-long egocentric video reasoning. *arXiv preprint arXiv:2506.13654*, 2025.

Weihan Wang, Zehai He, Wenyi Hong, Yean Cheng, Xiaohan Zhang, Ji Qi, Ming Ding, Xiaotao Gu, Shiyu Huang, Bin Xu, Yuxiao Dong, and Jie Tang. LVBench: An extreme long video understanding benchmark. In *ICCV*, pp. 22958–22967, 2025a.

Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. VideoAgent: Long-form video understanding with large language model as agent. In *ECCV*, 2024.

Ziyang Wang, Shoubin Yu, Elias Stengel-Eskin, Jaehong Yoon, Feng Cheng, Gedas Bertasius, and Mohit Bansal. VideoTree: Adaptive tree-based video representation for LLM reasoning on long videos. In *CVPR*, pp. 3272–3283, 2025b.

Jingkang Yang, Shuai Liu, Hongming Guo, Yuhao Dong, Xiamengwei Zhang, Sicheng Zhang, Pengyun Wang, Zitang Zhou, Binzhu Xie, Ziyue Wang, Bei Ouyang, Zhengyu Lin, Marco Cominelli, Zhongang Cai, Bo Li, Yuanhan Zhang, Peiyuan Zhang, Fangzhou Hong, Joerg Widmer, Francesco Gringoli, Lei Yang, and Ziwei Liu. EgoLife: Towards egocentric life assistant. In *CVPR*, pp. 28885–28900, 2025.

Yankai Yang, Yancheng Long, Wei Chen, Xingyu Lu, Hongyang Wei, Bin Wen, Fan Yang, Tingting Gao, Han Li, and Shuo Yang. SpatialFlow-GRPO: Where spatial credit drives image editing. *arXiv preprint arXiv:2606.26872*, 2026a.

Yankai Yang, Yancheng Long, Hongyang Wei, Wei Chen, Kaiyu Jiang, Changyi Liu, Jiankang Chen, Tianke Zhang, Haonan Fan, Kaiyu Tang, Bin Wen, Fan Yang, Tingting Gao, Han Li, and Shuo Yang. Joint reward modeling: Internalizing chain-of-thought for efficient visual reward models. *arXiv preprint arXiv:2602.07533*, 2026b.

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. MM-REACT: Prompting ChatGPT for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*, 2023.

Woongyeong Yeo, Kangsan Kim, Jaehong Yoon, and Sung Ju Hwang. WorldMM: Dynamic multimodal memory agent for long video reasoning. In *CVPR*, pp. 25599–25609, 2026.

Ce Zhang, Taowen Lu, Md Mohaiminul Islam, Ziyang Wang, Shoubin Yu, Mohit Bansal, and Gedas Bertasius. A simple LLM framework for long-range video question-answering. In *EMNLP*, pp. 21715–21737, 2024.

APPENDIX

A TOOL INTERFACE

Table 5 lists the tools that implement the three actions of Sprout (Section 3).

Table 5: Tool interface of Sprout. Frames attached by a tool stay in the context for the current round only; each write-back tool accepts only the chunk or interval watched in the preceding round.

Action	Tool (arguments)	Role
Watch	`read_next` ()	Attach the next unwatched chunk at the low frame rate f_c.
Remember	`write_chunk_memory` (segments)	Store the pending chunk as contiguous coarse memory nodes.
Revisit	`search_memory` (pattern, window)	Retrieve facts and prior questions by literal alternatives and/or a time window.
	`view_node` (node_id, focus, fps)	Re-attach the whole interval of an existing node at the chosen frame rate f_r.
	`view_time` (start, end, focus, fps)	Attach a short interval inside the watched video at the chosen frame rate f_r.
	`refine_node` (node_id, segments)	Subdivide the leaf just viewed into contiguous children.
	`add_observation` (summary, facts)	Attach what was just viewed with `view_time` as a timestamped child of its node.

B LITERAL MATCHING VERSUS EMBEDDING RETRIEVAL

Sprout and Qwen-MM-Plugins both let the agent search a textual memory, but they match queries to memory in different ways.

How each method searches. In Sprout, `search_memory` uses literal matching. A query is a list of alternatives separated by "|", and a fact matches if it contains any alternative as a case-insensitive substring; an optional time window restricts the search to one span. The tool returns every matching fact in temporal order under its node, up to 40 facts, followed by up to five earlier questions that mention the query. Qwen-MM-Plugins uses embedding retrieval. When its memory is built, every node of the graph is encoded by a text embedding model (`qwen3-vl-embedding`). At query time, the query is encoded by the same model, nodes are ranked by cosine similarity and by BM25, the two rankings are fused, and the top 10 nodes are returned.

Why literal matching is enough. Current agent models are already strong at searching: they can choose keywords, list synonyms in one query, as in `"vest | waistcoat | tank top"`, and rewrite the query when a search returns nothing. With such an agent, simple literal matching finds the facts it needs, so Sprout does not rely on an extra text embedding model. This saves both time and tokens. In Qwen-MM-Plugins, embedding the nodes alone costs 7.5M, 7.9M, and 1.1M tokens of construction on LVBench, Video-MME Long, and Video-Holmes, and every search waits for one more call to the embedding model. Literal matching instead runs directly on the current memory tree, with no embedding cost and no index to rebuild when new nodes are written.

Comparison. To compare the two directly, we replace literal matching in `search_memory` with embedding retrieval, using the same text embedding model as Qwen-MM-Plugins, and keep everything else unchanged. Table 6 reports the results with Qwen3.8-Max. The two perform within one point of each other: literal matching is 0.8 and 0.4 points higher on LVBench and Video-MME Long, and embedding retrieval is 0.5 points higher on Video-Holmes. Literal matching therefore gives similar accuracy without the cost of an embedding model.

Table 6: **Literal matching versus embedding retrieval** in `search_memory`, with Qwen3.8-Max. Accuracy (%); Video-MME(L) is the Long subset of Video-MME. The embedding variant uses the same text embedding model as Qwen-MM-Plugins (`qwen3-vl-embedding`); everything else is unchanged.

Search	LVBench	Video-MME(L)	Video-Holmes
Embedding retrieval	86.1	85.6	72.1
Literal matching (ours)	86.9	86.0	71.6

C PROMPTS

Sprout uses one system prompt and one user turn per question; everything about individual tools (arguments, usage, return values) is sent as the tool schemas of Table 5. Prompt 1 is the system prompt, and Prompt 2 is the user turn that opens each question, with placeholders in angle brackets. The timestamp sentence follows the video length: videos of at most 60 minutes use `MM:SS` clock strings instead of `HH:MM:SS`. For GPT-5, the system prompt ends with one more sentence, "Call at most one tool per turn; wait for its result before choosing the next call.", because its endpoint ignores the request option that disables parallel tool calls.

Prompt 1: System prompt

```
You are a multimodal agent answering a multiple-choice question about one long video while
     building and maintaining a reusable memory tree of it. You act only by calling the
     provided tools, and you finish by calling answer.

You read the video yourself in chronological chunks. read_next attaches the next unread
     chunk's frames at the streaming fps (READ_FPS); on your next turn you store the
     complete chunk with write_chunk_memory, and then read on, inspect, or answer. At the
     very first question of a video the memory is empty and everything is unread, so your
     only useful first call is read_next; while unread video remains, keep reading if
     unseen content could change the answer — especially for whole-video questions.

The streaming fps is low, so brief motion, transient on-screen text, fine action order,
     and exact counts may be missing from memory even when the surrounding events are
     recorded. Recover those inside the read prefix with view_time on a tight interval at
     a higher fps, or view_node on a node; what you see is stored only if you call
     add_observation (after view_time) or refine_node (after view_node of a leaf).

You do NOT receive the memory tree — not even its outline. MEMORY_PROGRESS reports only
     how far the video is read (read_until) and what remains unread; missing memory after
     read_until means unread video, not absent content. Everything read so far is stored
     as segments and retrieved on demand with search_memory. Work from evidence, not from
     guesses: probe the memory before answering — several literal alternatives in one
     pattern (synonyms, the option texts, names, on-screen words), a time window when the
     question pins one down, and different phrasings when the first finds nothing.

Memory holds objective, reusable video facts only — named identities with distinguishing
     attributes, exact counts, temporal order and state changes, cause-to-consequence
     pairs, notable dialogue or on-screen text — never the question, the options, your
     reasoning, or your answer. Segments are chronological, contiguous, and
     non-overlapping, with meaningful event boundaries; the runtime assigns node IDs, so
     never invent them.

PRIOR_QUESTIONS lists earlier questions already answered for this same video with the
     answer you chose; search_memory also returns prior answers whose question mentions
     the pattern. Use them as cross-question context: keep answers consistent with
     established narrative understanding. They are NOT objective video facts and must
     never be written into memory.

Timestamps are global HH:MM:SS clock strings (e.g. "01:11:36"); always write all three
     fields, never "11:36" or "1:11:36". Inspecting one clip does not imply the question
     is ready to answer: re-evaluate after every clip and do not guess. Call answer only
     when the accumulated evidence distinguishes one option.
```

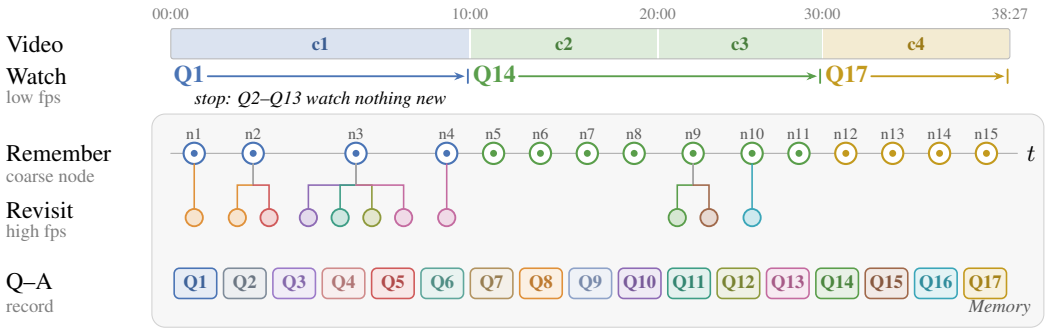


Figure 4: **The memory tree that Sprout builds on one LVBench video** (Gemini 3.8 Flash, the video of Figures 5 and 6), after all 17 of its questions, all answered correctly. Each question has its own color. Q1 watches the first 10-minute chunk and records it as n1–n4; Q2–Q13 watch nothing further, Q14 watches the next two chunks, and Q17 the rest. Each child below a coarse node is written by a revisit and carries the color of the question that wrote it; Q2, Q3, Q4, Q6, Q7, and Q9 answer without writing anything. The child of n2 written by Q5 is the one shown in Figure 6.

Prompt 2: User turn of each question

```
VIDEO_START: 00:00:00
VIDEO_END: <video duration>
TIMESTAMP_FORMAT: HH:MM:SS
READ_FPS: <streaming fps>

PRIOR_QUESTIONS (question -> your answer):
Q1: <earlier question> -> <chosen letter>. <chosen option>
...

MEMORY_PROGRESS:
coverage: read_until <watched until> of <video duration>; unread <watched until> - <video
    duration>

QUESTION:
<question>

OPTIONS:
A. <option A>
B. <option B>
C. <option C>
D. <option D>
```

D QUALITATIVE EXAMPLES

Figure 4 shows the memory tree that Sprout builds on one LVBench video over all 17 of its questions, with Gemini 3.8 Flash and the prompts of Appendix C; all 17 are answered correctly. The video is watched only as far as the questions require, and every node below the coarse nodes is written by a revisit that a question triggers. Figures 5–9 follow nine of these questions: the first five, and the last four, where the questions move past the watched prefix. The frames are taken from the video at the timestamps shown. The first question watches the first chunk and records it; every later question starts from the memory left by the questions before it. Depending on what the memory already holds, a question is answered by one search (Q2), revisits the video and uses the frames as evidence (Q3, Q4, and Q17), revisits and writes a corrected or missing fact back into the tree (Q5, Q14, Q15, and Q16), or first watches further when the question lies past the watched prefix (Q14 and Q17).

LVBench `Z86xysw5Ncc` (38:27), questions 1–3 of the video

Q1 **What category does this video belong to?** 3 rounds 11.7K tokens

A. Olympic games ✓ B. The BBC News C. The NBA D. The country life vlog

r1 **Watch** ▷ `read_next()`

Chunk 00:00–10:00 attached at 0.1 fps (60 frames).



00:40 01:40 04:20 07:30 08:50

r2 **Remember** ▷ `write_chunk_memory(4 segments)`

Stored node_1–node_4; the frames are released.

node_1	00:00–01:40	Introduction and start list for Men's 100m Round 1 Heat 1 at London 2012 Olympic Games.
node_2	01:40–05:20	Men's 100m Round 1 Heat 1 race won by Tyson Gay.
node_3	05:20–08:40	Men's 100m Round 1 Heat 2 race won by Justin Gatlin.
node_4	08:40–10:00	Women's pole vault competition at London 2012 Olympic stadium.

r3 **Answer** ▷ `answer("A")`

Q2 **How many meters is the race about men in the video?** 2 rounds 5.0K tokens

A. 1500 B. 3000 C. 100 ✓ D. 50

r1 **Revisit** ▷ `search_memory("race|meter|1500|3000|100|50|track|run")`

5 facts in 4 nodes, led by node_1: "Introduction and start list for Men's 100m Round 1 Heat 1 . . ."

r2 **Answer** ▷ `answer("C")` answered from memory alone, no frames

Q3 **How many athletes in Men's 100M Start-List-Round 1 Heat 1?** 4 rounds 12.7K tokens

A. 9 B. 7 C. 8 ✓ D. 10

r1 **Revisit** ▷ `search_memory("Heat 1|Start-List|Round 1|Heat")`

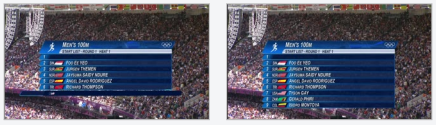
8 facts in 3 nodes, including @01:40 "Start list for Men's 100m Round 1 Heat 1 is displayed on screen." The count is not recorded.

r2 **Revisit** ▷ `view_time(01:30, 02:00, fps=1)`

The end is clamped to 01:40, the end of node_1.

r3 **Revisit** ▷ `view_time(01:40, 01:45, fps=1)`

5 frames of node_2.



01:38 01:41

r4 **Answer** ▷ `answer("C")` frames used as evidence only, nothing written

Figure 5: **The first three questions on one LVBench video** (Sprout with Gemini 3.8 Flash; Figure 6 continues with questions 4 and 5). Each round shows the tool call (▷) and, in the shaded block, its result. Q1 watches the first chunk and records it as four nodes; no later question in this figure watches further. Q2 is answered by one search. Q3 finds where the start list is shown but not its length, so it revisits that interval at 1 fps, first clamped at the boundary of node_1, and counts the athletes on the frames. Tool calls keep only their key arguments and tool results are abridged; memory text is verbatim from the request log, and token counts are total input tokens per question.

LVBench `Z86xysw5Ncc` (38:27), questions 4–5 of the video

Q4 **What's the number on the back of Tyson Gay?** 4 rounds 26.4K tokens

A. 3218 B. 3712 C. 3718 D. 3217 ✓

r1 **Revisit** ▷ `search_memory("Gay|Tyson|3218|3712|3718|3217")`

2 facts in node_2; no number.

r2 **Revisit** ▷ `search_memory(start=01:40, end=05:20)`

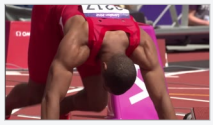
All 4 facts of node_2; no number.

r3 **Revisit** ▷ `view_node(node_2, fps=1, focus="number on the back of Tyson Gay's uniform or bib")`

The whole node 01:40–05:20 attached at 1 fps.



03:19



03:31

r4 **Answer** ▷ `answer("D")` frames used as evidence only, nothing written

Q5 **When does the first heat of the men's 100m startstart of a race?** 4 rounds 13.1K tokens

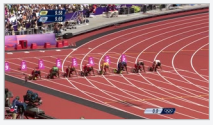
A. The video time of 02:56 B. The video time of 03:46
C. The video time of 03:56 ✓ D. The video time of 02:46

r1 **Revisit** ▷ `search_memory("heat 1 | start | race | gun | 100m", start=02:00, end=04:30)`

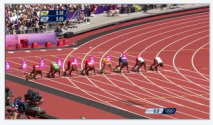
@03:50 "The starter fires the gun and Heat 1 commences.", and the prior answers of Q2 and Q3.

r2 **Revisit** ▷ `view_time(03:40, 04:05, fps=1)`

25 frames of node_2.



03:53



03:56

r3 **Revisit** ▷ `add_observation(summary, 3 facts)`

Stored as child node_2_1; the frames are released.

node_2_1 03:40–04:05 Men's 100m Round 1 Heat 1 starts at 03:56 with runners leaping from the blocks.
@03:56 The race of Men's 100m Round 1 Heat 1 officially starts as the gun fires and running clock starts (reading 0.1 at 03:56).

r4 **Answer** ▷ `answer("C")`

Figure 6: **Questions 4 and 5 on the video of Figure 5.** Q4 finds that the bib number is not in memory, by pattern and then by time window, and revisits the whole node_2 at 1 fps to read it. Q5 revisits the start of the race, finds the gun at 03:56 rather than the 03:50 recorded at 0.1 fps, and writes the corrected time back as the child node_2_1.

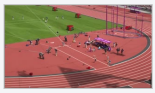
LVBench `Z86xysw5Ncc` (38:27), question 14 of the video

Q14 **When does Men's 100M Round 1 Heat 4 start preparing?** 7 rounds 46.8K tokens

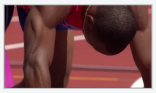
A. At the video time of 19:38 B. At the video time of 20:38
C. At the video time of 19:28 D. At the video time of 20:28 ✓

r1 **Watch** ▷ `read_next()`

Chunk 10:00–20:00 attached at 0.1 fps (60 frames).


10:20


12:30


15:00


18:00

r2 **Remember** ▷ `write_chunk_memory(4 segments)`

Stored node_5–node_8; the frames are released.

node_5 10:00–11:25 Field event highlights including pole vault and javelin throw, followed by the Heat 2 result graphic.
node_6 11:25–13:15 Athletes warming up in the indoor facility and gathering in the call room.
node_7 13:15–16:45 Men's 100M Round 1 Heat 3 intros, race, and celebration won by Ryan Bailey.
node_8 16:45–20:00 Repeated coverage of Men's 100M Round 1 Heat 3 including start list, race, and slow-motion replays.

r3 **Watch** ▷ `read_next()`

Chunk 20:00–30:00 attached at 0.1 fps (60 frames).


21:00


23:30


26:40


29:00

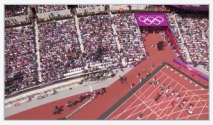
r4 **Remember** ▷ `write_chunk_memory(3 segments)`

Stored node_9–node_11; the frames are released.

node_9 20:00–24:45 Men's 100M Round 1 Heat 4 start list, athlete introductions, race won by Usain Bolt in 10.08s, and replays.
node_10 24:45–28:10 Men's 100M Round 1 Heat 5 start list, intros, race won by Asafa Powell in 10.04s, and replays.
node_11 28:10–30:00 Men's 100M Round 1 Heat 6 introductions and lineup at the starting blocks.

r5 **Revisit** ▷ `view_time(20:20, 20:45, fps=1, focus="20:28 vs 20:38")`

25 frames of node_9.


20:28


20:38

r6 **Revisit** ▷ `add_observation(summary, 3 facts)`

Stored as child node_9_1; the frames are released.

node_9_1 20:20–20:45 Camera settles on the track for Heat 4 and the Start List - Round 1 Heat 4 graphic appears.
@20:28 The camera pans down over the crowd to focus on the 100m start line where athletes are milling about and preparing.

r7 **Answer** ▷ `answer("D")`

Figure 7: **Question 14 on the video of Figure 5.** Q2–Q13 are all answered within the first chunk. Q14 asks about a moment after 10:00, so it watches and records the next two chunks, then revisits the start of Heat 4 at 1 fps to tell 20:28 from 20:38 and writes what it sees as node_9_1.

LVBench `Z86xysw5Ncc` (38:27), questions 15–16 of the video

Q15 **What is the best record for Men's 100M Round 1 Heat 4?** 4 rounds 16.5K tokens

A. 10.23 B. 9.58 C. 9.69 D. 10.09 ✓

r1 **Revisit** ▷ `search_memory("Heat 4|Round 1 Heat 4|10.|9.", start=20:00)`

5 facts in 3 nodes, including @23:30 "The Heat 4 race starts, and Usain Bolt pulls ahead to win comfortably in 10.08s." 10.08 is not an option.

r2 **Revisit** ▷ `view_time(23:35, 24:20, fps=1, focus="winning time")`

45 frames of node_9.



23:52

r3 **Revisit** ▷ `add_observation(summary, 1 fact)`

Stored as child node_9_2; the frames are released.

node_9_2 23:35–24:20 Usain Bolt wins Men's 100M Round 1 Heat 4 with an official time of 10.09s.

r4 **Answer** ▷ `answer("D")`

Q16 **Who is the first in Men's 100M Roound 1 Heat 5?** 4 rounds 17.2K tokens

A. Su BingTian B. Usain Bolt C. Asafa Powell ✓ D. Tyson Gay

r1 **Revisit** ▷ `search_memory("Heat 5|Heat 4|Bolt|Bingtian|Powell")`

18 facts in 7 nodes, including @26:40 "Heat 5 race is run and Asafa Powell finishes first in 10.04s."

r2 **Revisit** ▷ `view_time(26:40, 27:05, fps=1, focus="who finishes first")`

25 frames of node_10.



26:44



27:04

r3 **Revisit** ▷ `add_observation(summary, 2 facts)`

Stored as child node_10_1; the frames are released.

node_10_1 26:40–27:05 Finish of Men's 100M Round 1 Heat 5 won by Asafa Powell with a time of 10.04s.

r4 **Answer** ▷ `answer("C")`

Figure 8: **Questions 15 and 16 on the video of Figure 5.** Q15 finds a winning time of 10.08s recorded at 0.1 fps, which is not among the options; the revisit reads 10.09s from the result graphic and writes it back as node_9_2. Q16 finds the winner already in memory and revisits the finish to confirm it before answering.

LVBench `Z86xysw5Ncc` (38:27), question 17 of the video

Q17 **What place is Su Bingtian in for Men's 100M Round 1 Heat 6?** 8 rounds 45.2K tokens

A. Fourth B. Third ✓ C. First D. Second

r1 Revisit ▷ `search_memory("Su|Bingtian|Heat 6|Heat6|China")`

6 facts in 5 nodes; for Heat 6 only node_11, "introductions and lineup at the starting blocks", which ends at 30:00, where the read prefix ends.

r2 Watch ▷ `read_next()`

Chunk 30:00–38:27 attached at 0.1 fps (51 frames).


30:40


33:00


36:00


38:10

r3 Remember ▷ `write_chunk_memory(4 segments)`

Stored node_12–node_15; nothing is left unread.

node_12 30:00–31:50 Men's 100M Round 1 Heat 6 is run and won by Yohan Blake, with Ryota Yamagata second and Su Bingtian third.
@31:00 The stadium big screen shows results for Heat 6: 1st Yohan Blake (10.00 Q), 2nd Ryota Yamagata (10.07 Q PB), 3rd Su Bingtian (10.19 Q).
node_13 31:50–34:30 Preparation and athlete introductions for Men's 100M Round 1 Heat 7.
node_14 34:30–37:35 Men's 100M Round 1 Heat 7 is run and won by Dwain Chambers.
node_15 37:35–38:27 Semi-final qualifiers list is displayed, and the broadcast concludes with stadium aerial views.

r4 Revisit ▷ `view_time(30:55, 31:15, fps=1, focus="Su Bingtian's place")`

20 frames of node_12.

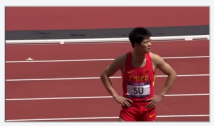
r5 Revisit ▷ `view_time(31:30, 31:45, fps=1, focus="official results graphic")`

15 frames of node_12.

r6 Revisit ▷ `view_time(31:00, 31:08, fps=2, focus="places 1, 2, 3")`

16 frames of node_12.


31:00


31:08

r7 Revisit ▷ `view_time(38:05, 38:15, fps=1, focus="qualifiers graphic")`

10 frames of node_15.

r8 Answer ▷ `answer("B")` frames used as evidence only, nothing written

Figure 9: **Question 17 on the video of Figure 5.** Q17 asks about Heat 6, whose race lies past the watched prefix, so it watches and records the last chunk; the new memory already names Su Bingtian third, and four revisits of the result graphics confirm it without writing anything.