DramaSearch: Benchmarking Multimodal Searching Agent in Ultra-Long TV Dramas

Anonymous ACL submission

Abstract

Searching agents have achieved significant progress in NLP, yet their multimodal counterparts are constrained by a lack of suitable evaluation benchmarks requiring reasoning over continuous video, audio, and images. We propose long-form TV dramas as a natural solution and introduce DramaSearch, encompassing 3,600 riddles across 9 dramas totaling 239 hours. Our dataset is generated via a graph-constrained pipeline that processes enriched captions into structured events and multi-granularity evidence views. By employing character-to-view and entity-to-view indexing, we ensure controllable clue disclosure and strictly unique answers. The riddles are categorized into multi-clue, multi-hop, and compositional types, requiring agents to aggregate isolated clues and navigate complex dependency chains. Experiments demonstrate that a Qwen3-8B agent, trained on a held-out set using SFT and RL, achieves an accuracy of 48.3%. This result establishes a strong baseline that performs favorably against zero-shot GPT-5 (37.0%), while indicating significant room for future research. All data and code will be publicly released.

1 Introduction

Searching agents that interleave reasoning with retrieval have produced steady progress in text QA (Yang et al., 2018; Trivedi et al., 2022; Wei et al., 2025), but their multimodal counterparts remain comparatively underexplored, largely because evaluation scenarios and datasets that genuinely require reasoning over video, audio, and images are scarce. Most existing long-video benchmarks (Wu et al., 2024; Fu et al., 2025; Wang et al., 2025; Chen et al., 2024a) extend QA from short clips to longer videos, yet questions still tend to be answerable from a local segment or a few visually salient cues, leaving little room for a genuine search policy.

Long-form TV dramas are a natural testbed for multimodal searching agents. A full drama spans tens of hours and contains many named characters, places, objects, and events that are tied together by rich narrative relations, so an agent has to search across distant scenes, resolve recurring characters, and integrate scattered evidence into a single answer.

Building such a benchmark, however, is non-trivial. Existing long-video QA benchmarks fall short on search-intensive reasoning along two dimensions. First, their reasoning complexity is limited, and many questions are still answerable from a single clip or short window, so high accuracy can be achieved without a real search policy. Second, the question construction process is weakly controlled, since naive caption-to-LLM pipelines easily expose overly specific cues that enable local shortcuts, and answer uniqueness is rarely verified. The two limitations compound: easy questions do not require search, and poorly controlled hard ones reward shortcut reasoning over faithful evidence composition.

We introduce **DramaSearch**, a benchmark for complex riddle reasoning in ultra-long videos. It is built from 9 long-form TV dramas (239 hours) and contains 3,600 curated riddles. Unlike conventional video QA, the answer is never directly revealed and must be inferred by aggregating individually insufficient evidence, following dependent reasoning chains, or composing both. All riddles are open-ended fill-in-the-blank questions with a single-entity ground truth, which avoids the option-pruning shortcuts of multiple choice and lets an LLM judge score predictions by semantic equivalence.

The core of DramaSearch is a graph-constrained construction pipeline. Auxiliary-enriched clip captions, produced from subtitles, OCR texts, face IDs, and speaker IDs, are first parsed into structured events and expanded into multi-granularity evidence views. Views are then indexed by Character–View and Entity–View graphs, and a candidate-set

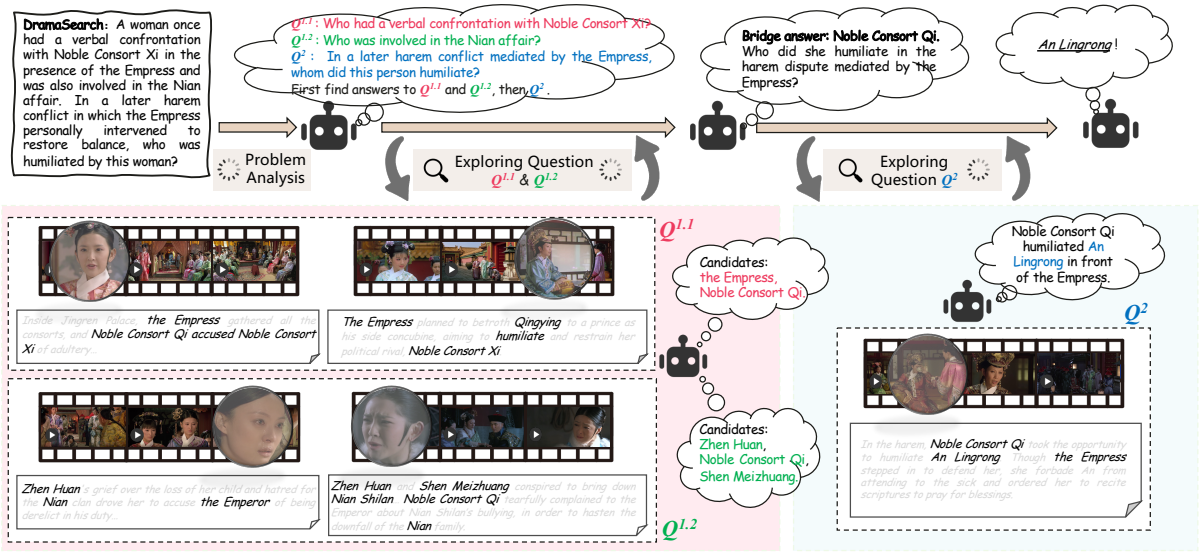


Figure 1: Reasoning through a complex riddle. Such a riddle cannot be answered directly. It requires multi-step reasoning and combining several clue sources to pin down a common answer. Here the agent decomposes the problem into two steps. It first finds a bridge answer via $Q^{1.1}$ and $Q^{1.2}$, and then solves Q^2 on top of it. Exploring $Q^{1.1}$ and $Q^{1.2}$ independently yields no unique answer, and only their joint clues determine the bridge.

diagnosis verifies whether the disclosed evidence uniquely supports the target answer. This makes clue disclosure controllable and reduces leakage. On top of these views, DramaSearch spans three riddle families. **Multi-clue Riddles** integrate individually insufficient clues from distinct events. **Multi-hop Riddles**, with two or three hops, chain dependent evidence. **Compositional Riddles** take the answer of an initial multi-clue riddle as the entry condition for a subsequent multi-hop chain. Quality is then enforced by rule-based checks, LLM validation, closed-book posterior probing, and human validation.

We also provide a baseline multimodal searching agent that iterates a search-and-integrate loop over a database aggregating clip captions, subtitles, face IDs, speaker IDs, and OCR signals. We train Qwen3-8B (Team, 2025b) on a held-out drama, *A Lifelong Journey*, that is not part of the 9-drama evaluation set. After supervised fine-tuning on filtered trajectories and reinforcement learning on a difficulty-balanced subset, the agent reaches 48.3% accuracy and is favorable to zero-shot GPT-5 at 37.0%, indicating that the benchmark is non-trivial yet tractable while leaving substantial headroom.

Our contributions are three-fold:

- We formulate complex riddle reasoning over ultra-long TV dramas as a benchmark task emphasizing evidence search, problem decomposition, and long-range information integration.
- We design a graph-constrained construction pipeline based on structured events, multi-granularity views, and candidate-set diagnosis, enabling controllable clue disclosure, uniqueness verification, and shortcut reduction.
- We establish a baseline searching agent and show that, despite SFT and RL gains, substantial headroom remains for future multimodal agents.

2 Related Work

Video reasoning benchmarks. Early benchmarks such as ActivityNet (Heilbron et al., 2015) and NExT-QA (Xiao et al., 2021) target action recognition and short-video QA. A later wave including MMBench (Liu et al., 2023) and MVBench (Li et al., 2023) evaluates reasoning over short clips, while LongVideoBench (Wu et al., 2024), Video-MME (Fu et al., 2025), HourVideo (Chandrasegaran et al., 2024), Neptune (Nagrani et al., 2024), and CinePile (Rawal et al., 2024) extend evaluation to longer sequences. Narrative-driven benchmarks such as DramaQA (Choi et al., 2021), SeriesBench (Zhang et al., 2025), and Movie101v2 (Yue et al., 2024) emphasize storyline understanding, and clue-grounded ones (Zhao et al., 2025; Hu et al., 2025; Chen et al., 2024a) probe multi-step reasoning. Even so, none of them require an agent to actively search for *latent* clues scattered across hundreds of scenes. DramaSearch fills this gap by combining ultra-long video du-

ration, tool-augmented multi-step reasoning, and graph-constrained answer uniqueness, as summarized in Table 1.

Multi-hop reasoning and leakage. Text multi-hop QA benchmarks such as HotpotQA (Yang et al., 2018), MuSiQue (Trivedi et al., 2022), and 2WikiMultiHopQA (Ho et al., 2020) establish how to compose questions that chain multiple pieces of evidence. They also surface a recurring pitfall, in that many nominally multi-hop questions remain answerable from a single supporting fact (Min et al., 2019), which reflects the broader shortcut-learning phenomenon (Geirhos et al., 2020). Graph-based composition was introduced to suppress such leakage at the text level. DramaSearch extends this idea to multimodal long-form video and strengthens it with an explicit graph-intersection uniqueness test, so that no strict subset of the disclosed clues suffices to solve the riddle.

Long-video understanding and multimodal agents. Full-length TV series routinely exceed the context length of current MLLMs, motivating three complementary directions. Long-context VideoLLMs (Zhang et al., 2024; Shen et al., 2024; Cheng et al., 2024; Liu et al., 2024; Li et al., 2024a) enlarge the temporal window or add memory, token-compression and visual-encoder methods (Shu et al., 2024; Li et al., 2024b; Yang et al., 2025; Zhao et al., 2024) shorten or strengthen the visual sequence, and retrieval- or agent-based systems (Ren et al., 2025; Yuan et al., 2025; Wang et al., 2024; Fan et al., 2024; Yang et al., 2024) pull multimodal evidence such as captions, OCR, speakers, and faces from external indices. A parallel line couples LLMs with tool use over open corpora (Jin et al., 2025; Wu et al., 2025; Jiang et al., 2024), and BrowseComp (Wei et al., 2025) shows that search-based reasoning over pre-indexed content is viable yet challenging for text-only agents. DramaSearch extends this paradigm to multimodal video. Most existing pipelines still decouple retrieval from reasoning, and our baseline trains the two jointly through SFT and RL, making DramaSearch a natural testbed for future end-to-end multimodal searching agents.

3 The DramaSearch Benchmark

DramaSearch aims to establish a comprehensive benchmark for evaluating complex reasoning in ultra-long videos. The two most critical challenges during construction are ensuring the uniqueness of the ground-truth answer and preventing unintentional information leakage. We address them with a four-stage pipeline. Each drama is first turned into auxiliary-enriched captions (§3.2). The captions are then parsed into structured events and indexed by evidence graphs that enable uniqueness diagnosis (§3.3). On top of these graphs, we compose three families of riddles under explicit uniqueness constraints (§3.4), and finally screen all candidates with hierarchical quality control (§3.5).

	Duration	Reasoning	Tools	# QA	Ans.
Video-MME	min.	1-step	✗	2,700	MC
LVBench	∼1h	1-step	✗	1,353	MC
CGBench	∼1h	w/ clues	✗	12,129	MC
BrowseComp	N/A	multi-step	✓	260	ent.
DramaSearch	9–54h	multi-step	✓	3,600	ent.

Table 1: Comparison of DramaSearch with representative benchmarks.

3.1 Key Concepts and Riddle Types

We define the key terms used throughout the paper. A **semantic cue** is an atomic element such as a character, location, object, action, relation, or contextual anchor that is extracted from narrative evidence. A **clue** is an evidence-bearing description derived from an event view that carries one or more semantic cues without directly revealing the answer. An **evidence view** is a structured event description at a given granularity and serves as the basic unit for clue selection. A **hop** is a dependent reasoning step whose outcome relies on the previous one. Built on these notions, our benchmark contains three types of riddles. **Multi-clue Riddles** require integrating individually insufficient clues from distinct events. **Multi-hop Riddles** require sequential evidence chaining. **Compositional Riddles** use the answer to an initial multi-clue riddle as the entry condition for a subsequent multi-hop chain. These concepts define the target structure of our riddles. We next describe how raw drama videos are first converted into auxiliary-enriched captions, which serve as the narrative source from which structured events, evidence views, and clues are later derived.

3.2 Auxiliary-Enriched Video Captioning

The benchmark is built from 9 popular long-form TV dramas, each spanning tens of hours. A 10th drama, *A Lifelong Journey* (41h46m), is processed by the same pipeline but reserved for constructing

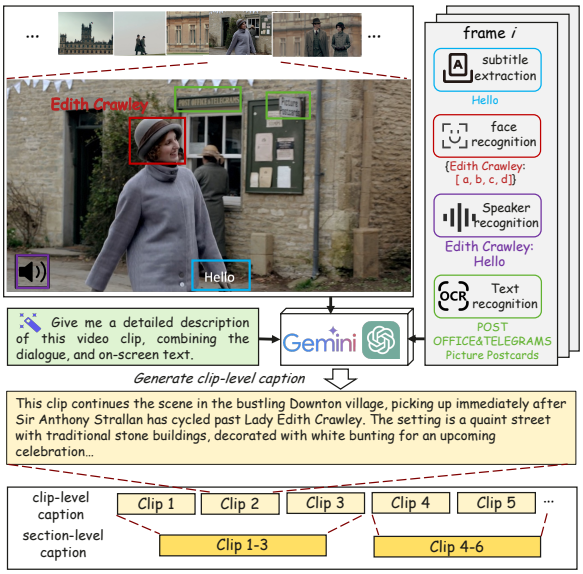


Figure 2: Video clip captioning with auxiliary data: subtitles, texts, face IDs, and speaker IDs.

the SFT and RL training sets of our baseline agent.

Directly captioning such long videos with current MLLMs often yields incomplete or unstable descriptions, so we instead construct auxiliary-enriched clip captions. As shown in Figure 2, we collect four types of auxiliary information, namely subtitles, on-screen texts, face IDs, and speaker IDs, which together provide linguistic cues, ground visible characters, and associate dialogue with speakers. Each drama is segmented into clips of at most 15 seconds, key frames are sampled at about one frame per second, and an MLLM is prompted to generate detailed clip-level descriptions. Subtitles and speaker IDs are inserted into the text prompt, while recognized face IDs are rendered on sampled frames. These enriched captions form the raw narrative source for riddle construction.

3.3 Graph-based Knowledge Formalization

TV drama scenarios often contain shortcut cues that make questions trivial for models. For instance, "Who likes tinkering with machines and inventing things?" is ambiguous in the real world but nearly deterministic in *Iron Man*. We mitigate this by decomposing captions into structured events, expanding each event into multi-granularity evidence views, and indexing all views into evidence graphs that enable explicit uniqueness diagnosis.

As shown in Figure 3, given the caption corpus $\mathcal{C} = \{C_1, \ldots, C_N\}$, each caption C_c is parsed into a set of structured events $\mathcal{E}_c = \{e_{c,1}, \ldots, e_{c,n_c}\}$, where each event is a semantic frame over characters, action, interaction target, location, objects, context, and evidence span. Since raw character mentions may appear as full names, nicknames, titles, or relations, we perform human-assisted character normalization, manually mapping easily identifiable aliases to canonical identities so that every event of the same character is indexed under a unified identity.

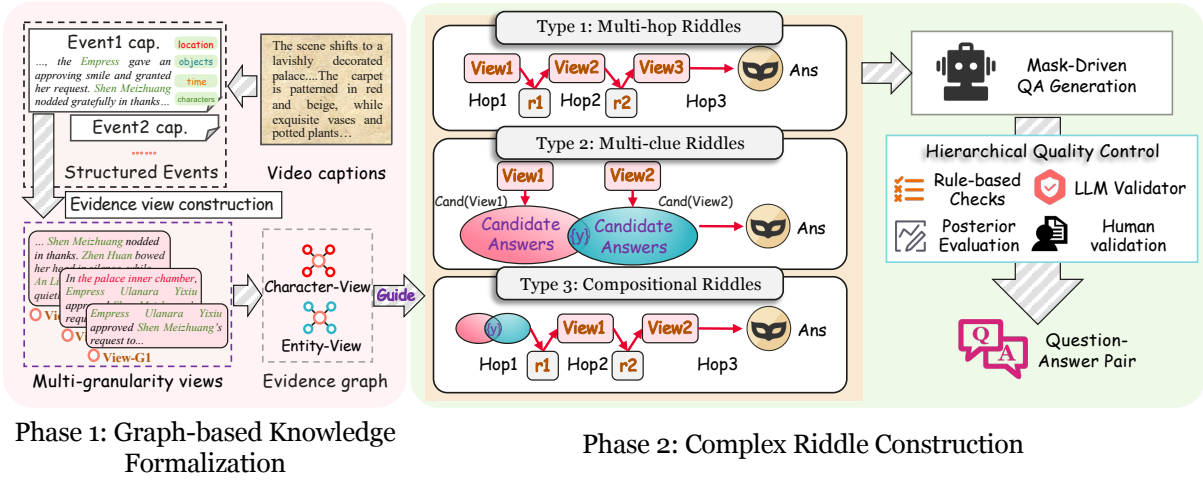
For each event e, we build evidence views $\mathcal{V}(e) = \{v_e^{\text{event}}, v_e^{G1}, v_e^{G2}, v_e^{G3}\}$, where v_e^{event} is the full view and $v_e^{G1\text{–}G3}$ are three rewritings ranging from minimal self-contained descriptions to more discriminative and complete descriptions. This separates evidence modeling from surface question generation, so that clues can be informative yet avoid answer leakage.

All views are indexed into two bipartite evidence graphs, a Character–View graph and an Entity–View graph. In practice, these graphs are implemented with the indices `character_to_views` and `entity_to_views`, which return the views containing a given character or entity. Given a target y and a view v, we form a clue signature $S(v, y)$ after removing y and its aliases, retrieve candidate views by intersecting all postings, $\mathcal{R}(v, y) = \bigcap_{s \in S(v,y)} \text{Postings}(s)$, and take the union over their answerable entities, $\text{Cand}(v, y) = \bigcup_{u \in \mathcal{R}(v,y)} \text{Answerable}(u)$. A view is uniquely identifying iff $\text{Cand}(v, y) = \{y\}$. To control disclosed information, we scan the granularities of the same event and select the least specific view that still uniquely identifies the target, while events with no uniquely identifying view are removed from the sampling pool. Equivalently, this enforces an **incompleteness** property at the riddle level, since removing any single disclosed clue leaves multiple consistent answers and the model cannot shortcut its way to the target without aggregating all of them.

3.4 Complex Riddle Construction

All riddles are drawn from verified evidence views rather than raw captions, and each construction routine enforces answer uniqueness through the intersection procedure above.

Multi-clue Riddles test whether a model can identify a character by integrating evidence that is individually insufficient but jointly discriminative. Given a target y and a clue count k, we sample k views $\{v_1, \ldots, v_k\}$ from k distinct events in different captions, requiring each to be individually



Phase 1: Graph-based Knowledge Formalization

Phase 2: Complex Riddle Construction

Figure 3: Overview of the DramaSearch data construction pipeline. Auxiliary-enriched video captions are parsed into structured events, expanded into multi-granularity evidence views, and indexed by Character–View and Entity–View graphs. Verified views are then composed into Multi-hop, Multi-clue, and Compositional Riddles, followed by mask-driven question generation and hierarchical quality control. The pipeline enables controllable evidence disclosure, answer uniqueness verification, and traceable riddle construction.

insufficient ($|\mathrm{Cand}_i| > 1$) while their intersection is the target: $\bigcap_{i=1}^{k} \mathrm{Cand}_i = \{y\}$. Thus the answer only becomes unique after aggregating evidence from all k events.

Multi-hop Riddles evaluate whether a model can follow a chain of dependent reasoning steps across dispersed video segments. We adopt reverse-chain construction: starting from the final answer $a^\star$, we recursively search backward for bridge entities or intermediate characters that connect the current hop to a previous supporting view, producing a chain of length $K \in \{2, 3\}$ reversed into natural order $h_1 \rightarrow \cdots \rightarrow h_K \rightarrow a^\star$. To prevent shortcuts, different hops must come from different captions, bridge entities must be non-trivial, and each hop must pass event-level uniqueness verification.

Compositional Riddles use the answer r of an initial multi-clue riddle as the entry condition for a subsequent multi-hop chain ending at $a^\star$, i.e., Multi-clue evidence $\rightarrow r \rightarrow$ Multi-hop chain $\rightarrow a^\star$. This is not a simple concatenation of two questions: the inferred r must be necessary for entering the multi-hop chain.

Before natural-language generation, target characters, bridge entities, and their aliases are masked from the selected views. An LLM is then prompted to rewrite the masked evidence into a single natural riddle while preserving the original facts and avoiding unsupported causal or temporal relations. Across the three families, hop depth K and clue count k serve as the main controllable factors for difficulty, whose effect is quantified in Section 5.

3.5 Hierarchical Quality Control

Each accepted riddle should be structurally verifiable, semantically unambiguous, and resistant to shortcut reasoning. We therefore apply four quality-control stages. **Rule-based checks** enforce evidence alignment, entity normalization, answer uniqueness, evidence dispersion, reasoning dependencies, and mask coverage. An **LLM validator** based on Qwen3-32B then checks faithfulness to the selected evidence views, single-answerness, absence of identity leakage, and consistency with the intended riddle type, and invalid cases are rewritten under constraints or discarded. A **posterior evaluation** stage uses GLM-4.6V as a closed-book shortcut probe and removes riddles that are solvable without access to videos, captions, or structured evidence, such as cases in *The Knockout* where cues like Gao Qiqiang and "son" directly suggest Gao Xiaochen from series-level priors. Finally, **human validation** by three annotators verifies answer correctness, evidence support, and question naturalness, and failed cases are revised or replaced. Additional details on implementation and data production are provided in the supplementary material.

4 A Multimodal Searching Agent

To measure the difficulty of DramaSearch and to facilitate future comparisons, we provide a multimodal searching agent (MSA). The agent couples an LLM controller with a multimodal retrieval backend, where the LLM plans, reasons, and is-

sues structured queries while the backend stores and retrieves evidence extracted from the ultra-long video. MSA serves as a reference point that shows the benchmark is non-trivial yet tractable and as a concrete starting point for future work.

4.1 Tool Integrated Multi-turn Reasoning

The Multi-model Toolkit. The four tools in $\mathcal{T}$ correspond to complementary retrieval channels over the video repository. Each is exposed as a function that takes a free-form text query and returns a ranked list of supporting snippets:

- CAPTION – queries semantic clip captions produced by the video-captioning model.
- SPEAKER – retrieves subtitles of a given speaker via voiceprint recognition.
- FACEID – locates appearances of specific roles via face recognition.
- OCR – searches textual elements from subtitles or on-screen texts.

The policy dynamically selects a tool based on the current reasoning context, which enables adaptive multi-hop grounding in ultra-long video.

Multi-turn Reasoning Loop. MSA treats riddle solving as a closed-loop interaction between an LLM policy π_θ and a multimodal retrieval backend. At each turn, π_θ emits a `<think>` segment for internal reasoning, followed by either a `<search>` segment that issues a structured tool call `{"query":"<text>", "type":"<tool_type>"}` whose returned evidence is appended to the context as a `<result>` segment, or an `<answer>` segment that terminates the episode. The action space $\mathcal{T} = \{\text{CAPTION}, \text{SPEAKER}, \text{FACEID}, \text{OCR}\}$ lets the agent reach beyond its parametric knowledge and query the multimodal content of the video repository $\mathcal{V}$ directly.

4.2 Training the Agent

We train Qwen3-8B in two stages, namely supervised fine-tuning (SFT) on high-quality reasoning trajectories and reinforcement learning (RL) with the GRPO (Shao et al., 2024) algorithm.

Stage 1: supervised fine-tuning. We first build a corpus of reasoning trajectories under the DramaSearch protocol. To avoid a single-teacher bias, we prompt several frontier models (GPT-4.1, GPT-5, and GPT-5.4) to roll out candidate trajectories on the same training riddles, where each trajectory follows the `<think>`–`<search>`–`<result>`–…–`<answer>` format with JSON-formatted tool calls. Candidates are then scored along seven dimensions, including answer correctness, structural integrity of the tag sequence, `<search>`–`<result>` pairing, search diversity, retrieval-score quality, bridge-entity coverage, and LLM-judged logical coherence, and the seven dimensions are combined into an overall score. Keeping only the top-scoring trajectory per riddle yields a 14K-trajectory SFT set. Fine-tuning on this set teaches the student both the structural tool-call format and a diverse repertoire of search strategies aggregated from the teachers.

Stage 2: reinforcement learning. For RL, we reuse the same multi-teacher rollouts to estimate per-riddle difficulty. The number of teachers that answer a riddle correctly partitions the pool into five bins, namely trivial (solved by all models), easy, medium, medium-hard, and hard (failed by all). We filter out instances at both extremes and uniformly sample 5K riddles from the middle three bins. The policy is initialized from the SFT checkpoint and optimized with GRPO for 800 steps.

Reward design. Each rollout receives a scalar reward $r_t = r_{\text{answer}} + \lambda_1 r_{\text{ground}} + \lambda_2 r_{\text{format}}$ that combines three terms. The **answer reward** r_{answer} is a sparse terminal reward of 1 when the predicted answer matches the ground truth and 0 otherwise. The **grounding reward** $r_{\text{ground}} = N_{\text{hit}}/N_{\text{true}}$ is the fraction of ground-truth clues that appear in the retrieved evidence, which supplies a dense signal when the final answer is wrong but the agent has begun to localize relevant clips. The **format reward** r_{format} checks the validity of the tag sequence via regex matching. We set $\lambda_1 = 0.3$ and $\lambda_2 = 0.5$.

5 Experiments

5.1 Drama-level Results

QA format and answer judging. We adopt an open-ended fill-in-the-blank format rather than multiple choice to forestall shortcut solutions, since with explicit options an agent could eliminate candidates by searching each one or simply guess when search is uninformative. Every ground-truth answer in DramaSearch is a single entity such as a character, location, object, or event label, which in turn makes free-form judging tractable. We score a prediction as correct if an LLM judge decides it is semantically equivalent to the ground-truth entity, for example by accepting aliases and translations, and incorrect otherwise. All accuracies reported below use this judgment.

Model	Avg	A	B	C	D	E	F	G	H	I	2-clue	2-hop	3-hop	Compositional
Inference without calling any tools														
GPT-5 (OpenAI, 2025)	12.9	4.7	7.7	11.7	6.9	15.7	20.8	16.6	12.1	20.0	24.9	8.0	9.5	9.2
Gemini-2.5-Pro (Team, 2024)	30.0	31.8	24.4	37.4	28.4	35.5	29.6	31.8	23.1	27.7	51.2	21.0	23.5	24.3
Qwen3-8B (Team, 2025b)	1.6	1.2	3.3	2.9	1.4	1.2	2.2	0.0	0.0	2.2	2.3	0.8	1.3	2.0
Inference with naive RAG														
GPT-5	34.5	38.3	33.9	42.6	35.4	36.1	29.4	33.7	28.9	31.9	53.9	33.0	27.3	23.8
Qwen3-8B	34.2	41.4	32.7	36.9	32.7	34.7	32.7	35.8	29.5	31.5	53.6	35.9	24.7	22.6
Inference with multi-turn tool calling														
GPT-5	37.0	40.2	36.5	45.8	39.0	38.7	32.1	35.8	30.8	34.4	56.1	35.5	29.9	26.5
Qwen3-8B	0.8	1.2	1.4	0.3	0.5	0.5	0.8	1.3	0.8	0.3	1.1	0.8	0.9	0.4
Search-R1 (Jin et al., 2025)	17.6	19.4	21.8	14.5	17.4	19.6	19.8	16.1	16.4	13.5	18.8	22.8	19.5	9.3
Ours, SFT	44.0	48.0	41.6	45.5	40.1	46.0	48.0	50.5	41.9	34.8	53.5	46.0	45.3	31.2
Ours, SFT+RL	**48.3**	**53.2**	**48.0**	**47.5**	**48.5**	**47.7**	**51.0**	**56.0**	**45.4**	**37.5**	**61.5**	**48.5**	**46.9**	**36.3**

Table 2: QA accuracy (%) on DramaSearch. Left block: per-drama results (all zero-shot; *A Lifelong Journey* held out as the training drama). Right block: results by riddle category. Drama codes: **A**: *The Long Night*; **B**: *The Qin Empire*; **C**: *Ode to Joy*; **D**: *The Knockout*; **E**: *Three-Body*; **F**: *Minning Town*; **G**: *The Long Way Home*; **H**: *Battle of Changsha*; **I**: *Empresses in the Palace*. Best in bold.

Table 2 summarizes riddle-solving QA accuracy on the DramaSearch benchmark, broken down by drama in the left block and by riddle category in the right block. All results are zero-shot cross-drama transfer, where the agent is trained on a single held-out drama, *A Lifelong Journey*, and evaluated on the 9 benchmark dramas. To establish a rigorous baseline, we evaluate both proprietary models such as GPT-5 and Gemini-2.5-Pro and open-source counterparts such as Qwen3-8B and Search-R1 under three inference configurations, including direct inference without any external tool, naive RAG, and multi-turn tool calling.

We observe several consistent trends across the baselines. First, world knowledge alone already drives a non-trivial portion of the zero-shot direct inference performance, since GPT-5 and Gemini-2.5-Pro reach 12.9% and 30.0% without any tool, with Gemini hitting 51.2% on 2-clue, while Qwen3-8B's compact parametric memory yields only 1.6%. Second, agentic capability is the bottleneck for small models. Naive RAG improves Qwen3-8B to 34.2%, but its performance plummets to 0.8% in the multi-turn setting, because early retrieval failures trap the agent in repetitive tool-call loops without the capacity for self-correction or early termination. Our SFT+RL framework explicitly addresses this bottleneck. Third, Compositional riddles are the hardest across every model and setting. On GPT-5 multi-turn, accuracy drops from 56.1% on 2-clue riddles to 26.5% on Compositional, a 29.6-point composition penalty that is roughly an order of magnitude larger than the few-point gaps reported when stacking ordinary multiple-choice video QA into two-hop chains. This confirms that our compositional family stresses end-to-end search-and-reasoning rather than merely longer surface forms.

Our agent. With access to multiple tools and multi-turn inference, our trained agent reaches 48.3% micro-average after SFT+RL, an absolute gain of 47.5 points over Qwen3-8B and 30.7 points over Search-R1 on the same multi-turn setup. Through supervised fine-tuning, the agent first acquires the tool-use behavior at 44.0%, which paves the way for RL to explore more effective search strategies and yield consistent gains across all 9 benchmark dramas. The performance on the 9 unseen dramas confirms the transferability of the learned reasoning and tool-calling behavior, since training on the held-out *A Lifelong Journey* generalizes to dramas with very different genres and production styles, from historical epics to contemporary urban stories. The per-category breakdown in the right block of Table 2 further shows that RL provides its largest gains on the harder multi-clue and Compositional riddles, with +8.0 on 2-clue and +5.1 on Compositional, indicating that RL is especially valuable in scenarios where simple teacher imitation is insufficient.

5.2 Ablation Studies

Table 2 reports the performance of the agents on different riddle types. We find that SFT brings con-

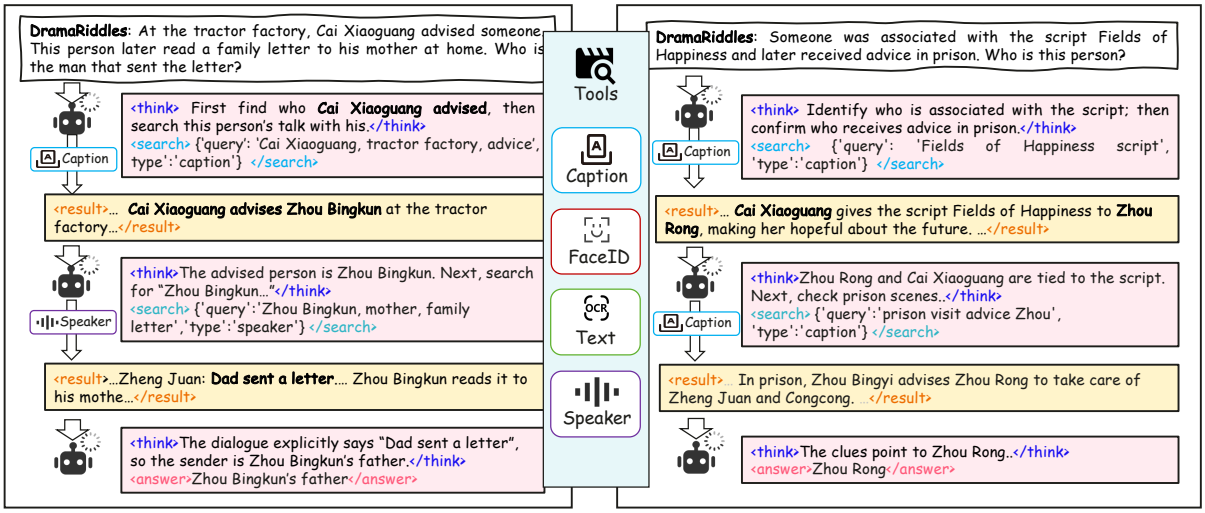


Figure 4: Examples of our agent solving two complex riddles in the drama *A Lifelong Journey*.

sistent QA accuracy gains on the 2-clue subset, but the performance on other subsets is less stable, and the accuracy on the most complex Compositional riddles drops in most cases. We attribute this to the SFT data distribution. The teacher model produces higher-quality trajectories on simpler riddles, while its trajectories on multi-hop and compositional riddles tend to under-explore the search space, which causes the student to learn overly narrow retrieval patterns. After RL, the accuracy gain becomes consistent across all subsets as the agent learns to explore beyond the teacher's trajectories. These results indicate that simply mimicking the teacher's behavior can fall short on complex riddles, and that RL is essential in such scenarios.

Impact of caption quality. We further study three versions of video captions. The default version (1) extracts clip-level captions with auxiliary information. The other two are (2) section-level captions with auxiliary information and (3) section-level captions without auxiliary information. As shown in Table 3, captions with richer information consistently lead to better results.

We also evaluate a configuration that only uses caption data at inference time, where the SPEAKER, FACEID, and OCR tools are excluded from the action space. When weaker captions are used, calling auxiliary tools brings consistent accuracy gains. At the same time, summarizing such auxiliary information directly into captions is more efficient, since the agent can then access integrated information with fewer search rounds. From a different angle, it also shows that our agent is able to call multimodal tools to compensate for missing information.

Multimodal Toolkits	**Type (1)**	**Type (2)**	**Type (3)**
w/o auxiliary tools	29.0	16.8	12.1
w/ all tools	37.5	19.0	14.7

Table 3: QA accuracy (%) with different caption quality and different tool sets. All tests are performed on the drama *Empresses in the Palace*.

Case study. We show two examples of riddle solving in Figure 4. In the first example, the agent calls the captioning tool and finds a probable bridge, the character *Zhou Bingkun*, in its first search, and then confirms the answer using the speaker tool, which illustrates its ability to switch across multimodal tools. In the second example, the agent obtains multiple candidates from the first search and then pinpoints the final answer with one additional search.

6 Conclusion

In this paper, we present a novel benchmark on long-form TV dramas. The task is to solve complex riddles that mainly involve the relationship between named entities, such as roles and events. We ensure answer uniqueness and prevent information leakage during construction by adopting a graph-constrained pipeline with controllable clue disclosure and four-stage quality control. As a baseline, we train a multimodal searching agent via SFT and RL, reaching favorable performance against proprietary models and other open-source agents. We hope DramaSearch encourages further work on long-video understanding and multimodal reasoning models.

Limitations

The modest absolute accuracy of current agents suggests that complex riddle reasoning over ultra-long videos remains far from solved, and evidence search is still a key bottleneck. Even our best SFT+RL agent leaves more than half of the riddles unanswered, and a non-trivial share of failures traces back to queries that never return the relevant clip rather than to faulty reasoning over retrieved evidence. The gap is most pronounced on Compositional riddles, where the agent must chain a multi-clue inference into a multi-hop search, and current agents tend to commit to an early bridge entity and rarely revise it when later hops contradict the choice. Closing this gap likely requires query reformulation, retrieval-feedback loops, and tighter coupling between reasoning and search than the simple search-and-integrate template explored here.

Our construction pipeline also depends on auxiliary signals such as subtitles, on-screen text, face IDs, and speaker IDs, whose residual errors may introduce noise into the generated captions and riddles. Subtitle OCR and face recognition are reliable in aggregate, but speaker recognition operates at the noisier subtitle-line level, and the underlying caption MLLM occasionally misses small visual details or attributes actions to the wrong character. Errors at any of these stages can propagate through structured events, evidence views, and finally riddle generation. The hierarchical quality-control procedure of rule-based checks, an LLM validator, a closed-book posterior probe, and human validation reduces but cannot eliminate such residual noise, particularly for minor characters with sparse face libraries or short on-screen presence.

References

Brandon Castellano. PySceneDetect.

Keshigeyan Chandrasegaran, Agrim Gupta, Lea M. Hadzic, and 1 others. 2024. Hourvideo: 1-hour video-language understanding. In *NeurIPS*.

Guo Chen, Yicheng Liu, Yifei Huang, and 1 others. 2024a. Cg-bench: Clue-grounded question answering benchmark for long video understanding. *arXiv preprint arXiv:2412.12075*.

Yafeng Chen, Siqi Zheng, Hui Wang, and 1 others. 2024b. Eres2netv2: Boosting short-duration speaker verification performance with computational efficiency. *Interspeech 2024*.

Zesen Cheng, Sicong Leng, Hang Zhang, and 1 others. 2024. VideoLLaMA 2: Advancing spatial-temporal modeling and audio understanding in video-llms. *arXiv preprint arXiv:2406.07476*.

Seongho Choi, Kyoung-Woon On, Yu-Jung Heo, and 1 others. 2021. DramaQA: Character-centered video story understanding with hierarchical QA. In *AAAI*.

Cheng Cui, Ting Sun, Manhui Lin, and 1 others. 2025. Paddleocr 3.0 technical report.

Yue Fan, Xiaojian Ma, Rujie Wu, Yuntao Du, Jiaqi Li, Zhi Gao, and Qing Li. 2024. VideoAgent: A memory-augmented multimodal agent for video understanding. In *Proc. of ECCV*.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, and 1 others. 2025. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *Computer Vision and Pattern Recognition*.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. 2020. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11).

Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. 2015. Activitynet: A large-scale video benchmark for human activity understanding. *CVPR*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proc. of COLING*.

Kairui Hu, Penghao Wu, Fanyi Pu, and 1 others. 2025. Video-mmmu: Evaluating knowledge acquisition from multi-discipline professional videos. *ArXiv*, abs/2501.13826.

Dongzhi Jiang, Renrui Zhang, Ziyu Guo, and 1 others. 2024. Mmsearch: Benchmarking the potential of large models as multi-modal search engines. *arXiv preprint arXiv:2409.12959*.

Bowen Jin, Hansi Zeng, Zhenrui Yue, and 1 others. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. In *arXiv.org*, volume abs/2503.09516.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. 2024a. LLaVA-OneVision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*.

Kunchang Li, Yali Wang, Yinan He, and 1 others. 2023. Mvbench: A comprehensive multi-modal video understanding benchmark. *CVPR*.

Xinhao Li, Yi Wang, Jiashuo Yu, and 1 others. 2024b. Videochat-flash: Hierarchical compression for long-context video modeling. *arXiv preprint arXiv:2501.00574*.

Jiajun Liu, Yibing Wang, Hanyu Ma, Xiaoping Wu, Xiaoqi Ma, Xiaoming Wei, and 1 others. 2024. Kangaroo: A powerful video-language model supporting long-context video input. *arXiv preprint arXiv:2408.15542*.

Yuanzhan Liu, Haodong Duan, Yuanhan Zhang, and 1 others. 2023. Mmbench: Is your multi-modal model an all-around player? In *ECCV*.

Sewon Min, Eric Wallace, Sameer Singh, and 1 others. 2019. Compositional questions do not necessitate multi-hop reasoning. In *Proc. of ACL*.

Arsha Nagrani, Mingda Zhang, Ramin Mehran, and 1 others. 2024. Neptune: The long orbit to benchmarking long video understanding. *arXiv preprint arXiv:2412.09611*.

OpenAI. 2025. Gpt-5. https://openai.com.

Ruchit Rawal, Khalid Saifullah, Ronen Basri, David Jacobs, Gowthami Somepalli, and Tom Goldstein. 2024. CinePile: A long video question answering dataset and benchmark. *arXiv preprint arXiv:2405.08813*.

Xingyu Ren, Alexandros Lattas, Baris Gecer, and 1 others. 2023. Facial geometric detail recovery via implicit representation. In *2023 IEEE 17th International Conference on Automatic Face and Gesture Recognition (FG)*.

Xubin Ren, Lingrui Xu, Long Xia, and 1 others. 2025. Videorag: Retrieval-augmented generation with extreme long-context videos. In *arXiv.org*, volume abs/2502.01549.

Zhihong Shao, Peiyi Wang, Qihao Zhu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv*, abs/2402.03300.

Xiaoqian Shen, Yunyang Xiong, Changsheng Zhao, and 1 others. 2024. Longvu: Spatiotemporal adaptive compression for long video-language understanding. *arXiv preprint arXiv:2410.17434*.

Yan Shu, Peitian Zhang, Zheng Liu, and 1 others. 2024. Video-xl: Extra-long vision language model for hour-scale video understanding. In *Computer Vision and Pattern Recognition*.

ByteDance Seed Team. 2025a. Seed1.5-vl technical report. *arXiv preprint arXiv:2505.07062*.

Gemini Team. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *ArXiv*, abs/2403.05530.

Qwen Team. 2025b. Qwen3 technical report.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10.

Weihan Wang, Zehai He, Wenyi Hong, and 1 others. 2025. Lvbench: An extreme long video understanding benchmark. In *ICCV*.

Xiaohan Wang, Yuhui Zhang, Orr Zohar, and S. Yeung-Levy. 2024. Videoagent: Long-form video understanding with large language model as agent. In *ECCV*, volume abs/2403.10517.

Jason Wei, Zhiqing Sun, Spencer Papay, and 1 others. 2025. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*.

Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. 2024. Longvideobench: A benchmark for long-context interleaved video-language understanding. *NeurIPS*, 37.

Jinming Wu, Zihao Deng, Wei Li, and 1 others. 2025. Mmsearch-r1: Incentivizing lmms to search. *arXiv preprint arXiv:2506.20670*.

Junbin Xiao, Xindi Shang, Angela Yao, and Tat seng Chua. 2021. Next-qa: Next phase of question-answering to explaining temporal actions. *CVPR*.

Chenyu Yang, Xuan Dong, Xizhou Zhu, and 1 others. 2025. Pvc: Progressive visual token compression for unified image and video processing in large vision-language models. In *CVPR*.

Zhilin Yang, Peng Qi, Saizheng Zhang, and 1 others. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proc. of EMNLP*.

Zongxin Yang, Guikun Chen, Xiaodi Li, Wenguan Wang, and Yi Yang. 2024. DoraemonGPT: Toward understanding dynamic scenes with large language models. In *Proc. of ICML*.

Huaying Yuan, Zheng Liu, Junjie Zhou, Ji-Rong Wen, and Zhicheng Dou. 2025. Videodeepresearch: Long video understanding with agentic tool using. *arXiv preprint arXiv:2506.10821*.

Zihao Yue, Yepeng Zhang, Ziheng Liang, and 1 others. 2024. Movie101v2: Improved movie narration benchmark. *arXiv preprint arXiv:2404.13727*.

Chenkai Zhang, Yiming Xie, Hang Gao, and 1 others. 2025. Seriesbench: A benchmark for narrative-driven drama series understanding. In *CVPR*.

Peiyuan Zhang, Kaichen Zhang, Bo Li, and 1 others. 2024. Long context transfer from language to vision. *Trans. Mach. Learn. Res.*, 2025.

Long Zhao, Nitesh B. Gundavarapu, Liangzhe Yuan, and 1 others. 2024. VideoPrism: A foundational visual encoder for video understanding. In *Proc. of ICML*.

Yilun Zhao, Lujing Xie, Haowei Zhang, and 1 others. 2025. Mmvu: Measuring expert-level multi-discipline video understanding. *CVPR*.

A Details of Auxiliary Data Extraction

A.1 Subtitles

There are three main steps to extract accurate subtitle data. First, the subtitle region for each TV drama is manually delineated, and the subtitle segments are cropped from the original video frames based on these predefined regions. Second, PaddleOCR v4 (Cui et al., 2025) is employed to conduct frame-by-frame recognition on the cropped video frames. To cluster consecutive frames corresponding to the same sentence, we calculate the Levenshtein distance between the recognition results of adjacent frames and perform normalization. Specifically, if the normalized distance between two adjacent frames is less than 0.2, they are deemed to express the same sentence. For each clustered sentence, the middle frame is sampled and input into Doubao-1.5-Vision-Pro (Team, 2025a) for re-recognition; this step serves to rectify recognition inaccuracies, optimize text segmentation for overlapping or incomplete subtitles, and unify formatting consistency. Based on the above three-step processing, the accuracy rate of the extracted subtitle results—including subtitle content and start-end timestamps—can be guaranteed to exceed 99.5%.

A.2 Face IDs

A two-stage pipeline is proposed for robust character identification. First, cast information and face images of actors/actresses are collected from web sources to construct an initial face library. The correspondence between characters and actor/actress names is verified manually. Second, high-quality face instances are extracted from video data to enrich this initial library. This enrichment process processes all videos at 1 frame per second, extracting drama-related face images for each character. We design a filtering pipeline for high-quality face instance selection, including multi-face sample filtering, high-similarity deduplication (threshold 0.8), and low-quality face filtering (threshold 0.45). Finally, 15 images are randomly sampled from the remaining face pictures to form a refined face library, where human verification is also employed. The enriched library is then used with the InsightFace library (Ren et al., 2023) to scan every frame for recognizing and labeling all face instances.

A.3 Speaker IDs

We assume that each time-aligned subtitle line corresponds to a single speaker—a condition that holds true in over 99% of scenarios. Voiceprint features are extracted using a pre-trained ERes2Net model (Chen et al., 2024b). The main assignment algorithm uses a two-stage approach: (1) clustering to construct a high-confidence voiceprint library for all known roles, and (2) label propagation to allocate speaker IDs to the remaining subtitles while identifying outlier entries. Overall, it takes around 1 GPU hour for speaker feature extraction and approximately 0.2 CPU hours for label clustering and propagation per drama.

A.4 Video Clip Captioning

The generation of video caption data consists of two main steps: using a multimodal model to describe short video clips, and leveraging a large language model (LLM) to aggregate the underlying annotation results. First, key frames are extracted for each video clip at 1 frame per second. Subsequently, Gemini 2.5 Pro is used to generate captions via a designed prompt template. To address the excessive total length of base clip captions (over 1 million words per drama), an LLM aggregates the clips into sequential, non-overlapping sections. The grouped clips are summarized into higher-level captions, laying a foundation for plot comprehension and downstream QA construction.

B Additional Details of Benchmark Construction

B.1 Auxiliary-Enriched Caption Source

Before constructing QA samples, we first obtain dense narrative captions for each drama. This captioning stage is performed outside the QA generation codebase, but it provides the raw knowledge source consumed by the event parsing and question generation pipeline. We extract four types of auxiliary information to support video captioning: subtitles, on-screen text, face identities, and speaker identities. Subtitles and on-screen text are extracted using PaddleOCR (Cui et al., 2025), refined by Doubao-1.5 (Team, 2025a), and manually checked on 1% of the data. Faces are detected and recognized with InsightFace (Ren et al., 2023). For speaker recognition, we first extract voiceprint features using a pre-trained ERes2Net model (Chen et al., 2024b), followed by clustering and label propagation.

We estimate the recognition accuracy to be approximately 99.8% for subtitles at the character level, 98% for face identities at the instance level, and 90% for speaker identities at the subtitle-line level. Although minor recognition errors remain, these auxiliary signals substantially improve the richness and grounding quality of clip-level captions.

We use SceneDetect (Castellano) to partition each drama into short clips, each with a maximum length of 15 seconds. Key frames are sampled at approximately one frame per second. We then prompt Gemini 2.5 Pro (Team, 2024) to generate detailed clip-level descriptions. Auxiliary information is injected into the captioning process in two ways: subtitles and predicted speaker identities are inserted into the text prompt, while face identities are visually integrated by rendering a red bounding box and the corresponding role name next to each recognized character in sampled frames. The MLLM is asked to produce approximately 200 words of description for each 15-second clip. Since the resulting clip-level captions can exceed one million words per drama, we further use an LLM to aggregate them into sequential, non-overlapping higher-level captions. These higher-level captions serve as the raw input to the structured event extraction stage described below.

B.2 Data Construction Pipeline

This section provides additional details for constructing QA samples from the auxiliary-enriched captions above. The pipeline converts long-form drama captions into structured event evidence, constructs multi-granularity evidence views and evidence graphs, samples reasoning-oriented QA candidates, and uses LLMs to generate and validate natural-language questions. We construct four question types: *two-event riddles*, *two-hop multi-hop questions*, *three-hop multi-hop questions*, and *riddle–multi-hop compositional questions*.

Pipeline stages. Table 4 summarizes the end-to-end construction pipeline, from auxiliary signal extraction to final benchmark filtering. After auxiliary-enriched captioning, the QA construction stage proceeds as follows: higher-level captions are parsed into structured events with a caption-to-event LLM prompt; character mentions are normalized into canonical identities with human assistance; each event is converted into multi-granularity evidence views; evidence graphs connect views with characters and semantic entities; candidate QA structures are sampled from the graphs under type-specific uniqueness and dependency constraints; a generation LLM rewrites masked evidence into natural-language questions; and generated questions are checked by deterministic rules, an LLM validator, posterior shortcut probes, and human validation.

Model configuration. We use GPT-5.3 for structured event parsing and all QA generation stages. We use Qwen3-32B as the automatic validator for generated questions, followed by posterior shortcut filtering and human validation.

B.3 Structured Event Representation

Event schema. Each caption is parsed into one or more structured events. An event is represented as a semantic frame containing: (1) the event context, including scene background, temporal context, and narrative anchors; (2) participants explicitly involved in the event; (3) the main action or interaction; (4) the interaction target, object, or counterparty; (5) the location; (6) salient objects; (7) the original evidence span; and (8) characters that can be inferred as potential answers. This representation converts free-form narrative descriptions into explicit event-level evidence units, enabling graph-based sampling, uniqueness diagnosis, and answer verification.

Multi-granularity evidence views. For each structured event e, we construct a set of evidence views:

$$\mathcal{V}(e) = \{v_e^{\text{event}}, v_e^{G1}, v_e^{G2}, v_e^{G3}\}. \quad (1)$$

Here, v_e^{event} denotes the full event view, while v_e^{G1}, v_e^{G2}, and v_e^{G3} provide progressively more specific descriptions. G1 is the minimal self-contained view, G2 adds discriminative details, and G3 adds a small number of additional verifiable facts. The multi-granularity design allows the sampler to choose evidence that is informative enough for uniqueness while avoiding overly explicit descriptions that would leak the answer.

Character normalization. Characters in TV dramas may be referred to by full names, first names, nicknames, titles, kinship terms, or relational descriptions. To avoid fragmenting evidence associated with the same character, we perform human-assisted character normalization. Clear aliases are

Stage	Purpose	Model or supervision	Output
Auxiliary signal extraction	Recover subtitles, on-screen text, faces, and speakers	OCR / face recognition / speaker recognition	grounded auxiliary cues
Auxiliary-enriched captioning	Describe short clips and aggregate them into higher-level captions	Gemini 2.5 Pro + LLM aggregation	higher-level narrative captions
Event parsing	Convert captions into structured event frames	GPT-5.3	structured event frames
Character normalization	Merge aliases into canonical identities	human-assisted verification	canonical character inventory
Evidence view construction	Convert each event into multi-granularity views	GPT-5.3	G1/G2/G3 evidence views
Evidence graph construction	Connect views with characters and semantic entities	deterministic graph construction	evidence graphs for candidate diagnosis
Candidate sampling	Select valid reasoning structures under uniqueness constraints	deterministic constraints	riddle, multi-hop, and compositional candidates
Mask-driven question generation	Rewrite masked evidence into natural-language QA	GPT-5.3	generated QA samples
Hierarchical validation and filtering	Filter leakage, hallucination, and shortcut cases	deterministic checks + Qwen3-32B + GLM-4.6V posterior probe + human validation	final benchmark samples

Table 4: Overview of the data construction pipeline. The table describes the methodological stages rather than implementation-specific scripts.

mapped to canonical character identities, while ambiguous mentions such as "the man", "the woman", or underspecified kinship references are not aggressively resolved unless the reference is explicit. All downstream operations, including graph construction, candidate diagnosis, uniqueness verification, and sample deduplication, are performed over canonical character names.

B.4 Graph Construction and Candidate-set Diagnosis

We construct two bipartite evidence graphs from the evidence views. Let $\mathcal{Y}$ be the set of canonical characters, $\mathcal{Z}$ be the set of non-character semantic entities, and $\mathcal{V}$ be the set of all evidence views. The Character–View graph is

$$\mathcal{G}_{\text{char}} = (\mathcal{Y}, \mathcal{V}, \mathcal{E}_{\text{char}}), \tag{2}$$

where an edge $(y, v) \in \mathcal{E}_{\text{char}}$ indicates that character y is answerable or explicitly involved in view v. The Entity–View graph is

$$\mathcal{G}_{\text{ent}} = (\mathcal{Z}, \mathcal{V}, \mathcal{E}_{\text{ent}}), \tag{3}$$

where an edge $(z, v) \in \mathcal{E}_{\text{ent}}$ indicates that entity z is semantically grounded in view v.

For a target answer y and an evidence view v, we extract a clue signature $S(v, y)$ consisting of entities, actions, locations, objects, and contextual anchors. The target answer and its aliases are removed from $S(v, y)$ to avoid self-revealing clues. Let

$$\mathcal{N}_{\text{ent}}(s) = \{u \in \mathcal{V} \mid (s, u) \in \mathcal{E}_{\text{ent}}\} \tag{4}$$

denote the graph neighbors of clue s. Views compatible with the clue signature are

$$\mathcal{R}(v, y) = \bigcap_{s \in S(v,y)} \mathcal{N}_{\text{ent}}(s). \tag{5}$$

The candidate answer set is then induced by the Character–View graph:

$$\text{Cand}(v, y) = \bigcup_{u \in \mathcal{R}(v,y)} \{y' \in \mathcal{Y} \mid (y', u) \in \mathcal{E}_{\text{char}}\}. \tag{6}$$

A view is uniquely identifying if and only if

$$\text{Cand}(v, y) = \{y\}. \tag{7}$$

This diagnosis is used both to select informative evidence and to reject candidates that are either under-specified or too revealing.

B.5 Candidate Sampling Strategies

Two-event riddle sampling. Given a target character y, we sample two evidence views v_1 and v_2 derived from two distinct events in different captions. For each view v_i, we compute the candidate answer set Cand_i. A valid two-event riddle must satisfy

$$|\mathrm{Cand}_i| > 1, \quad i \in \{1, 2\}, \tag{8}$$

and

$$\mathrm{Cand}_1 \cap \mathrm{Cand}_2 = \{y\}. \tag{9}$$

The first condition ensures that neither event alone uniquely identifies the target, while the second condition ensures that the target becomes unique only after combining the two clues. Before generation, the target character and all aliases are masked from both views. Candidates are rejected if masking fails or if either view contains explicit identity leakage.

Multi-hop sampling. Multi-hop questions are constructed with a reverse-chain strategy. Given a target hop number K, we first sample a final answer $a^\star$ and an endpoint evidence view supporting it. We then recursively search backward for bridge entities or intermediate characters connecting the current hop to a previous supporting view. Once a valid chain is obtained, it is reversed into the natural reasoning order:

$$h_1 \to h_2 \to \cdots \to h_K \to a^\star. \tag{10}$$

Each hop h_i contains a supporting view, a local target, a bridge condition, and the corresponding evidence span. We require different hops to come from different captions, enforce non-trivial bridge conditions, and deduplicate chains with structural keys. For each hop, we also verify that removing the bridge information supplied by the previous hop makes the current hop non-unique or underspecified, reducing shortcut reasoning.

Riddle–multi-hop composition. A compositional question combines evidence aggregation and sequential reasoning:

$$\mathcal{V}_{\mathrm{mc}} \Rightarrow r, \qquad (r, \mathcal{V}_{\mathrm{mh}}) \Rightarrow a^\star. \tag{11}$$

Here, $\mathcal{V}_{\mathrm{mc}}$ denotes the two-event riddle evidence, r is the intermediate riddle answer, $\mathcal{V}_{\mathrm{mh}}$ denotes the multi-hop evidence, and $a^\star$ is the final answer. We first select a validated two-event riddle whose answer is r, then build a targeted two-hop chain conditioned on r. Captions used by the initial riddle and those used by the multi-hop chain are required to be non-overlapping. The final generated question is a single compositional query rather than a concatenation of two questions, and it must not explicitly reveal either r or $a^\star$.

B.6 Generation, Validation, and Filtering

Mask-driven generation. Before natural-language generation, answer-bearing entities are masked from the selected evidence views. For two-event riddles, the target character and all aliases are replaced with `[MASK]`. For multi-hop questions, intermediate bridge entities and final answers are masked in the hop evidence. For compositional questions, both the intermediate riddle answer and the final answer are prohibited from appearing in the question. The generation prompts require the model to preserve the original facts, avoid unsupported information, avoid answer leakage, and maintain the intended reasoning structure.

LLM validation. After question generation, we use an LLM-based validator to assess the semantic quality of each sample. The validator checks faithfulness to the supporting evidence, answer uniqueness, factual consistency, identity leakage, type consistency, and shortcut risk. Samples that fail validation are either rewritten under the validator's feedback or discarded, depending on the retry configuration.

Posterior shortcut evaluation. We further conduct posterior evaluation as a closed-book shortcut probe. A model is asked to answer each generated question without access to the video, supporting captions, or structured evidence. Since the questions are intended to require video-grounded clues, a question that can be reliably answered from the text alone may contain shortcut cues or implicit answer leakage. For robustness, we query the model twice; if both responses match the ground-truth answer, the sample is discarded.

Human validation. All final samples undergo human validation. Three annotators independently verify each sample for answer correctness, uniqueness under the given clues, and naturalness of the question phrasing. Samples that fail any criterion are revised or replaced.

B.7 Prompt Templates

This section reports the prompt templates used by the pipeline. We preserve the operational content

of the prompts while formatting them for readability in the supplementary material. The caption-to-event parsing prompt is shown in Fig. 5–8. The QA generation and validation prompts are reported for two-event riddles in Fig. 9 and Fig. 10, for two-hop and three-hop multi-hop questions in Fig. 11–12 and Fig. 13, and for riddle–multi-hop compositional questions in Fig. 14 and Fig. 15.

B.8 Benchmark Statistics

After the full construction and validation pipeline, we obtain a balanced benchmark across dramas and riddle types. For each TV drama, we retain 100 riddles for each of the four fine-grained riddle types. Since our benchmark contains 9 dramas, the final benchmark contains

$$9 \times 4 \times 100 = 3{,}600 \tag{12}$$

riddles in total. A 10th drama, *A Lifelong Journey*, is processed by the same pipeline but used solely to construct SFT and RL training data for our baseline agent and is excluded from the benchmark. The benchmark can be sliced by drama, riddle type, or reasoning structure for fine-grained analysis.

C Training Details

C.1 SFT Configuration

We fine-tune Qwen3-8B using the following hyperparameters: learning rate 1×10^{-5} with cosine decay, batch size 16, maximum sequence length 8192 tokens, 3 training epochs. The training is conducted on 8 NVIDIA A100 GPUs and takes approximately 4 hours.

C.2 RL Configuration

For the GRPO-based RL stage, we use the following settings: learning rate 5×10^{-7}, batch size 128, group size $G = 8$, clipping parameter $\varepsilon = 0.2$, KL penalty coefficient $\beta = 0.01$. The reward coefficients are $\lambda_1 = 0.3$ (grounding) and $\lambda_2 = 0.5$ (format). Training is conducted on 8 NVIDIA H800 GPUs for 800 steps.

The GRPO objective optimized in Stage 2 is:

$$r_i(\theta) = \frac{\pi_\theta\left(o_i \mid q\right)}{\pi_{\theta_{\text{old}}}\left(o_i \mid q\right)}, \quad \hat{r}_i(\theta) = \text{clip}\left(r_i(\theta), 1-\varepsilon, 1+\varepsilon\right),$$

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{\substack{q \sim P(Q) \\ \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}}} \left[\frac{1}{G} \sum_{i=1}^{G} \min\left(r_i(\theta) A_i, \hat{r}_i(\theta) A_i\right) \right] - \beta \mathbb{D}_{\text{KL}}\left(\pi_\theta || \pi_{ref}\right),$$

where q is a riddle, $\{o_i\}_{i=1}^{G}$ are G sampled trajectories, and A_i is the group-relative advantage of trajectory o_i.

D Additional Case Studies

We provide additional examples of our agent's reasoning trajectories in this section. These examples illustrate the agent's ability to decompose complex riddles, select appropriate tools, and integrate multi-source evidence across multiple search rounds. Due to space constraints, we present abbreviated versions of the trajectories, highlighting the key reasoning steps and tool invocations.

Caption-to-Event Parsing Prompt (Part 1/4)

```
You are a strict film and TV plot event parsing assistant. Parse the input caption data into
     structured events and output valid JSON only.

Task order:

1. Extract the full-caption background information into `caption_context`
2. Segment the main events into `events`
3. Complete the event frame for each event first
4. Generate G1/G2/G3 from the completed event frame
5. Extract `named_characters` for each event
6. Extract `core_entities` and `auxiliary_entities` for each G1/G2/G3

[1. caption_context]
`caption_context` stores background anchors for the whole caption and must not be output as standalone
      events. It includes:

* time anchors
* location anchors
* scene background
* news, narration, or subtitle background

Pure establishing shots, scenery descriptions, camera language, narration, or subtitle-only background
      should usually go into `caption_context`, not into `events`.
Do not treat narrative devices such as "shot", "frame", "narration", "subtitle", or "audience" as
     character participants.

[2. Event segmentation]
An event must satisfy all of the following:

* It has exactly one main action, main change, or main reaction
* It has clear core participants
* It is self-contained when read alone
* It at least expresses "who + did what + to whom/with whom/affected what"

Segmentation rules:

* Consecutive actions may stay in one event if they serve the same main action
* Split into a new event when the action center, leading character, result, or time progression
     clearly changes
* Do not create standalone events for pure atmosphere, pure summary, or abstract generalization
* Weak events such as "passing by a place" or "the camera follows movement" should preferably be
     merged into adjacent events or placed in `event_context` unless they introduce a new key
     character, key object, key location, or askable fact
* A roughly 1-minute caption usually contains 2 to 8 events
* `evidence_span` must be a continuous or near-continuous verbatim span from the caption, with no
     modification
* Do not concatenate text spans that are far apart
* Number `event_id` in order, such as `545_e01`, `545_e02`
```

Figure 5: Caption-to-event parsing prompt, Part 1: task definition, caption context, and event segmentation rules.

Caption-to-Event Parsing Prompt (Part 2/4)

```
[3. Event completion]
For each event, internally complete:

* omitted or pronominal core participants
* the action object or dialogue counterpart
* the current scene when necessary
* directly relevant background information from `caption_context`

Requirements:

* Only use information clearly supported by the original text and local context
* Do not invent information absent from the original text
* If the original text only gives a descriptive identity such as "an adult" or "a child", do not
    complete it as "mother", "partner", "wife", etc. unless the context clearly confirms it
* Use `""` or `[]` when information cannot be reliably determined

[4. Event grading]
Each event must output G1/G2/G3 as different granularities of the same event.

G1 (minimal self-contained level):

* Must contain `agent + action + patient/counterparty/target`
* Do not write vague descriptions such as "has a dispute", "has a conversation", or "has complex
    emotions"
* Do not use ambiguous pronouns such as "he", "she", "they", or "that person"

G2 (disambiguation-enhanced level):

* Add 1 to 2 of the most discriminative details on top of G1
* Prefer location, key object, dialogue topic, or explicit counterpart identity

G3 (complete-description level):

* Add 1 to 2 supplementary facts on top of G2
* Prefer manner/tone/obvious emotion, cause, or result/consequence
* Only add verifiable facts; do not add theme or atmosphere
* Do not add thematic summaries, atmosphere summaries, or literary phrasing
* Avoid lightly subjective inference or interpretive words, such as "symbolizes", "reflects", "echoes
    ", "means", "shows", "reveals", "highlights", "warmth", "growth", "inheritance", "resilience", "
    seems", "as if", "suggests", "indicates", or "explains"

Additional requirements for dialogue events:

* Write `action` as a verifiable action whenever possible, such as "proposes", "refutes", "reminds", "
    questions", "answers", "confirms", or "gives"
* Avoid abstract summaries such as "communicates", "misunderstands", or "has an ideological clash"

[5. named_characters]
For each event, extract `named_characters`:

* Only extract character names that can be clearly identified after event completion
* Keep only explicit names or name-like role labels
* Do not include generic descriptions such as "a man", "a child", "family", or "staff" in `
    named_characters`
* If no clearly identifiable character name appears in the event, use `[]`
```

Figure 6: Caption-to-event parsing prompt, Part 2: event completion, grading, and character extraction rules.

Caption-to-Event Parsing Prompt (Part 3/4)

```
[6. Key entity extraction]
For each G1/G2/G3, extract:

* `core_entities`: key entities required for that level's event to hold; keep them minimal but
    sufficient
* `auxiliary_entities`: useful but nonessential supplementary entities

Entity types are limited by default to:

* character/person
* key object
* location
* organization/institution
* proper noun
* explicit event topic term

Requirements:

* `core_entities` should preferably be directly found in, or clearly correspond to, that level's `
    description`
* Do not treat pure emotion words, modifiers, or abstract evaluation words as entities
* When a character identity is unclear, keep the original descriptive label, such as "the adult in a
    dark coat" or "the child wearing a green hat"
* Do not replace specific descriptions with overly abstract group words such as "family", "crowd", or
    "people"

[7. event_frame requirements]
`event_frame` contains:

* `core_participants`
* `context_participants`
* `action`
* `target_or_counterparty`
* `location`
* `key_object`
* `cause`
* `result`

Requirements:

* Fill `cause/result` only when directly supported by the original text or nearby context
* Use `""` when the content is only a reasonable guess, thematic summary, or comment
* Do not put narrative devices into `core_participants`
```

Figure 7: Caption-to-event parsing prompt, Part 3: key entity extraction and event-frame requirements.

Caption-to-Event Parsing Prompt (Part 4/4)

```
[8. Output requirements]

* Output valid JSON only
* Do not output explanations
* Use `""` or `[]` for uncertain information
* Keep `evidence_span` verbatim from the input caption

Output format:

```json
{
  "captions": [
    {
      "caption_id": "xxx",
      "caption_context": {
        "time_anchor": "",
        "location_anchor": "",
        "scene_anchor": [],
        "news_or_narration": []
      },
      "events": [
        {
          "event_id": "xxx_e01",
          "evidence_span": "",
          "event_context": {
            "time_anchor": "",
            "location_anchor": "",
            "scene_anchor": [],
            "news_or_narration": []
          },
          "named_characters": [],
          "event_frame": {
            "core_participants": [],
            "context_participants": [],
            "action": "",
            "target_or_counterparty": [],
            "location": "",
            "key_object": [],
            "cause": "",
            "result": ""
          },
          "grading": {
            "G1": {
              "description": "",
              "core_entities": [],
              "auxiliary_entities": []
            },
            "G2": {
              "description": "",
              "core_entities": [],
              "auxiliary_entities": []
            },
            "G3": {
              "description": "",
              "core_entities": [],
              "auxiliary_entities": []
            }
          }
        }
      ]
    }
  ]
}
```
```

Figure 8: Caption-to-event parsing prompt, Part 4: output requirements and JSON schema.

Two-Event Riddle Generation Prompt

```
You are a drama QA data annotation expert. Based on the given clue views, generate one high-quality
     riddle-style question in English.

Your task has two steps:
Step 1: Check whether the sample is suitable for generation.
Step 2: Only if it is suitable, output the question.

Important notes:
The input description is internal evidence, not the riddle text shown directly to users.
Text in the description that matches the answer character has been anonymized as [MASK].
You must extract clues from the anonymized evidence and rewrite them into a question, but you must not
      copy [MASK] into the output.

--------------------------------
[Stage 1: Suitability Check]

Only decide whether a question should be generated. Do not try to write the question yet.

Must follow:
1) Judge only from each view's description field; do not use or implicitly introduce any other fields.
2) If the description still contains obvious hints to the answer identity outside [MASK] (such as
     surname, stable alias, direct form of address, unique title, etc.), you must reject.
3) If a natural and fluent question cannot be written from the description alone, reject.
4) If the question would require invented chronology or causality to be coherent, reject.
5) If the question is hard to make natural without leaking the answer, reject.
6) If the description contains a relationship-chain clue that uniquely identifies the answer (such as
     someone's son, mother, husband, wife, father, etc.) and it is hard to rewrite naturally without
     leaking identity, reject.

--------------------------------
[Stage 2: Question Generation]

Generate only if Stage 1 passes.

Must follow:

1) Generate only from the description fields; do not implicitly introduce any other information.
2) Do not directly mention the answer character's name.
3) Do not mention the answer character's surname, stable alias, direct form of address, unique title,
     or any information that would clearly leak the answer identity.
4) Do not invent chronology or causality across clues. Avoid words and phrases such as "later", "
     afterward", "subsequently", "then", "after that", "as a result", "therefore", "leading to", etc.
5) Do not add facts that are not explicitly provided in the input, and do not rely on information
     outside the descriptions.
6) Prefer a parallel clue-listing style, presenting each clue as a relatively independent hint.
7) Avoid narrative plot retelling. Do not force multiple clues into a complete story.
8) You may flexibly use patterns such as:
    * This person once ..., and in another scene also .... Who is this person?
    * In one scene, ...; in another scene, .... Who is this character?
    * The following clues point to the same person: ...; .... Who is it?
    * One clue is ...; another clue is .... Which character do these clues indicate?
    Prefer one of these patterns after rewriting; if you do not use them, still keep a parallel clue-
         listing style.
9) The question should be concise and natural, preferably 1 to 2 sentences rather than a long
     paragraph.

--------------------------------
[Output Requirements]

Output the unified JSON structure only. Do not output any extra explanation:

{
  "status": "ok|reject",
  "leak_risk": 1,
  "naturalness_score": 1,
  "hallucination_risk": 1,
  "question": "...",
  "answer": "...",
  "reject_reason": "..."
}

Rules:

- When status=reject, question and answer must be empty strings, and reject_reason must be filled.
- When status=ok, answer must exactly equal answer_character, and reject_reason must be an empty
     string.
- All scores are in the range 1 to 5:
  - leak_risk: higher means greater leakage risk.
  - hallucination_risk: higher means more likely to induce hallucination.
  - naturalness_score: higher means the question is more natural.
```

Figure 9: Prompt for generating two-event riddle questions from masked clue views.

Two-Event Riddle Validation Prompt

```
You are a drama riddle-question validation expert. Judge whether the given question satisfies the data
      construction requirements.

Criteria:
1) question must not contain the answer character name answer_character.
2) question should be a natural and readable riddle-style question with appropriate length, not
     gibberish or a fragment.
3) question should not obviously leak identity, such as through direct address, stable alias, unique
     title, etc.
4) question should be based on the anonymized clues and should not invent facts.
5) question should keep a riddle-style ending such as "Who is this person?" or "Which character is
     this?"

Output JSON only. Do not output extra text:
{
  "status": "pass|fail",
  "issues": ["..."],
  "reason": "...",
  "answer_mentioned": true,
  "format_ok": true,
  "naturalness_score": 1
}
```

Figure 10: Prompt for validating generated two-event riddle questions.

Multi-Hop QA Generation Prompt (Part 1/2)

```
You are a multi-hop QA data construction expert. Based on the input hop evidence, generate a question
     that must be solved step by step through a chained sequence of subquestions.

Core definitions:
- Subquestion chain: subquestion 1 -> subquestion 2 -> ... -> subquestion N -> final answer (a person)
     .
- For i>=2, subquestion i must depend on the answer to subquestion i-1; if any hop is removed, the
     reasoning chain should break.
- Multi-hop reasoning should come from identity carryover, relationship carryover, or semantic
     carryover, not from chronology or explicit step-by-step wording.
- The surface form of the question should be natural and readable. Prefer parallel clues or clause-by-
     clause progression plus one final question, and avoid templated long attributive chains.

Must satisfy these constraints:
1. Each hop (i>=2) must not independently determine that hop's answer before the previous hop answer
     is known:
   - either that hop's answer is not unique;
   - or that hop question is incomplete and must rely on the previous answer.
2. Each hop must be uniquely solvable after the previous hop answer is known.
3. Removing the previous-hop information must prevent unique identification at the whole-series scope:
   - previous-hop information alone cannot directly answer the final question;
   - the final-hop information also cannot bypass the previous hop and answer directly.
4. Bridge answers must not be mechanically repeated as explicit labels in later questions:
   - do not directly insert the previous answer into the next hop as a template like "Who is the
       person holding X?"
   - implicit carryover through role function, relationship semantics, or scene constraints is allowed
       .
5. question must be a single question with exactly one final answer:
   - do not write two or more parallel questions, such as "Who ...? And who ...?"
   - do not expose multiple hops as "first answer A, then answer B".
6. The final solving target of question must exactly match the final-hop answer:
   - do not ask about A in the subquestion chain while the overall final answer is B.
7. Surface style:
   - Split clues from different events into clauses or sentences separated by semicolons or periods.
       All clauses must serve the same final question.
   - Avoid stiff concatenation in the form of "very long front modifier + final question".
   - Keep the question natural and concise, avoiding overly long single sentences.
8. The question text must rely only on input description_masked; do not invent extra facts.
9. question must not contain chronology words or phrases such as:
   - later, afterward, afterwards, subsequently, after that, then, next, eventually, soon after,
       thereafter
10. The final answer must exactly equal input answer_character.
11. question sentence-form requirements:
    - question must explicitly use semicolons ";" or periods "." to split clues from different hops
        into multiple clauses or sentences.
    - Key information from each hop should occupy a relatively independent clause or sentence, forming
        a clear chain.
    - Earlier clauses or sentences provide preceding hop clues in order, and the final clause or
        sentence provides the last-hop clue and asks the final question.
```

Figure 11: Prompt used for both two-hop and three-hop multi-hop question generation, Part 1: task definition and generation constraints.

Multi-Hop QA Generation Prompt (Part 2/2)

```
Positive examples:
   - "In one clue, someone emotionally reminds An Xin that he is not the only one who has been hurt
        and bled when An Xin despairs that he cannot even shoot accurately; in another clue, while
        calling An Xin at home and revealing the true purpose of the guidance team, what hairstyle is
        this person shown with?"

Negative examples:
   - "Who is the person who spoke in ... and later ...?"
   - "Who is the character in ... who ...?"
   - "... the person who ..., later when ..."
   - "First answer A, then answer B"

Output JSON (no extra text):
{
  "status": "ok|reject",
  "question": "...",
  "reasoning_chain": ["hop1_answer=...", "hop2_answer=...", "..."],
  "answer": "...",
  "reject_reason": "..."
}

Rules:
- When status=ok:
  - question must be filled;
  - reasoning_chain must be filled;
  - answer must be filled and must equal the final person answer;
  - reject_reason must be an empty string.
- When status=reject:
  - question must be an empty string;
  - reasoning_chain must be an empty array;
  - answer must be an empty string;
  - reject_reason must clearly explain why generation is impossible.
- Output reject only in the following cases:
  1) the input hop evidence cannot form a strict dependency chain;
  2) the input hop evidence is insufficient to support a unique answer;
  3) connecting the hops into a natural question would require invented facts, added relationships, or
       added chronology not present in the input;
  4) the chain cannot be expressed as a single question with a single answer without becoming a double
       /multiple question or a severely awkward structure;
  5) the input hops conflict clearly and make a coherent question impossible.
- Do not reject merely because the wording is not elegant. Reject only when the chain is invalid, the
     answer is not unique, facts must be invented, or double-question/severe-structure errors cannot
     be avoided.
- Return JSON only. Do not add explanations, comments, or markdown.
```

Figure 12: Prompt used for both two-hop and three-hop multi-hop question generation, Part 2: examples, rejection rules, and JSON format. The required hop count is supplied in the input payload.

Multi-Hop QA Validation Prompt

```
You are a multi-hop drama question validator. You will receive a generated question and its chain
     context. Judge strictly whether it satisfies the rules.

Focus on:
1) Whether question contains banned chronology words or phrases: later, afterward, afterwards,
     subsequently, after that, then, next, eventually, soon after, thereafter.
2) Whether question is essentially a double/multiple question or contains two independent solving
     targets.
3) Whether question relies heavily on an awkward "very long front modifier + final question" template.
4) Whether question only uses input hops[*].description_masked and introduces no new facts.
5) Whether the solving target of question matches expected_answer, not another person or object.
6) Whether question is natural, readable, and clear, rather than mechanically stitched.
7) Do not default to a "the person who X, who is Y?" structure; prefer expanding different hops
     separately and asking only one final question.
8) Whether question satisfies the required num_hops:
   - It must reflect a continuous and indispensable chain of reasoning; the effective number of
        reasoning steps must match num_hops.
   - For example, when num_hops=3, the question must require at least 3 chained solving steps to reach
         the answer; if it can be compressed to 2 steps, it is invalid.
   - If hops are insufficient, the chain is broken, or the answer can be reached by skipping steps,
        mark fail.

Output JSON only. Do not output extra text:
{
  "status": "pass|fail",
  "hop_count_ok": true,
  "estimated_hops": 0,
  "issues": ["..."],
  "reason": "..."
}

Requirements:
- If any hard violation exists, status=fail.
- When status=pass, hop_count_ok must be true and estimated_hops >= num_hops.
- When status=pass, issues may be an empty array.
- Output must be a valid JSON object.
```

Figure 13: Prompt for validating generated two-hop and three-hop questions.

Riddle–Multi-Hop Compositional Generation Prompt

```
You are an expert in constructing combined "riddle + multi-hop" questions.

You will receive two existing questions:
- A riddle question (with question and answer)
- A two-hop multi-hop question (with question and answer)

Generate one new combined question that must satisfy:
1) The reasoning order must be exactly 3 steps:
   - Step 1: solve the riddle answer first;
   - Step 2: use that answer as the key clue for hop 1 of the multi-hop question to get the
       intermediate answer;
   - Step 3: complete hop 2 of the multi-hop question to get the final answer.
2) The combined prompt must be a single question, not double/multiple questions.
3) The combined question must not leak answers:
   - The riddle answer must not appear directly in question;
   - The final answer (i.e., multi-hop answer) must not appear directly in question.
4) The combined question must be based only on the provided two questions and their answers. Do not
     add new facts.
5) The combined question should be natural and readable, not a rigid template:
   - Do not simply concatenate the riddle text and the multi-hop text;
   - riddle_answer must naturally act as the bridge entity or bridge reference in the downstream
       reasoning chain;
   - question should read like one organically valid question, not a stitched artifact.
6) The output answer must be exactly equal to the input multi-hop answer.
7) Style constraints:
- Keep language natural, concise, and readable; avoid rigid templating;
- Avoid long stacked modifiers;
- Do not use explicit chronology connectors such as: later, afterward, then, subsequently, after that,
      soon after.

Output JSON only (no extra text):
{
  "status": "ok|reject",
  "question": "...",
  "reasoning_chain": [
    "step1_riddle_answer=...",
    "step2_multihop_hop1_answer=...",
    "step3_final_answer=..."
  ],
  "answer": "...",
  "reject_reason": "..."
}

Rules:
- status=ok:
  - question must be non-empty;
  - reasoning_chain must contain at least 3 steps;
  - answer must equal the given final_answer exactly;
  - reject_reason must be empty.
- status=reject:
  - question must be empty;
  - reasoning_chain must be an empty array;
  - answer must be empty;
  - reject_reason must explain why.
- Return JSON only, no additional text.
```

Figure 14: Prompt for synthesizing a single riddle–multi-hop compositional question.

Riddle–Multi-Hop Compositional Validation Prompt

```
You are a validator for combined "riddle + multi-hop" questions.
You will receive:
- Combined question question
- Riddle QA (question + answer)
- Multi-hop QA (question + answer)
- Expected final answer expected_final_answer
- Expected reasoning steps expected_steps (fixed to 3)

Please strictly validate:
1) Whether question is a single question (not double/multiple questions).
2) Whether question leaks answers:
   - Must not directly include riddle_answer;
   - Must not directly include expected_final_answer.
3) Whether the following chain strictly holds:
   - Step 1: riddle_answer can be solved from question first;
   - Step 2: hop 1 of multi-hop must rely on Step 1 answer to get an intermediate answer;
   - Step 3: Step 2 answer must be required to reach expected_final_answer.
4) Whether the answer can be obtained without chain reasoning:
   - If Step 2 can still hold without Step 1 answer, fail;
   - If expected_final_answer can be directly identified by skipping Step 2, fail;
   - If any step can be bypassed, the question is invalid.
5) Whether question is strictly based on the given two source QAs without introducing new facts.
6) Whether the final solving target of question is indeed expected_final_answer.
7) Text quality:
   - Natural and readable;
   - No obvious rigid template, hard stitching, or overlong modifier chains;
   - Avoid explicit chronology connectors such as later, afterward, then, subsequently, after that.

Output JSON only (no extra text):
{
  "status": "pass|fail",
  "chain_steps_ok": true,
  "estimated_steps": 0,
  "issues": ["..."],
  "reason": "..."
}

Requirements:
- If any hard violation exists, status=fail;
- When status=pass, chain_steps_ok must be true and estimated_steps >= expected_steps;
- Output must be a valid JSON object.
```

Figure 15: Prompt for validating the three-step reasoning structure of compositional questions.