Video-LLMs with Temporal Visual Screening

Zheyu Fan[*1,2], **Jiateng Liu**[1], **Yuji Zhang**[1], **Zihan Wang**[2],
Yi R. (May) Fung[1], **Manling Li**[2], **Heng Ji**[1]
[1]University of Illinois Urbana-Champaign [2]Northwestern University
zheyufan@u.northwestern.edu hengji@illinois.edu

Abstract

Current Video-LLMs lack human-like temporal screening capabilities and their behavioral manifestations (e.g. scrubbing the seek bar), struggling with fine-grained semantics under a static, sparse frame-sampling paradigm. We introduce **T**emporal **V**isual **S**creening (TVS), a cognitively inspired task formalized as an endomorphic transformation $\mathcal{F}_{TVS} : (V, Q) \rightarrow (v, q)$ that maps a task instance to a computationally tractable, semantically coherent one within the same bimodal space, excising irrelevant segments while synchronously rewriting the query to ensure answer invariance. As a modular front-end adapter, TVS plugs into **both** Video Instruction Tuning (training) and Video Question Answering (inference) pipelines. We release the first dedicated benchmark and propose ReSimplifyIt, a baseline that surpasses prior methods from similar tasks by 0.40 F-1 with competitive query rewriting. Integrating TVS yields significant relative gains of 7.33% in training and 33.7% in inference, validating our approach for video-language understanding.[1]

1 Introduction

Human cognition optimizes information processing via a two-stage control mechanism: a fast, pre-attentive screening to prune redundancy, followed by focused reasoning on a compacted, goal-aligned representation (Treisman and Gelade, 1980; Wolfe, 1994; Zacks et al., 2007; Hochstein and Ahissar, 2002). Behaviorally, this manifests as scrubbing the seek bar to locate regions of interest before detailed viewing (Wu and Xie, 2024), minimizing Extraneous Load to purify Germane Load (Sweller, 1988; Mayer and Moreno, 2003; Paas et al., 2003), enabling **goal**-oriented problem reconstruction before further reasoning.

In contrast, current Video-LLMs (Chen et al., 2023; Li et al., 2023a; Xu et al., 2024a; Maaz et al., 2024; Li et al., 2024c; Ma et al., 2024) ingest all frames uniformly and confront the full reasoning load of the raw $(Video, Query)$ instance. Without reasoning-guided temporal skimming, superfluous perceptual burdens dilute semantic focus on critical segments and inject noise, weakening supervision signals during training (Lin et al., 2024; Zhang et al., 2025) and reducing task-oriented focus at inference.

Traditional grounding or grounded QA (Xiao et al., 2024; Chen et al., 2024a) fails to address these issues: it relies on **depictive-level** similarity and **perceptual-level** alignment to enhance *visual cues*, while the underlying problem–goal misalignment runs deeper—queries often entangle multi-hop temporal, relational, and causal dependencies with sparse, weakly localized visual evidence.

This calls for an interpretable, controllable, goal-oriented mechanism that jointly optimizes reasoning structure and cognitive load while treating input modalities **as a whole**. We propose **T**emporal **V**isual **S**creening (TVS), a cognitively inspired **endomorphic** transformation:

$$\mathcal{F}_{\text{TVS}} : (V, Q) \rightarrow (v, q)$$

that reconstructs a computationally tractable, semantically coherent instance within the same multimodal task space. TVS is goal-invariant by design, aligning with the fixity of the Goal State in Problem Space Theory (Newell and Simon, 1972), sharpening cognitive-level multimodal alignment, and focusing subsequent multimodal understanding.

TVS yields four concrete benefits: (i) Training alignment: under fixed budgets, removing off-target supervision boosts gradient signal-to-noise ratios and reduces spurious correlations; (ii) Inference robustness: offloading non-informative perceptual burdens shortens the implicit reasoning pro-

[*]Work done during internship at UIUC.
[1]Our code is available here.

Findings of the Association for Computational Linguistics: ACL 2026, pages 137–162
July 2-7, 2026

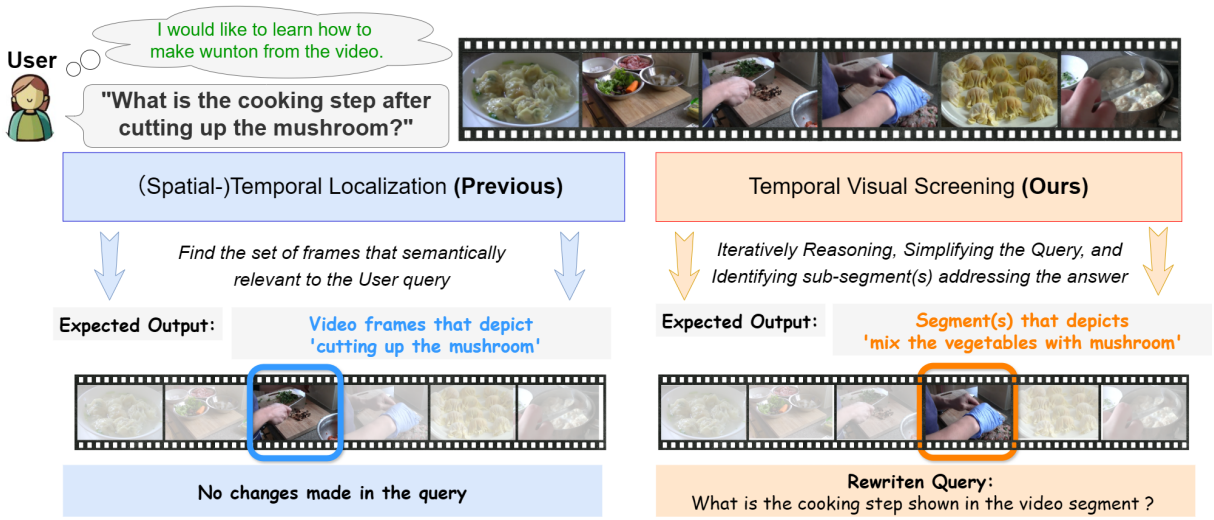

Figure 1: Input and output example of the TVS task as a side-by side comparison to a superficially similar task: temporal localization. TVS focuses on goal-driven cognitive alignment and performs reasoning-guided problem reconstruction through information screening, whereas grounding only emphasizes perceptual alignment and locates depictive level visual evidence by direct moment retrieval.

gram, unleashing reasoning capabilities; (iii) Interpretability: the compact (v, q) is a controllable, faithful artifact exposing model bottlenecks; (iv) Generalizability: as a model-agnostic front-end, TVS benefits diverse VideoQA agents.

We present *ReSimplifyIt*, a plug-and-play multi-agent framework drawing on cognitive science principles to **ReS**iliently **imp**rovise and qua**lify** proposals **It**eratively. Each screening round runs a language-only Launcher that hypothesizes a trimming instruction and rewritten query (using the intrinsic cross-modal relation as a prior), a Validator that executes and critiques these plans through a Viewer module, and memory trackers for self-correction. This operationalizes a competence-from-consequence (Brooks, 1991) principle: trial execution provides constructive feedback that tightens the coupling between hypothesized reasoning structure and available evidence.

We further construct YouCookII-TVS, the first benchmark dedicated to evaluating TVS as a joint transformation over video and query, enabling assessment of both screening quality and downstream VideoQA performance.

Through quantitative benchmarking on both downstream training and inference, we show that TVS is a non-trivial task that remains far from solved and reveals a key bottleneck for downstream VideoQA reasoning, deserving independent formulation, benchmarking, and study.

To our knowledge, ours is the first work to formally recognize that grounding and trimming a video creates a semantic mismatch with complex, context-dependent queries, to formalize the task on this premise, and to extensively explore temporal filtering within the Video-LLM framework. Our results show that TVS not only improves inference-time performance but also enables more structured and effective training through finer-grained multi-modal alignment, paving the way for scalable, interpretable, cognitively-inspired video-language understanding systems.

The main contributions of this work are summarized below:

- We propose temporal visual screening (TVS), a novel cognitively-inspired task that guides task-aware reasoning structure filtering and addresses a key bottleneck in multi-modal alignment. We also release YouCookII-TVS, a benchmark with 238.2 hours of video and 2,754 questions.

- We introduce ReSimplifyIt, the first baseline for the TVS task: a model-agnostic plug-in compatible with **any** existing videoQA or instruction-tuning pipeline.

- We quantitatively evaluate TVS in VideoQA and Video Instruction Tuning, where our method delivers over 10% and 5% absolute gains at inference and training, respectively, across multiple strong baselines and benchmarks.

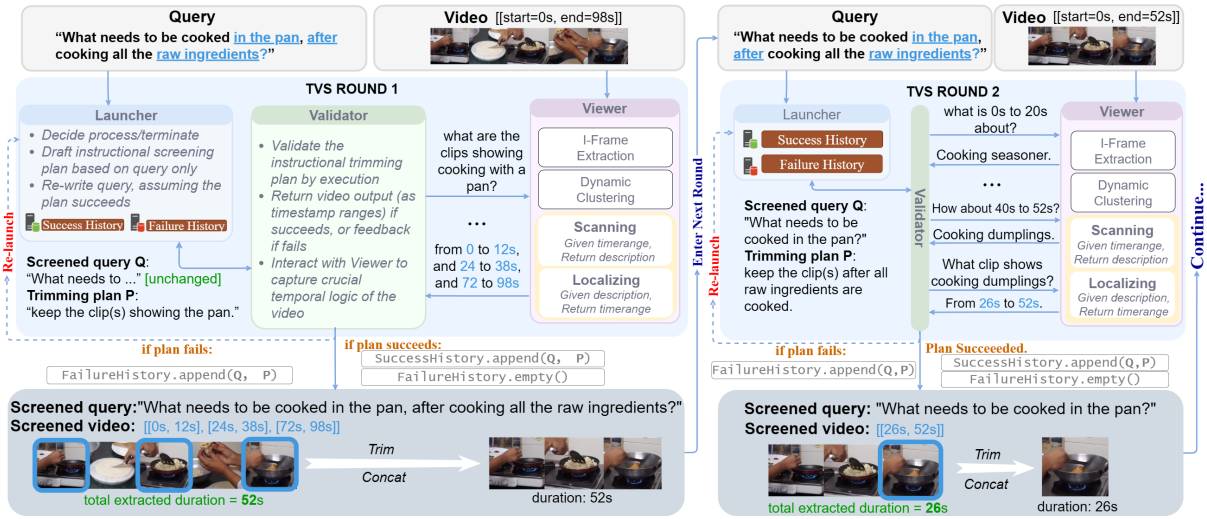

Figure 2: Snapshot examples of the workflow of our proposed ReSimplifyIt framework.

2 Formulation of Temporal Visual Screening (TVS)

2.1 Task Definition

TVS is elegantly designed to pinpoint critical video focus without altering data formats, ensuring full procedural compatibility with any existing VideoQA and Video Instruction Tuning pipelines as a front-end adapter. Refer to Figure 1 for illustration. Formally, given a full-length video $V = \{F_1, F_2, \ldots, F_T\}$ consisting of T ordered frames and a natural language query Q, the **Temporal Visual Screening (TVS)** task aims to output:

- a reconstructed query q, derived from Q, and
- a new **continuous** video

$$v = \{F_{s_1}, F_{s_1+1}, \ldots, F_{e_1},\ F_{s_2}, F_{s_2+1}, \ldots, F_{e_2},\ \ldots, F_{s_n}, F_{s_n+1}, \ldots, F_{e_n}\} \subseteq V$$

being the concatenation of n non-overlapping segments of the original video, where $1 \leq s_1 \leq e_n \leq T$ and $n \geq 1$,

such that v contains the minimal but sufficient visual information to correctly answer q, and preserves *low-level temporal continuity* essential for motion, spatial, and object-level reasoning. Specifically, we define a function:

$$\mathcal{F}_{\text{TVS}} : (V, Q) \rightarrow (v, q)$$

subject to the efficacy conditions given next.

Efficacy conditions for TVS Maintaining both *uni-modal efficacy* and *cross-modal synergy*, TVS facilitates temporal visual focus by trimming out insignificant segments of the video, while *synchronously* reconstructing the text query to preserve original semantic alignment. For demonstration, refer to the Appendix. The transformations applied to q and v must be tightly coupled to preserve answer consistency despite input reduction, ensuring semantic alignment and outcome equivalence between the reduced input and the original task. Formally, It must meet the following:

1. **Continuity for Video Output**: v must be a temporally **contiguous** subsegment of V, or a concatenation of such segments, preserving low-level visual integrity necessary for tracking motion, appearance, and scene dynamics.

2. **Minimal Sufficiency for Answer**: A frame $F_t \in v$ is included *if and only if* removing it would alter the model output: $\exists\, \delta a$ such that $\mathcal{M}(q, v \setminus \{F_t\}) = a + \delta a,\ \ |\delta a| > \epsilon$ where ϵ is a task-specific confidence or semantic tolerance threshold. This guarantees that each included frame contributes non-trivially to the answer.

3. **Reasoning Minimality for Query Output**:

$$q^* = \underset{q}{\text{argmin}}\ \text{Infer}(q)$$

where $\text{Infer}(q)$ denotes the number of reasoning steps needed to derive the answer from visual input. For example, reasoning on the phrase "within 5 seconds after event A" requires three reasoning steps on the temporal axis: localization on event A, relational reasoning on "after", and temporal reasoning on "5 seconds". This ensures that q focuses on

a *minimal grounded fact* answerable within a concise visual span.

4. **Cross-modal Synergy**: For any reasonable VideoQA agent $\mathcal{M}$, the answer to q given v must match the answer given the full video:

$$\forall \mathcal{M}, \quad \mathcal{M}(q, v) = \mathcal{M}(Q, V).$$

This enforces that v retains all information necessary to support the same answer as the full input, regardless of model architecture.

This formulation enables precise, interpretable alignment between question, answer, and visual evidence, while ensuring temporal coherence and model-agnostic consistency. Refer to Figure 3 for more demonstrations of the TVS task.

2.2 Integrating TVS into Video-LLM Workflows

Temporal Visual Screening (TVS) can be effectively incorporated into both the training and inference workflows of video-language models, enabling cleaner supervision, stronger alignment, and more interpretable model behavior. Below, we describe two major modes of application.

TVS for Inference-Time Simplification in Video-LLMs At inference, TVS acts as a lightweight pre-processing module. Given a video V and query Q, it identifies a salient segment v and reconstructs a simplified query q (see Section 3). This yields a new input pair (v, q), reducing the temporal and semantic complexity the model must handle. Rather than processing the full video or ambiguous and overgeneralized instructions, the model concentrates on compact, relevant content, facilitating fine-grained temporal reasoning and direct visual grounding. This design aims to improve inference efficiency, robustness, and interpretability.

TVS for Improving Training-Time Visual-Language Alignment Training Video-LLMs often involves complex supervision tuples (V, Q, a), where queries may be abstract or multi-hop. TVS mitigates this by transforming them into more compact triples (v, q, a), where v is a high-utility segment and q is a grounded reformulation of the original query. This decomposition offloads off-target supervision to upstream modules, allowing the model to specialize in two core competencies: temporal understanding and multimodal alignment.

3 Method

3.1 ReSimplifyIt Framework

Drawing inspirations from *dynamic and interactive attention deployment* (Treisman and Gelade, 1980; Wolfe, 1994) and humans' information screening process by dragging the video's progress bar, we propose our multi-agent ReSimplifyIt framework. The framework is composed of three main agentic components: the Launcher, the Validator, and the Viewer, accompanied by two extra helper modules as memory trackers: the Failure History and the Success History. For implementation details, please refer to Section 4. We denote $\{(v_r, q_r) | 1 \leq r \leq R\}$ as the whole TVS process, where v_r, q_r represents the output video clip and question of the rth round, respectively, and R sets the maximum number of rounds. Refer to Algorithm 2 for the complete algorithm of our TVS process performed by our ReSimplifyIt framework, and Figure 2 for a snapshot demonstration.

Initiating a TVS Round: Launcher drafting a video trimming instruction The TVS process unfolds over iterative rounds, akin to how humans drag the progress bar multiple times based solely on empirical surmise before actually viewing the video content. In each round r, a **Launcher** module generates a trimming instruction i_r and a revised query q_r as a trial, where the trimming instruction may only contain *high-level* semantics yielding a declarative goal (e.g. "keep the clip after event X" instead of "clip([5s,10s])"), based solely on the previous query q_{r-1}, without vision access. This trial-based methodology strikes an effective balance between performance and efficiency as the inherent semantic relevance between the query and video input can thereby serve as a prior to enabling plausible instruction generation without video access.

Akin to human landing on undesired segments after dragging the progress bar, we equip the Launcher module with self-correction based adaptive refinement mechanisms to tackle rare failure cases and ensure correctness. Specifically, we maintain two memory tracker modules: **Success History (SH)** and **Failure History (FH)**. A successful trial is added to SH and terminates the round; otherwise, it is recorded in FH and retriggers the Launcher with updated feedback.

We formalize the Launcher's behavior as:

$$(i_r, q_r) = \texttt{Launcher}(q_{r-1}, FH, SH, t)$$

where t denotes the task prompt.

Our feedback-driven formulation achieves iterative self-correction, a key distinction from prior work (Wang et al., 2024b; Shang et al., 2024). It decomposes the task into progressive rounds to enable iterative self-correction and foster structured reasoning, which enhances multi-hop robustness while minimizing correction costs. The lightweight, video-free Launcher further boosts efficiency by performing abductive reasoning on the language query without sacrificing fidelity.

Validating a trial: Validator executing instruction The **Validator module** receives the high-level instruction i_r from the Launcher and determines its success through execution. The module performs two tightly coupled tasks: (i) *assessing the feasibility* of the instruction (i.e., whether it is able to succeed), and (ii) *executing* the instruction to obtain results or feedback. Unlike the Launcher, the Validator *has indirect access to visual semantics* by interacting with the Viewer module while not taking visual inputs or frame captions itself.

It returns a tuple (d_r, m_r), where:

- d_r indicates whether the instruction is deemed `succeeded` (feasible) or `failed` (infeasible),
- m_r contains either the resulting trimmed video (in the form of timestamp ranges) if successfully executed the instruction, or an explanation if failed.

$$(d_r, m_r) = \texttt{Validator}(i_r, v_{r-1}, t)$$

Here, `Validator()` denotes the Validator module, v_{r-1} is the previous video state, and t is the task prompt.

The Validator may also decide to call the Viewer module before returning to Launcher. In this scenario, d_r and m_r stand for this decision symbol and the message to the Viewer, respectively.

Our unified Validator assesses plan viability via direct execution rather than handcrafted rules, embodying a "competence from consequence" philosophy (Brooks, 1991). This integration simplifies control flow and reduces inter-module communication. Crucially, failures are not terminal but provide constructive feedback for the Launcher to refine instructions, transforming execution into a mechanism for both validation and continual adaptation.

Scanning the video: Viewer scanning and localizing Inspired by the Bottom-Up (stimulus-driven) and Top-Down (goal-driven) Activation schema proposed by GS2.0 (Wolfe, 1994), which also guides humans' scrubbing the seek bar when watching videos, we design the **Viewer** module to explicitly incorporate two complementary tasks:

(i) Scanning: summarize the content of a video snippet given a timestamp range;

(ii) Localizing: retrieve a timestamp range given a text summarization of a video snippet.

Following complementarily symmetric task structure, this design of the Viewer module achieves elegant and efficient bidirectional navigation by providing both top-down (content-driven) and bottom-up (time-driven) exploration, making the Viewer module highly flexible.

To support both tasks, we first perform a two-stage keyframe extraction and captioning process: (1) extract I-frames using MPEG-4 compression to capture key visual content, and (2) apply a dynamic frame clustering algorithm to select the final keyframes, which adaptively adjust the number of clusters without supervision. Compared to frame clustering techniques in previous work (Wang et al., 2024c), this approach demonstrates stronger generalizability and robustness. For the algorithm of the keyframe extraction process, please refer to algorithm 1. This pre-processing is denoted as:

$$C = prep(start, end).$$

Scanning executes by summarizing the snippet content via LLM reasoning over C, optionally querying additional frames, and is denoted as:

$$Cap = Scanner(prep(start, end), t).$$

Localizing follows a lightweight three-stage search: (1) locate top-k candidate timestamps, (2) select the best, and (3) expand it into a full range. The LLM may optionally query extra frames for confirmation, ensuring minimal frame access while maintaining accuracy. This process is denoted as:

$$(t_{start}, t_{end}) = Localizer(prep(0, d), q).$$

This lightweight yet effective three-stage design achieves fine-grained temporal grounding with minimal overhead, showcasing the Viewer's adaptability and plug-and-play potential.

Algorithms of the keyframe extraction and localization stages are provided in appendix A.

3.2 TVS-guided inference

Attributed to its endomorphic property, TVS can serve as a plug-and-play adapter process for any VideoQA pipeline, including both inference and

training stages. Given the input question Q and video V, TVS-guided inference is conducted as:

$$response = VideoQA(v, q)$$
$$v, q = TVS(V, Q)$$

where $VideoQA$ can be *any* VideoQA pipeline, including video-LLMs, LLM-assisted pipelines, or any others alternatives, and $TVS()$ stands for the Temporal Visual Screening process.

3.3 TVS-guided training

TVS can also be applied to training stage to purify supervision signal. Given input question Q, input video V, and ground truth answer a, we compute the likelihood during the TVS-guided training as:

$$p\left(\mathbf{X}_{\mathrm{A}} \mid \mathbf{X}_{\mathrm{V}}, \mathbf{X}_{\mathrm{Q}}\right) = \prod_{i=1}^{L} p_\theta\left(\mathbf{X}_{\mathrm{A}}^{[i]} \mid \mathbf{X}_{\mathrm{V}}, \mathbf{X}_{\mathrm{Q}}, \mathbf{X}_{\mathrm{A}}^{[1:i-1]}\right)$$

$$\mathbf{X}_{\mathrm{A}}, \mathbf{X}_{\mathrm{Q}}, \mathbf{X}_{\mathrm{V}} = f_t(a), f_t(q), f_v(v)$$

$$v, q = TVS(V, Q)$$

where f_t, f_v are the text and visual tokenizers.

4 Experiment Settings

4.1 YouCookII-TVS benchmark

As TVS is a new task, few benchmarks is capable of performing the evaluation. While some studies (Lei et al., 2018; Chen et al., 2024b; Lei et al., 2020) have explored grounded VideoQA which may be mistakenly seen as equivalent, they mainly focus on improving visual evidence. See Appendix F for details. The synchronous update of video and query inputs in TVS naturally resolves any potential semantic mismatch, maximally enhancing generalizability and adaptability of our work.

To bridge this gap, we introduce the YouCookII-TVS benchmark, which provides both TVS and standard VideoQA labels. Built upon the YouCookII dataset (Zhou et al., 2018), it comprises 2,754 datapoints split into training, validation, and test sets (roughly 7:2:1). This dual-task benchmark enables unified evaluation of both TVS and VideoQA. Refer to Appendix B for more details.

4.2 Datasets and Benchmarks

We conduct the evaluation from three aspects.

Firstly, we evaluate the quality of the TVS process on **YouCookII-TVS** (Section 4.1). Next, we examine the impact of TVS on downstream VideoQA performance across the following benchmarks: **YouCookII-TVS**, which supports evaluation of both the TVS process and VideoQA performance; **ActivityNet-QA** (Yu et al., 2019), **NExT-OE** (Xiao et al., 2021), **Video-MME** (Fu et al., 2024), **MLVU** (Zhou et al., 2024a), **LVBench**(Wang et al., 2024a), and **EgoSchema** (Mangalam et al., 2023). Lastly, we investigate the role of TVS in video instruction tuning by fine-tuning Video-LLM baselines on **YouCookII-TVS**, and comparing their downstream performance across various benchmarks. For **NExT-QA** (Xiao et al., 2021), we conduct open-ended generation during inference and map predictions to MCQ format using GPT-3.5, ensuring consistent evaluation across baselines and benchmarks (see the Appendix for details).

4.3 Implementation Details

We provide implementation details in Appendix D.

4.4 Baselines

Temporal Visual Screening Considering the lack of baselines of the new TVS task, we adopt two baselines from the training-free temporal localization task which are considered to be robust and generalizable to unseen datasets, to make a side-by-side comparison of the video output alone of TVS. Specifically, we adopt VTG-GPT (Xu et al., 2024b), a proposal-based method which made one of the first attempts to training-free video temporal grounding, and Zheng et al. (2024a)'s work which comprehend candidate proposals based on both static and dynamic matching scores. For the query output of the TVS process, we report open-ended evaluation result of our proposed framework.

TVS-guided inference Integrating TVS as a front-end module into existing VideoQA agents yields a novel VideoQA framework. We adopt several Video-LLMs—Video-ChatGPT (Maaz et al., 2024), Video-LLaVA (Lin et al., 2023), ChatUniVi (Jin et al., 2023), LLaVA-NeXT (Liu et al., 2024b), InternVL3.5 (Wang et al., 2025), Qwen2.5VL (Bai et al., 2025b), Qwen3VL (Bai et al., 2025a), LLaVA-OneVision (Li et al., 2024a)—as representative baselines. We also examine LLM-assisted frameworks such as VideoTree (Wang et al., 2024c) and VideoAgent (Wang et al., 2024b), which, unlike the static encoding approaches of Video-LLMs, dynamically extract video frames based on the textual query.

TVS-guided training Following Section 3.3, we performed video instruction tuning on Video-

Method	Temporal Relational		Timepoint Indexed		Multifaceted Integrative		**Average**	
	mIoU	F1	mIoU	F1	mIoU	F1	mIoU	F1
VTG-GPT (Xu et al., 2024b)	0.17	0.29	0.15	0.26	0.16	0.24	0.16	0.27
Zheng et al. (2024a)	0.11	0.19	0.04	0.08	0.06	0.10	0.07	0.12
ReSimplifyIt (Ours)	**0.23**	**0.37**	**0.98**	**0.99**	**0.47**	**0.64**	**0.56**	**0.67**

(a) Results on video output

Method	Temporal Relational	Timepoint Indexed	Multifaceted Integrative	**Average**
ReSimplifyIt (Ours)	66.8	78.5	72.8	72.7

(b) Results on query rewriting

Table 1: Stage-1 evaluation results on YouCookII-TVS dataset.

Method	**Size**	**ActivityNetQA**		**YC2TVS**		**EgoSchema**		**LVBench**		**MLVU**		**Video-MME**		**NExT-OE**	
		w/tvs	w/o	w/tvs	w/o	w/tvs	w/o	w/tvs	w/o	w/tvs	w/o	w/tvs	w/o	w/tvs	w/o
Video-ChatGPT (Maaz et al., 2024)	7B	**58.5**	50.5	**38.9**	28.91	**29.2**	23.0	**23.5**	22.9	**22.5**	18.6	**31.1**	29.5	**49.8**	41.5
Video-LLaVA (Lin et al., 2023)	7B	**61.2**	52.1	**43.2**	32.4	**43.8**	40.0	**26.5**	23.2	**31.9**	27.5	**43.4**	39.1	**48.0**	43.3
ChatUniVi (Jin et al., 2023)	7B	**63.2**	52.0	**56.5**	47.4	–	–	–	–	–	–	–	–	**34.5**	25.9
LLaVA-NExT (Liu et al., 2024b)	7B	**67.8**	62.5	**58.2**	48.6	**38.6**	38.2	**31.4**	23.8	**34.2**	28.3	**43.8**	40.3	**52.3**	43.7
InternVL3.5 (Wang et al., 2025)	8B	**59.9**	57.2	**51.5**	45.7	**67.2**	61.8	**46.3**	39.6	**46.8**	45.1	**60.4**	56.9	**59.1**	53.9
InternVL3.5 (Wang et al., 2025)	14B	**60.1**	58.9	**52.1**	48.8	**70.6**	67.6	**47.8**	42.9	**46.5**	45.2	**62.2**	61.1	**58.2**	55.6
Qwen2.5-VL (Bai et al., 2025b)	3B	**57.6**	55.8	**47.9**	40.3	**61.8**	57.2	**44.2**	36.3	**45.3**	41.3	**59.0**	57.7	**54.8**	51.6
Qwen2.5-VL (Bai et al., 2025b)	7B	**58.0**	55.2	**48.0**	41.1	**68.2**	66.4	**43.0**	34.8	**43.1**	37.6	**60.0**	58.5	**55.2**	53.4
Qwen3-VL (Bai et al., 2025a)	4B	**59.9**	57.7	**48.0**	45.6	**71.2**	70.2	**43.2**	37.5	**49.0**	42.8	60.5	**60.6**	**61.1**	58.2
Qwen3-VL(Bai et al., 2025a)	32B	**60.9**	60.5	**47.8**	45.2	**72.8**	72.6	–	–	**44.9**	38.7	**64.0**	61.2	**61.4**	58.9
LLaVA-OneVision (Li et al., 2024a)	4B	–	–	–	–	**18.6**	18.2	–	–	**13.5**	5.5	–	–	–	–
LLaVA-OneVision (Li et al., 2024a)	8B	**32.4**	27.5	–	–	**38.4**	32.5	–	–	**14.1**	12.2	51.1	**51.3**	–	–
VideoAgent (Wang et al., 2024b)	-	**61.5**	60.2	**53.9**	38.4	–	–	–	–	–	–	–	–	47.8	**49.6**
VideoTree (Wang et al., 2024c)	-	**63.6**	59.0	**69.4**	57.1	–	–	–	–	–	–	–	–	**61.9**	57.7
GPT-4o (OpenAI et al., 2024a)	-	**75.65**	72.2	**73.84**	63.59	**72.38**	71.17	**46.22**	35.04	**48.92**	44.73	**66.91**	61.63	**62.25**	54.9
GPT-4.1-mini (OpenAI et al., 2024b)	-	**77.07**	75.19	**76.04**	68.74	71.12	**72.02**	**46.35**	34.43	**50.67**	39.90	**62.45**	61.20	**68.93**	57.45
GPT-4-turbo (OpenAI et al., 2024b)	-	**77.1**	74.34	**79.39**	70.61	**69.20**	66.60	–	–	–	–	–	–	**70.35**	61.4

Table 2: Evaluation results of TVS-guided inference. **Bold** values indicate the better-performing result for each baseline, comparing the ***with TVS*** *v.s.* ***without TVS*** configurations on each benchmark.

ChatGPT (Maaz et al., 2024), Video-LLaVA (Lin et al., 2023), and ChatUniVi (Jin et al., 2023) on YouCookII-TVS. We kept the LLM backbone frozen, and only tuned their multimodal projectors.

5 Evaluation Results

We highlight the importance of the TVS task, as it addresses a core bottleneck in VideoQA by enabling models to screen for semantically aligned moments in long videos by abstracting task-level semantics and guiding modality-aware information filtering, which in turn enhances multi-modal alignment. In this section, we reveal its importance through: (1) using ground-truth TVS significantly boosts performance (e.g., +7.3% relative gain in Table 3), showing its potential as a key supervision signal; (2) current methods fail to solve TVS effectively (Table 1), calling for dedicated solutions. These motivate our design of ReSimplifyIt as a plug-and-play approach to this essential task.

Temporal Visual Screening Our ReSimplifyIt framework outperform both baselines significantly, on every metric and subset of our YouCookII-TVS benchmark. On the average performance of the whole test set of YouCookII-TVS, ReSimplifyIt achieved an mIoU score higher than 300% of the baselines' performance. For query reshaping, our proposed framework also achieved notably good performance. Refer to Table 1 and Table 5 for results. We provide the prompt in Appendix E.

Ablation Study of ReSimplifyIt Framework We conduct ablation study on ReSimplifyIt framework to evaluate the effectiveness its design. More details are provided in Appendix C.

TVS guided inference As shown in Table 2 and Table 6, nearly all baselines - ranging from open source models of different sizes and architectures to proprietary models - saw solid absolute performance gain, suggesting that Video-LLMs largely suffer from superfluous cognition noise. The **consistent performance gains across all benchmarks**

Method		ActivityNetQA		YouCookII-TVS		NExT-QA		NExT-OE	
		w/tvs	w/o	w/tvs	w/o	w/tvs	w/o	w/tvs	w/o
Video-ChatGPT	w/o tvs	57.4	49.8	45.1	35.6	48.6	41.3	46.2	40.0
	w/ tvs (gt)	62.7	54.4	49.8	38.9	51.0	45.7	49.2	43.5
	w/ tvs (pred)	60.0	51.3	48.2	37.1	49.6	44.3	48.7	41.9
Video-LLaVA	w/o tvs	53.9	45.1	46.3	35.9	50.2	47.8	47.0	39.1
	w/ tvs (gt)	57.0	48.9	52.6	41.6	55.7	50.8	52.1	44.9
	w/ tvs (pred)	54.2	46.9	48.1	37.4	51.0	49.3	50.7	41.8
ChatUniVi	w/o tvs	63.2	50.0	62.4	57.4	–	–	29.0	22.5
	w/ tvs (gt)	65.6	54.4	67.8	63.8	–	–	34.3	26.0
	w/ tvs (pred)	64.9	52.6	62.8	61.6	–	–	32.1	24.0

Table 3: Evaluation results of TVS-guided training by downstream inference. On the rows, *w/o tvs*, *w/tvs (gt)*, and *w/tvs (pred)* indicates training with vanilla YouCookII-TVS dataset, training with ground truth TVS labels, and training with the predicted TVS results, respectively; on the columns, *w/o tvs* and *w/tvs* indicates the inference mode.

underscore the universality of this general reasoning load optimization problem across diverse tasks and scenarios. In contrast, both LLM-assisted reasoning frameworks saw smaller gain. We hypothesize that the design and implementation of these frameworks have intrinsically incorporated the TVS process, by leveraging strong reasoning ability of external LLMs or multi-turn interaction.

TVS guided training All checkpoints trained on ground truth TVS labels achieved stable performance gain over their counterparts trained on the vanilla YouCookII-TVS benchmark, which performance gain of the checkpoints trained on the predicted TVS output of ReSimplifyIt framework were weaker. We argue that TVS-guided training benefits Video-LLMs better, particularly when faced with untrimmed videos and complex text instructions requiring multi-hop reasoning. Refer to Table 3 for further details.

6 Efficiency Analysis

We now quantify the practical overhead of the TVS screening loop against the savings it yields downstream. To remain comparable across hardware, API endpoints, and network conditions, we report *hardware-agnostic proxies*—LLM/tool/caption call counts, output tokens, duration reduction, and screening success rate—together with closed-loop metrics under realistic frame budgets.

Cost drivers and screening effectiveness. For each sample we record the number of external LLM turns, tool invocations, total captions, and output tokens, together with the duration ratios $\mathrm{DurAll} = \mathbb{E}[|v|/|V|]$ and $\mathrm{DurScrn}$ (restricted to successfully screened samples), and $\mathrm{Screen}\%$, the fraction of samples for which $\mathcal{F}_{\mathrm{TVS}}(V, Q) \neq (V, Q)$. See Appendix G (Tables 8, 9, and 10) for per-dataset values and the full per-source caption breakdown. ReSimplifyIt-simple averages only $\sim$22 captions, 3–10 LLM calls, and 300–1,500 output tokens per sample, succeeding on 94.4%–100% of samples across all seven benchmarks. The full multi-agent ReSimplifyIt uses more resources due to its iterative refinement, with roughly half of its captions pre-loaded in agent prompts and the rest fetched via the Viewer's tools. Both variants reduce videos to a small fraction of their original duration on successfully screened samples—down to 1.9% (simple) and 4.8% (full) on LVBench—with compression strongest on the longest benchmarks.

Closing the loop under fixed frame budgets. A naive comparison via the count of downstream frames is misleading: a fixed budget of K frames sampled uniformly over a long video is extremely sparse and may under-cover the evidence, while the same K frames concentrated on the screened segment yield much denser coverage. To make this precise, we define

$$\mathrm{DensAmp} = \mathbb{E}\left[\frac{1}{\mathrm{DurRatio}}\right],$$
$$\mathrm{EquivFr}(K) = K \cdot \mathrm{DensAmp},$$
$$\mathrm{CostRatio}(K) = \frac{K + \mathrm{TotalCap}}{\mathrm{EquivFr}(K)}.$$

Here $\mathrm{DensAmp}$ captures how much denser downstream sampling becomes after screening: K post-

Dataset	DensAmp	OT/1%	CostRatio(K)			
			K=8	K=16	K=32	K=100
ActivityNet-QA	25.0×	3.8	0.15	0.09	0.07	0.05
YouCookII-TVS	29.7×	3.9	0.13	0.08	0.06	0.04
NExT-OE	20.7×	5.2	0.19	0.12	0.08	0.06
EgoSchema	6.3×	6.0	0.59	0.38	0.27	0.19
LVBench	320.7×	15.1	0.01	0.01	0.01	0.00
MLVU	70.6×	10.8	0.06	0.04	0.02	0.02
Video-MME	51.2×	8.8	0.08	0.05	0.03	0.02

(a) ReSimplifyIt-simple

Dataset	DensAmp	OT/1%	CostRatio(K)			
			K=8	K=16	K=32	K=100
ActivityNet-QA	65.2×	39.6	0.18	0.10	0.06	0.03
YouCookII-TVS	14.7×	25.5	0.70	0.38	0.23	0.12
NExT-OE	12.3×	34.9	0.83	0.45	0.27	0.14
EgoSchema	25.8×	77.3	0.67	0.36	0.20	0.09
LVBench	852.8×	38.5	0.02	0.01	0.01	0.00
MLVU	308.5×	48.9	0.06	0.03	0.02	0.01
Video-MME	114.5×	43.2	0.12	0.07	0.04	0.02

(b) ReSimplifyIt (full).

Table 4: Density amplification (DensAmp), output-token cost per 1% of video-length reduction (OT/1%), and visual CostRatio under four downstream frame budgets K. Statistics are averaged over successfully screened samples.

screening frames are equivalent to $\mathrm{EquivFr}(K)$ frames uniformly sampled over the original video. $\mathrm{CostRatio}(K)$ then expresses the total visual sampling cost (K + TotalCap) as a fraction of the dense-sampling cost needed to reach the same evidence density. A lower CostRatio therefore means TVS achieves the same effective evidence density at a smaller fraction of the dense-sampling cost: CostRatio=0.1 means the no-TVS baseline would incur 10× the frame-sampling cost *to reach the same evidence density*. We also report the output-token cost of each 1% reduction in video duration,

$$\mathrm{OutTok/1\%Red} = \frac{\mathbb{E}[\#\mathrm{OutTok}]}{(1 - \mathbb{E}[\mathrm{DurScrn}]) \cdot 100}.$$

Density amplification and end-to-end cost. Table 4 quantifies how the screening overhead is repaid by concentrated downstream sampling. ReSimplifyIt-simple achieves 6.3×–320.7× density amplification; on LVBench, 8 frames sampled from the screened segment match the temporal density of 2,566 frames sampled uniformly over the full video. The full ReSimplifyIt reaches 12.3×–852.8× on successfully screened samples, with output-token costs of 3.8–15.1 tokens per 1% duration reduction (simple) and 25.5–77.3 (full). At K=8, the simple variant's CostRatio ≤ 0.19 on 6 of 7 datasets (LVBench reaching 0.01, a ∼100× reduction); the full variant reports CostRatio $\in [0.02, 0.18]$ on the long-video benchmarks where it is most effective. As K grows to $16, 32, 100$, CostRatio decreases monotonically across all datasets, indicating that TVS becomes *increasingly* cost-effective as downstream Video-LLMs adopt denser frame sampling. Combined with the accuracy gains of the previous section, TVS is a practically deployable front-end whose three instantiations (Appendix C) further offer an explicit performance–cost spectrum.

7 Related Work

We provide more details in appendix F.

Video-LLMs for VideoQA Video-LLMs have spurred a wave of models aimed at enhancing video understanding (Lin et al., 2023; Ma et al., 2024; Li et al., 2024b, 2023b; Liu et al., 2024b; Xu et al., 2024a), while the sparsity and query-invariant nature of their encoding limits efficacy in capturing fine-grained spatial-temporal details.

LLM-assisted Agentic Reasoning for VideoQA Some other work proposing robust VideoQA baselines opt to explore pure-text LLM assisted frameworks or multi-agent systems for VideoQA (Wang et al., 2024c; Shang et al., 2024; Wang et al., 2024b). These methods adopt LLM-based methods to serve as a scheduler, which implicitly fullfills the TVS objective to a significant extent.

8 Conclusion

We introduce Temporal Visual Screening (TVS), a cognitively inspired task that reconstructs each VideoQA instance (V, Q) into an answer-preserving compact pair (v, q) to purify cognition load. We realize TVS via ReSimplifyIt, a plug-and-play, model-agnostic multi-agent framework, and release YouCookII-TVS to benchmark screening-centric reasoning. Across models and datasets, TVS consistently strengthens both training-time alignment and inference-time robustness, yielding notable accuracy gains and exposing bottlenecks via compact, auditable rationales, positioning information screening as a key direction for advancing video-language understanding.

Acknowledgments

This research is based upon work supported by U.S. DARPA ECOLE Program No. #HR00112390060. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either

expressed or implied, of DARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

Limitations

While applying the initiative of information screening brings considerable performance boost, there are still limitations and room for improvements of our work. Firstly, as the name suggests, temporal visual screening is only applied on the temporal axis and is unable to filter redundant spatial visual information. We recognize spatial visual screening as the more difficult counterpart of our task. Secondly, although our proposed framework's nature of being plug-and-play and adaptable to any VideoQA agent is considered as a strength, it's end-to-end counterpart, which may mainly relate to Video-LLMs' new visual encoder capable of inter-frame reasoning and query-adaptive frame sampling, is left unexplored. In fact, we believe that both extending plug-and-play external modules and improving end-to-end internal model structure can potentially pave the way to information screening.

References

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025a. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025b. Qwen2.5-vl technical report. *Preprint*, arXiv:2502.13923.

Gedas Bertasius, Heng Wang, and Lorenzo Torresani. 2021. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning (ICML)*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.

Rodney A Brooks. 1991. Intelligence without representation. *Artificial intelligence*, 47(1-3):139–159.

Guo Chen, Yicheng Liu, Yifei Huang, Yuping He, Baoqi Pei, Jilan Xu, Yali Wang, Tong Lu, and Limin Wang. 2024a. Cg-bench: Clue-grounded question answering benchmark for long video understanding. *Preprint*, arXiv:2412.12075.

Guo Chen, Yin-Dong Zheng, Jiahao Wang, Jilan Xu, Yifei Huang, Junting Pan, Yi Wang, Yali Wang, Yu Qiao, Tong Lu, and Limin Wang. 2023. Videollm: Modeling video sequence with large language models. *Preprint*, arXiv:2305.13292.

Qirui Chen, Shangzhe Di, and Weidi Xie. 2024b. Grounded multi-hop videoqa in long-form egocentric videos. *Preprint*, arXiv:2408.14469.

Shaoxiang Chen and Yu-Gang Jiang. 2019. Semantic proposal for activity localization in videos via sentence query. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press.

Chaoyou Fu, Yuhan Dai, Yondong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, and 1 others. 2024. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. *arXiv preprint arXiv:2405.21075*.

Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. 2017. Tall: Temporal activity localization via language query. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5277–5285.

Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. 2017. Localizing moments in video with natural language. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5804–5813.

Shaul Hochstein and Merav Ahissar. 2002. View from the top: Hierarchies and reverse hierarchies in the visual system. *Neuron*, 36(5):791–804.

Peng Jin, Ryuichi Takanobu, Caiwan Zhang, Xiaochun Cao, and Li Yuan. 2023. Chat-univi: Unified visual representation empowers large language models with image and video understanding. *arXiv preprint arXiv:2311.08046*.

Didier Le Gall. 1991. Mpeg: a video compression standard for multimedia applications. *Commun. ACM*, 34(4):46–58.

Jie Lei, Licheng Yu, Mohit Bansal, and Tamara Berg. 2018. TVQA: Localized, compositional video question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1369–1379, Brussels, Belgium. Association for Computational Linguistics.

Jie Lei, Licheng Yu, Tamara Berg, and Mohit Bansal. 2020. Tvqa+: Spatio-temporal grounding for video question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8211–8225, Online. Association for Computational Linguistics.

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

Kunchang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. 2023a. Videochat: Chat-centric video understanding. *arXiv preprint arXiv:2305.06355*.

Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Lou, Limin Wang, and Yu Qiao. 2024b. Mvbench: A comprehensive multi-modal video understanding benchmark. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22195–22206.

Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, Limin Wang, and Yu Qiao. 2024c. Mvbench: A comprehensive multi-modal video understanding benchmark. *Preprint*, arXiv:2311.17005.

Yanwei Li, Chengyao Wang, and Jiaya Jia. 2023b. Llama-vid: An image is worth 2 tokens in large language models. In *European Conference on Computer Vision*.

Bin Lin, Bin Zhu, Yang Ye, Munan Ning, Peng Jin, and Li Yuan. 2023. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*.

Yijie Lin, Jie Zhang, Zhenyu Huang, Jia Liu, Zujie Wen, and Xi Peng. 2024. Multi-granularity correspondence learning from long-term noisy videos. In *Proceedings of the International Conference on Learning Representations*.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024a. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26296–26306.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. 2024b. Llava-next: Improved reasoning, ocr, and world knowledge.

Ruyang Liu, Chen Li, Yixiao Ge, Thomas H. Li, Ying Shan, and Ge Li. 2024c. Bt-adapter: Video conversation is feasible without video instruction tuning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13658–13667.

Fan Ma, Xiaojie Jin, Heng Wang, Yuchen Xian, Jiashi Feng, and Yi Yang. 2024. Vista-llama: Reducing hallucination in video language models via equal distance to visual tokens. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13151–13160.

Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Khan. 2024. Video-ChatGPT: Towards detailed video understanding via large vision and language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12585–12602, Bangkok, Thailand. Association for Computational Linguistics.

Karttikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. 2023. Egoschema: A diagnostic benchmark for very long-form video language understanding. *Preprint*, arXiv:2308.09126.

Richard E. Mayer and Roxana Moreno. 2003. Nine ways to reduce cognitive load in multimedia learning. *Educational Psychologist*, 38(1):43–52.

Allen Newell and Herbert A. Simon. 1972. *Human Problem Solving*. Prentice-Hall, Englewood Cliffs, NJ.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, and 401 others. 2024a. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024b. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Fred Paas, Alexander Renkl, and John Sweller. 2003. Cognitive load theory and instructional design: Recent developments. *Educational Psychologist*, 38(1):1–4.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. *Preprint*, arXiv:2103.00020.

Chuyi Shang, Amos You, Sanjay Subramanian, Trevor Darrell, and Roei Herzig. 2024. TraveLER: A modular multi-LMM agent framework for video question-answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9740–9766, Miami, Florida, USA. Association for Computational Linguistics.

John Sweller. 1988. Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12(2):257–285.

Anne M. Treisman and Garry Gelade. 1980. A feature-integration theory of attention. *Cognitive Psychology*, 12(1):97–136.

Weihan Wang, Zehai He, Wenyi Hong, Yean Cheng, Xiaohan Zhang, Ji Qi, Shiyu Huang, Bin Xu, Yuxiao Dong, Ming Ding, and Jie Tang. 2024a. Lvbench: An extreme long video understanding benchmark. *Preprint*, arXiv:2406.08035.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, and 56 others. 2025. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *Preprint*, arXiv:2508.18265.

Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. 2024b. Videoagent: Long-form video understanding with large language model as agent. *European Conference on Computer Vision (ECCV)*.

Ziyang Wang, Shoubin Yu, Elias Stengel-Eskin, Jaehong Yoon, Feng Cheng, Gedas Bertasius, and Mohit Bansal. 2024c. Videotree: Adaptive tree-based video representation for llm reasoning on long videos. *Preprint*, arXiv:2405.19209.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jeremy M. Wolfe. 1994. Guided search 2.0: A revised model of visual search. *Psychonomic Bulletin & Review*, 1(2):202–238.

Penghao Wu and Saining Xie. 2024. V?: Guided visual search as a core mechanism in multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13084–13094.

Junbin Xiao, Xindi Shang, Angela Yao, and Tat-Seng Chua. 2021. Next-qa: Next phase of question-answering to explaining temporal actions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9777–9786.

Junbin Xiao, Angela Yao, Yicong Li, and Tat-Seng Chua. 2024. Can i trust your answer? visually grounded video question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13204–13214.

Huijuan Xu, Kun He, Bryan A. Plummer, Leonid Sigal, Stan Sclaroff, and Kate Saenko. 2019. Multilevel language and vision integration for text-to-clip retrieval. In *AAAI*.

Lin Xu, Yilin Zhao, Daquan Zhou, Zhijie Lin, See Kiong Ng, and Jiashi Feng. 2024a. Pllava : Parameter-free llava extension from images to videos for video dense captioning. *ArXiv*, abs/2404.16994.

Yifang Xu, Yunzhuo Sun, Zien Xie, Benxiang Zhai, and Sidan Du. 2024b. Vtg-gpt: Tuning-free zero-shot video temporal grounding with gpt. *Applied Sciences*, 14(5):1894.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Zhou Yu, Dejing Xu, Jun Yu, Ting Yu, Zhou Zhao, Yueting Zhuang, and Dacheng Tao. 2019. Activitynet-qa: A dataset for understanding complex web videos via question answering. In *AAAI*, pages 9127–9134.

Yitian Yuan, Tao Mei, and Wenwu Zhu. 2019. To find where you talk: temporal sentence localization in video with attention based location regression. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press.

Jeffrey M. Zacks, Nicole K. Speer, Khena M. Swallow, Todd S. Braver, and Jeremy R. Reynolds. 2007. Event perception: A mind/brain perspective. *Psychological Bulletin*, 133(2):273–293.

Hao Zhang, Aixin Sun, Wei Jing, and Joey Tianyi Zhou. 2020. Span-based localizing network for natural language video localization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6543–6554, Online. Association for Computational Linguistics.

Yuji Zhang, Sha Li, Cheng Qian, Jiateng Liu, Pengfei Yu, Chi Han, Yi R Fung, Kathleen McKeown, Chengxiang Zhai, Manling Li, and 1 others. 2025. The law of knowledge overshadowing: Towards understanding, predicting, and preventing llm hallucination. *arXiv preprint arXiv:2502.16143*.

Minghang Zheng, Xinhao Cai, Qingchao Chen, Yuxin Peng, and Yang Liu. 2024a. Training-free video temporal grounding using large-scale pre-trained models. *Preprint*, arXiv:2408.16219.

Minghang Zheng, Xinhao Cai, Qingchao Chen, Yuxin Peng, and Yang Liu. 2024b. Training-free video temporal grounding using large-scale pre-trained models. *Preprint*, arXiv:2408.16219.

Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Shitao Xiao, Xi Yang, Yongping Xiong, Bo Zhang, Tiejun Huang, and Zheng Liu. 2024a. Mlvu: A comprehensive benchmark for multi-task long video understanding. *arXiv preprint arXiv:2406.04264*.

Luowei Zhou, Chenliang Xu, and Jason J Corso. 2018. Towards automatic learning of procedures from web instructional videos. In *AAAI Conference on Artificial Intelligence*, pages 7590–7598.

Xingyi Zhou, Anurag Arnab, Shyamal Buch, Shen Yan, Austin Myers, Xuehan Xiong, Arsha Nagrani, and Cordelia Schmid. 2024b. Streaming dense video captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18243–18252.

A Viewer Implementation Details

Keyframe Extraction and Captioning. We utilize a two-stage keyframe extraction process combined with frame captioning. In the first stage, we implement the MPEG-4 compression technique (Le Gall, 1991) to extract all I-frames as candidate keyframes. I-frames typically contain rich visual content and clarity, or represent scene transitions.

In the second stage, we apply a modified Isodata clustering algorithm to the visual features of these I-frames, selecting cluster centers as final keyframes. This algorithm adaptively determines the number of clusters, unlike KNN-based clustering (Wang et al., 2024c; Zhou et al., 2024b), which requires a pre-defined number of clusters or external supervision. Our approach ensures generalizability and robustness for diverse video types.

After clustering, we utilize an off-the-shelf vision-language model (VLM) to generate frame captions for the selected keyframes. The overall process is denoted by:

$$C = prep(start, end)$$

where C is the set of frame captions between the given timestamps.

Scanning. Given a timestamp range $(start, end)$, the Viewer calls an external LLM with C to produce an overview caption. It may query additional frames via a captioning tool to refine its summary, mimicking how users drag to specific timestamps for clarification. The process is:

$$Cap = Scanner(prep(start, end), t)$$

where t is the task prompt.

Localizing. To identify the timestamp range matching a textual description, we propose a lightweight three-stage search process:

Stage 1: We feed an external LLM the set of keyframe captions to let it acquire an overall capture of the video content. At the same time, we instruct the LLM to output five most possible timestamps that depicts the text query. The LLM is able to call frame caption tool to acquire extra captions at arbitrary timestamps, before it's confident enough to output the answer. Formally:

$$P = stage_1(prep(0, d), e, t)$$

where e are extra captions and t is the prompt. P gives the resulting list of five timestamps which depicts the language query the best.

Stage 2: We initialize the conversation and feed it the frame captions at these five timestamps, while instructing it to pick the one timestamp that best depicts the language query, out of the five candidates. Also, the LLM is able to call frame caption tool to acquire extra captions at arbitrary timestamps before it's confident enough to output the answer.

$$t_{best} = stage_2(P, e, t)$$

where P is the output of the previous step, and t is the task prompt.

Stage 3: Last, we initialize the conversation again, and instruct the external LLM to expand the single timestamp t_{best} from the last step to a timestamp range. The frame caption at t_{best} is provided, and the LLM is still able to acquire more captions by tool calling to confirm the boundary. Considering the difficulty in dealing with dynamic transitions of events, as explored by some previous work (Zheng et al., 2024b), we simply apply a hard value of 5 seconds on the output. Formally:

$$(t'_{start}, t'_{end}) = stage_3(t_{best}, e, t)$$
$$(t_{start}, t_{end}) = (t'_{start} - 5, t'_{end} + 5)$$

This three-stage process balances precision and efficiency by minimizing frame access while ensuring robust temporal grounding. For visual clarity of the Isodata clustering, refer to Algorithm 1 for details.

B Details on the YouCookII-TVS benchmark

We constructed a synthetic question-answering dataset named **YouCookII-TVS**, based on the YouCookII dataset, to support fine-grained temporal and semantic understanding of cooking videos.

Method	Temporal Relational				Timepoint Indexed				Multifaceted Integrative				**Average**			
	mIoU	Pre.	Cov.	F1	mIoU	Pre.	Cov.	F1	mIoU	Pre.	Cov.	F1	mIoU	Pre.	Cov.	F1
VTG-GPT (Xu et al., 2024b)	0.17	0.26	0.32	0.29	0.15	0.22	0.30	0.26	0.16	0.23	0.31	0.24	0.16	0.27	0.31	0.27
Zheng et al. (2024a)	0.11	0.18	0.21	0.19	0.04	0.08	0.09	0.08	0.06	0.10	0.11	0.10	0.07	0.12	0.13	0.12
ReSimplifyIt (Ours)	**0.23**	**0.36**	**0.39**	**0.37**	**0.98**	**1.0**	**0.98**	**0.99**	**0.47**	**0.60**	**0.69**	**0.64**	**0.56**	**0.65**	**0.69**	**0.67**

(a) more results on stage 1 video output

Table 5: More results on Stage-1 evaluation on YouCookII-TVS dataset. 'Pre.' and 'Cov.' stands for 'precision' and 'coverage'.

B.1 Source Data Preparation

The original YouCookII dataset (Zhou et al., 2018) contains temporally annotated instructional videos. Each annotation includes a segment $[s_i, e_i]$ representing start and end times (in seconds), along with a natural language description of the cooking step.

To ensure video consistency and avoid duplications, we have verified that each video clip name is unique across the dataset source.

B.2 Annotation Grouping via Temporal Connectivity

We define a temporal connectivity criterion to group sequential cooking steps into higher-level event triplets. Given two segments $[s_1, e_1]$ and $[s_2, e_2]$, we define their overlap ratio as:

$$\text{overlap\_ratio} = \frac{|\min(e_1, e_2) - \max(s_1, s_2)|}{\max(e_1, e_2) - \min(s_1, s_2)}$$

Two segments are considered *connectable* if:

$$s_2 > s_1, \quad e_2 > e_1, \quad \text{and} \quad \text{overlap\_ratio} \leq \theta$$

We set $\theta = 0.1$ in our experiments. Using this rule, we perform a greedy grouping of annotations into connected segments, and extract all valid length-3 subsequences (triplets) from each group.

B.3 Triplet-Based Question Generation

Each triplet $T = \{t_1, t_2, t_3\}$ consists of three temporally ordered steps. For each T, we generate nine different types of question-answer pairs by instantiating predefined templates. The question types are categorized into three groups:

Temporal Relational Reasoning (TRR)

- `trr1`: What is the cooking step after of $[description]$?
- `trr2`: What is the cooking step before $[description]$?
- `trr3`: What is the cooking step between $[description]$ and $[description]$?

Timepoint Indexed Reasoning (TIR)

- `tir1`: What is the step between timestamps s_2 and e_2?
- `tir2`: What is the step between frame indices $f_{s_2} = s_2 \cdot r$ and $f_{e_2} = e_2 \cdot r$?
- `tir3`: What step appears within $f_d = (e_2 - s_2) \cdot r$ frames after s_2 seconds?

Here, r denotes the video frame rate.

Multifaceted Integrative Reasoning (MIR)

- `mir1`: What is the first step after timestamp s_2?
- `mir2`: What is the last step before timestamp e_2?
- `mir3`: Within s_1 and e_3, what is (are) the cooking step(s) apart from $[description]$ and $[description]$?

Template instantiation is performed by replacing placeholders with actual sentences and timestamps (framestamps) from the triplet.

B.4 Data Structuring and Metadata

Each generated data point is stored with the following fields:

- `vid_name`, `vid_fname`: Video ID and filename.
- `vid_duration`, `vid_frame_rate`: Metadata from video parsing.
- `type`: One of the nine QA types.
- `question`: Instantiated natural language query.
- `answer`: Corresponding ground-truth step description, serving as the ground-truth label for the VideoQA task.

Method	ActivityNetQA	
	w/tvs	w/o
Video-ChatGPT (Maaz et al., 2024)	58.5	50.5
Video-LLaVA (Lin et al., 2023)	61.2	52.1
ChatUniVi (Jin et al., 2023)	63.2	52.0
LLaVA-NExT (Liu et al., 2024b)	67.8	62.5
VideoAgent (Wang et al., 2024b)	61.5	60.2
VideoTree (Wang et al., 2024c)	59.0	63.6

Method	YouCookII-TVS							
	TRR.		TIR.		MIR.		Avg.	
	w/tvs	w/o	w/tvs	w/o	w/tvs	w/o	w/tvs	w/o
Video-ChatGPT (Maaz et al., 2024)	31.7	30.5	45.2	26.5	39.8	29.73	38.9	28.91
Video-LLaVA (Lin et al., 2023)	39.3	37.4	46.8	28.2	43.5	31.6	43.2	32.4
ChatUniVi (Jin et al., 2023)	54.4	39.8	57.4	52.8	58.7	49.6	56.5	47.4
LLaVA-NExT (Liu et al., 2024b)	46.5	44.1	65.9	33.4	62.2	38.3	58.2	48.6
VideoAgent (Wang et al., 2024b)	30.2	48.7	46.1	55.2	38.9	57.8	38.4	53.9
VideoTree (Wang et al., 2024c)	59.3	64.0	59.2	72.2	52.8	72.0	57.1	69.4

Method	NExT-QA							
	Tem.		Cau.		Des.		Avg.	
	w/tvs	w/o	w/tvs	w/o	w/tvs	w/o	w/tvs	w/o
Video-ChatGPT (Maaz et al., 2024)	45.5	23.7	45.5	56.7	45.6	43.0	45.5	44.2
Video-LLaVA (Lin et al., 2023)	42.8	42.8	50.4	48.9	55.1	44.5	52.2	46.3
ChatUniVi (Jin et al., 2023)	-	-	-	-	-	-	5	28
LLaVA-NExT (Liu et al., 2024b)	57.5	52.3	62.0	59.0	61.4	56.5	60.5	56.6
VideoAgent (Wang et al., 2024b)	49.2	47.3	43.6	41.9	51.7	51.1	47.0	45.0
VideoTree (Wang et al., 2024c)	62.4	59.9	57.7	66.1	66.8	72.3	60.0	65.2

Method	NExT-OE							
	Tem.		Des.		Cau.		Avg.	
	w/tvs	w/o	w/tvs	w/o	w/tvs	w/o	w/tvs	w/o
Video-ChatGPT (Maaz et al., 2024)	49.6	40.8	51.2	43.2	46.9	38.6	49.8	41.5
Video-LLaVA (Lin et al., 2023)	37.5	37.5	53.2	46.7	47.6	43.4	48.0	43.3
ChatUniVi (Jin et al., 2023)	30.8	30.8	36.9	23.1	34.0	26.4	34.5	25.9
LLaVA-NExT (Liu et al., 2024b)	41.4	39.11	59.7	46.0	50.1	44.6	52.3	43.7
VideoAgent (Wang et al., 2024b)	44.1	53.4	48.8	46.0	50.4	52.6	47.8	49.6
VideoTree (Wang et al., 2024c)	52.1	61.1	58.0	60.7	64.4	65.6	57.7	61.9

Table 6: Evaluation results of TVS-guided inference on four benchmark: ActivityNetQA, YouCookII-TVS, NExT-QA, and NExT-OE, with the sub-categories presented in each benchmark.

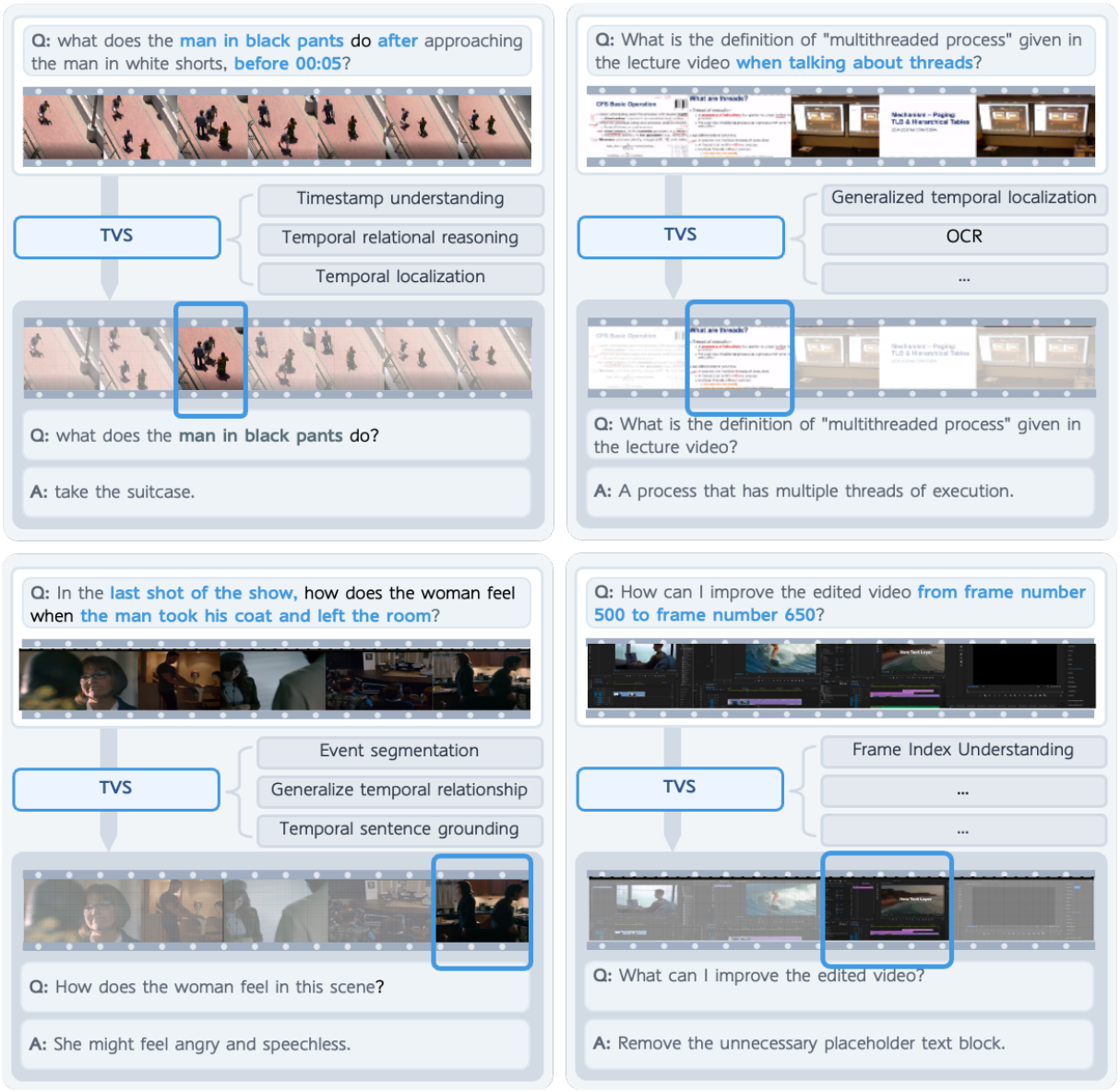

Figure 3: More examples of the TVS task.

- `gt_timestamp`: Temporal segment(s) serving as the ground-truth label for our TVS task on video trimming.
- `gt_rewritten_query`: Natural language query serving as the ground-truth label for our TVS task on query re-writing.

We generated a total of $N = 2754$ QA samples, covering all types evenly.

B.5 Dataset Splitting

To support evaluation, we partition the dataset into training, validation, and test splits. For each question type, we allocated:

$$\text{train: } 1926, \quad \text{val: } 270, \quad \text{test: } 558$$

This roughly follows 7:1:2.

More details about the dataset statistics can be found in Figure 4.

C Ablation Study

C.1 Experiment Settings

We first ablate the modular design—responsible for facilitating structured reasoning—by proposing the **ReSimplifyIt-simple** variant. In this setting, all underlying interfaces for handling textual and visual inputs, such as ISODATA-clustering (Algorithm 1) and frame-caption querying, are preserved. However, the entire reasoning pipeline is collapsed into a single, universal agent. Specifically, we initialize an external agentic LLM with task descriptions and operational instructions, and expose the aforementioned interfaces either through conver-

Algorithm 1 *ISODATA Clustering*

Require: Data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, Candidate frames $\mathbf{F} \in \mathbb{R}^{n \times c \times w \times h}$, frame feature extractor $Enc()$, initial cluster count k, max iterations T, minimum intra-cluster similarity θ_{split}, maximum inter-cluster similarity θ_{merge}, max clusters k_{max}, min clusters k_{min}, max center shift δ_{max}, min elements per cluster n_{min}

Ensure: Cluster assignments $\mathbf{C}$, Final cluster centers $\mathbf{M}$

$\mathbf{X} \leftarrow \text{Enc}(\mathbf{F})$
$\mathbf{X}_{norm} \leftarrow \text{normalize}(\mathbf{X}, \ell_2)$
$\mathbf{M} \leftarrow \text{random\_sample}(\mathbf{X}, k)$
$t \leftarrow 0$
repeat
    Compute cosine similarity matrix $\mathbf{S} = \mathbf{X}_{norm}\mathbf{M}_{norm}^T$
    $\mathbf{C} \leftarrow \arg\max(\mathbf{S}, \text{axis} = 1)$ ▷ Assign points to nearest clusters
    for each cluster i **do**
        Update center: $\mathbf{m}_i \leftarrow \text{mean}(\mathbf{X}[\mathbf{C} == i])$
    end for
    if any cluster size $< n_{min}$ **then**
        Merge smallest cluster with nearest neighbor ▷ Minimum elements enforcement
    end if
    for each cluster i **do**
        if intra-cluster similarity$(\mathbf{X}[\mathbf{C} == i], \mathbf{m}_i) < \theta_{split}$ **and** $k < k_{max}$ **then**
            Split cluster i into two new clusters ▷ Splitting phase
            $k \leftarrow k + 1$
        end if
    end for
    for all cluster pairs (i, j) **do**
        if inter-cluster similarity$(\mathbf{m}_i, \mathbf{m}_j) > \theta_{merge}$ **and** $k > k_{min}$ **then**
            Merge clusters i and j ▷ Merging phase
            $k \leftarrow k - 1$
        end if
    end for
    Compute center shifts $\Delta\mathbf{M} = \|\mathbf{M}_{new} - \mathbf{M}_{old}\|$
    $t \leftarrow t + 1$
until $t \geq T$ **or** $\max(\Delta\mathbf{M}) < \delta_{max}$
return $\mathbf{C}, \mathbf{M}$

Algorithm 2 *ReSimplifyIt*

Require: Video v, question q
Ensure: Screened video V', screened question q'

$V\_copy, q\_copy \leftarrow V, q$
$success\_history, failure\_history \leftarrow$ `SuccessHistory()`, `FailureHistory()`
while true **do**
  $launcher, validator, viewer \leftarrow$ `Launcher()`, `Validator()`, `Viewer()`
  $decision, q', trimming\_instruction \leftarrow$ `launcher`$(q\_copy, success\_history, failure\_history)$
  if $decision == "proceed"$ **then**
    $judgement, request, result, reason \leftarrow$ `validator`$(q\_copy, q', trimming\_instruction)$
    while $judgement == "view"$ **do**
      $response \leftarrow$ `viewer`$(V', request)$
      `validator.read_response`$(response)$
    end while
    if $judgement == "succeeded"$ **then**
      $V' \leftarrow result$
      $success\_history.append([q\_copy, q', trimming\_instruction])$
      $failure\_history.empty()$
      $V\_copy, q\_copy \leftarrow V', q'$
    else
      $failure\_history.append([q\_copy, q', trimming\_instruction])$
    end if
  else
    return V_copy, q_copy
  end if
end while

sational context or tool invocation. This unified agent is then responsible for all reasoning procedures—functionally covering the roles of the Launcher, Validator, and Viewer modules, as well as memory tracking—in the original ReSimplifyIt framework, and ultimately produces the final output.

Then, we further ablate the video access completely, i.e. No access to the video content is provided throughout the *entire* reasoning process of the external agent. As the multi-turn interaction (feedback) are all essentially from the reference to video information to refine text input, disabling video access also renders multi-turn interactions unnecessary. To reflect this, we introduce the **ReSimplifyIt-blind** variant, in which a tool-calling LLM generates a rewritten query and a fixed sequence of tool invocations within a single-turn conversation. This sequence is subsequently executed by a separate executor module to produce the video output.

Refer to Figure 5 for an illustrations of these frameworks.

C.2 Evaluation Results

Evaluation results are presented in Table 7. Although the removal of modular design yields comparable video outputs, the modular architecture still demonstrates advantages—particularly on the *Multifaceted Integrative* subset, which involves more complex multi-hop reasoning, demonstrating the effectiveness of modular design and structured reasoning in more complex and intricate reasoning scenario. In comparison, the ablation of video access brings notable performance drop, underscoring the critical role of cooperative, feedback-driven multi-turn interaction. This aligns with the interdependency of the vision and text modality, which lies at the core of our TVS task formulation, reflecting the essence of the task initiative.

A performance–cost spectrum of TVS instantiations. Taken together, ReSimplifyIt, ReSimplifyIt-simple, and ReSimplifyIt-blind expose an explicit performance–cost spectrum of TVS rather than a single operating point. The full ReSimplifyIt prioritizes robustness and exploratory reasoning through iterative multi-agent interaction, at the cost of more tool calls, captions, and LLM turns. ReSimplifyIt-simple flattens the interaction structure into a single unified agent to substantially reduce overhead, at a small accuracy cost. ReSimplifyIt-blind further removes all video access during screening, representing the lowest-cost extreme in which the trimming plan is produced entirely from the language query. This spectrum demonstrates that TVS can be configured either for efficiency-oriented deployment (Simple/Blind) or for maximal reasoning capability (Full) depending on application requirements, and

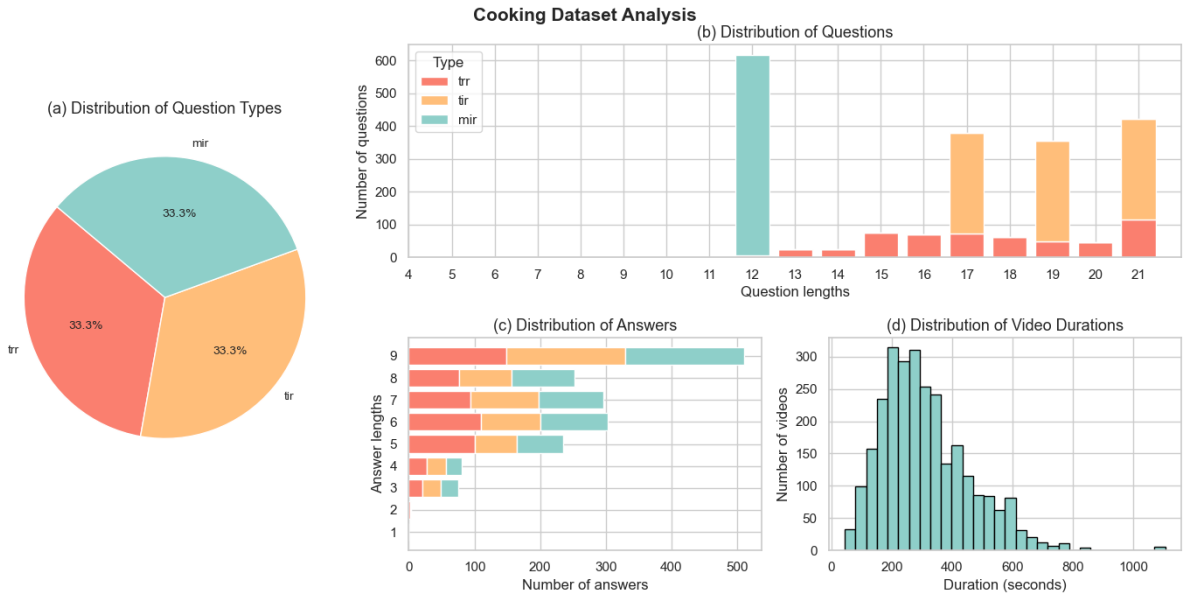

Figure 4: YouCookII-TVS statistics.

Method	Temporal Relational		Timepoint Indexed		Multifaceted Integrative		**Average**	
	mIoU	F1	mIoU	F1	mIoU	F1	mIoU	F1
ReSimplifyIt (Ours)	0.23	**0.37**	**0.98**	**0.99**	**0.47**	**0.64**	**0.56**	**0.67**
ReSimplifyIt-simple (Ours)	**0.24**	**0.37**	**0.98**	**0.99**	0.42	0.57	0.55	0.64
ReSimplifyIt-blind (Ours)	0.12	0.20	0.97	**0.99**	0.38	0.55	0.49	0.58

(a) Results on video output.

Method	Temporal Relational	Timepoint Indexed	Multifaceted Integrative	**Average**
ReSimplifyIt (Ours)	66.8	78.5	72.8	**72.7**
ReSimplifyIt-simple (Ours)	66.2	81.9	68.1	72.0
ReSimplifyIt-blind (Ours)	65.0	80.5	70.7	72.1

(b) Results on query rewriting.

Table 7: Ablation studies on our ReSimplifyIt framework. ReSimplifyIt-simple and ReSimplifyIt-blind represent ablations on modular design and feedback from video access, respectively.

that the observed gains arise from the screening mechanism itself rather than from any fixed amount of inference compute. Section 6 quantifies the cost characteristics of each variant across benchmarks.

C.3 Full list of tools for ReSimplifyIt-blind framework

```
1. get_duration():
   Return the duration of the video as a
     floating point value.

2. get_resolution():
   Return the resolution of the video, as a
     tuple.

3. get_total_frame_num():
   Return total number of the frames of the
     video, as an integer.

4. grounding_select(obj_name,
     concerned_indices_input):
   Return, in the form of a list of integers,
     the indices of all frames
   containing the object given by obj_name,
     after taking the intersection
   of indices provided by
     concerned_indices_input. If None is passed,
   selects all frames.

5. indices_list_intersect(list1, list2):
   Return the intersection of two lists of
     indices.

6. indices_list_union(list1, list2):
   Return the union of two lists of indices.

7. indices_concat_and_fill(list1, list2):
   Return the sorted union of list1 and list2,
     then fill in missing
   values to make the sequence continuous.

8. indices_concat(list1, list2):
```

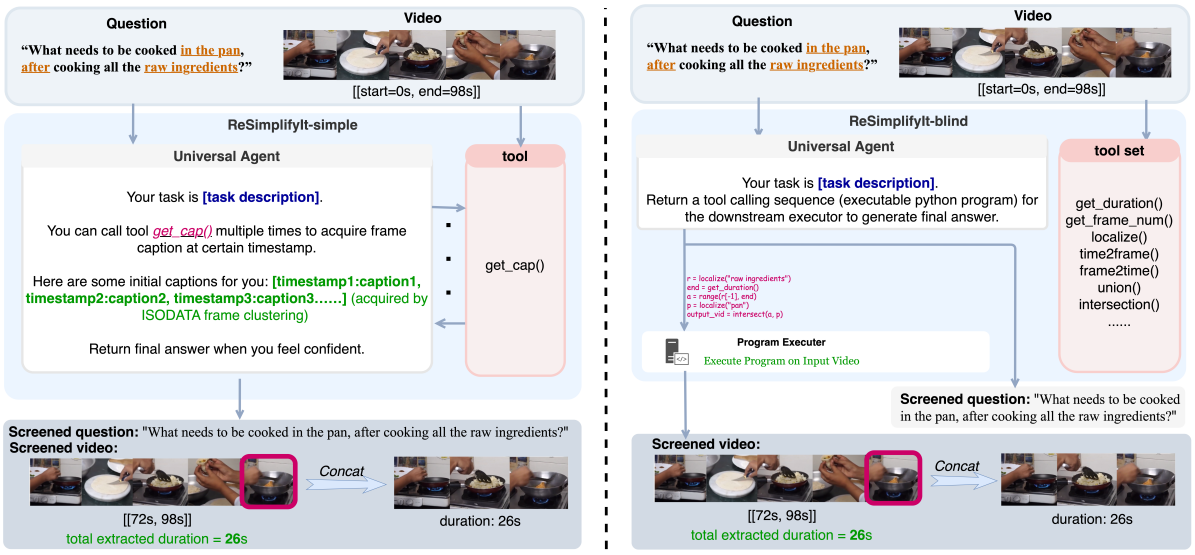

Figure 5: Workflow of *ReSimplifyIt-simple* (left) and *ReSimplifyIt-blind* (right).

```
   Return the concatenation of the two lists.

9. timestamp_to_single_index(timestamp):
   Return the frame index corresponding to the
    given timestamp (in seconds).

10. single_timestamp_to_index_range(timestamp):
    Return indices of 60 consecutive frames
     centered at the timestamp.

11. range_timestamp_to_index_range(start, end):
    Return all frame indices between the start
     and end timestamps.
```

D Implementation details of ReSimplifyIt framework

The Launcher module is based on single-turn conversation with GPT-4o. The Validator and Viewer module, including the scanner and localizer, are primarily based on multi-turn conversation with GPT-4o. For ISODATA frame clustering, we use CLIP (Radford et al., 2021) to obtain the visual features of the I-frames. We adopt the off-the-shelf LLaVA-1.5 (Liu et al., 2024a) for frame captioning.

E Prompts

E.1 Prompt used for evaluation of query rewriting of TVS

```
You are a helpful assistant to evaluate the
    quality of an output of a special type of
    sentence compression, which sentence is in
    the form of a question.

You will be given an output of such compression
    process and the ground truth answer, and the
     original input question sentence before the
     compression. Please evaluate the output
    based on the following three criterias:

1. Relevance: to minimize unwanted
information
- in this criteria, a candidate output gets
full mark if it doesn't contain any
information (phrases, concepts, etc.) out of
 the scope of the original input question.
- the more information it contains that is
not mentioned in the original input question
, the more marks are deducted.

2. Simplicity: to minimize tangential
information
- in this criteria, a candidate output gets
full mark if doesn't contain any information
 (phrases, concepts, etc.) that is included
in the original input question but not
included in the ground truth compressed
question. In other words, whether the
question sentence is fully compressed.
- the more information it contains that is
included in the original input question but
not included in the ground truth question,
the more marks will be deducted.

3. Completeness: to minimize over-
compression
- in this criteria, a candidate output gets
full mark if it contains all information
included in the ground truth compressed
question.
- the more information contained in the
ground truth output question is found
missing in the output compressed question,
the more marks will be deducted.

Here is the original input question: [
original_question_flag], and
here is the ground truth compressed question:
 [ground_truth_screened_question_flag], and
here is the output compressed question: [
output_screened_question_flag].

Please rate the output compressed question
on a scale from 0 to 100, with 0 being the
```

```
    worst and 100 being the best (full mark).
    Now, rate the quality of the output
    compressed question based on all the
    information above.
    Return your answer in this json format: {"
    score": [your score, from 0 to 100]}.
```

E.2 Prompts used for ReSimplifyIt framework

Prompt for the Launcher module:

```
Given a video question answering case, i.e. a
    video, a question, and an answer, sometimes
    it is possible to cut the video by only
    keeping a sub-clip or several sub-clips (
    concatenate if so) to be the video output
    and simultaneously modify the text question
    to be the paired text output while keeping
    the answer consistent and unchanged, so that
     the answer is still completely compatible
    to the modified question and the obtained
    video clip.
We define this action as "screening". By doing
    screening, the video becomes shorter, and
    therefore introduces lower cost to the
    downstream video question answering model.
    When modifying the question, note that the
    downstream video question answering model
    will only be seeing the sub-clip and think
    that the shown sub-clip is the whole video,
    so make sure the modified question is
    perfectly paired and aligned with and
    adapted to the modification.

[In context examples flag 1]

Sometimes, such screening can be repeatedly
    applied sequentially by several rounds,
    where the input in each round is the output
    video clip and the output modified question
    of its preceeding round. As mentioned, the
    answer consistency should be always ensured
    throughout the whole process, and is always
    completely compatible to the resulting video
     clip and modified question of every round.
    If succeeded, each round would make the
    video shorter and making the situation
    closer to the optimal case.

[In context examples flag 2]


Your duty is to help complete the screening. Due
     to some limit, you are only provided with
    the text question but not the video input.
Your task is to initiate an immediate plan for
    the screening operation, including a natural
     language instruction telling which video
    clip(s) to keep (similar to examples above)
    and the paired modified text question. If
    you think the screening may compose more
    than one round, you only need to perform one
     round next.
As you cannot see the actual video, your plan
    might fail, if the downstream video editting
     agent taking and executing your instruction
     on video clipping finds it infeasible.
    Therefore, your plan is refered to as a '
    trial'.
In your current situation, some rounds or trials
     of the screening process might have already
     been conducted, and privided as following
    information:
1, the 'failure history': it is about the
    history of failed previous trials of the
    screening of your current round. Here is the
     failure history for you (an empty indicates
     that you are making the first trial of this
     round): [failure_history_flag];
2, the 'success history': it is about the
    history of the one success trial of all
    previous round. Here is the failure history
    for you (an empty indicates that you are at
    the first round): [success_history_flag].

Now, here is the original question of this round
     for video question answering: "[
    quesion_flag]"
Again, please tell your plan which describes
    what part of the video should be kept. Also,
     give the modified question under the
    assumption that the video is processed
    smoothly according to your plan.

If you feel that there is no room to make such
    screening (e.g. when the question is being
    general like "what is this video about") so
    you feel that you shouldn't make any plan,
    you should decide to terminate the process.


Hints:
1. Before processing, remember to take a look in
     the 'failure history' and 'success history'
     information;
2. If you find that the success history contains
     a case whose modified_question and the
    description both significantly overlap with
    the ones you are about to make, then you
    should avoid making the same plan again. In
    this case, you should switch to a clear
    reasonable sensible alternative plan, or
    make a decision to terminate the
    modification process if you can't
    confidently find one;
3. If you find that the failure history contains
     a case whose modified_question and the
    description both significantly overlap with
    the ones you are about to make, or that any
    of the "reason" in the failure history
    records is going to make your attempt fail,
    then you should avoid making the same plan
    again. In this case, you should switch to a
    clear reasonable sensible alternative plan,
    or make a decision to terminate the
    modification process if you can't
    confidently find one.

Return your plan in this json format (keep in
    mind here, that your response should be in
    json format):
{"decision": [your decision, either "process" or
     "terminate"], "modified_question": [the
    modified question, or "N/A" if your previous
     decision is "terminate"], "description": [
    Description of what part of the video should
     be kept as wanted sub-clip. The description
     will be passed to downstream processer to
```

```
    validate. Return "N/A" if your decision is "
    terminate".]}

"""
```

Prompt for the Validator module:

```
You will be given a natural language instruction
     telling you to trim a video, which
    instruction itself might be infeasible. The
    reason it might be infeasible is because the
     agent who gave the instruction had no
    access to the actual video content, so it
    might be infeasible if take the actual video
     contenet into consideration.

Therefore, I need you to be a helpful assistant
    to confirm if the trimming plan is feasible.
Specifically, your job is to act as a validator
    to validate whether it is feasible (whether
    the video content really supports the plan).
If it is feasible, you will need to implement
    the plan and return the resulting sub-clip
    of the plan in the form of a two-layer list.
     [In context example flag 1]
If it is not feasible, you will need to tell the
     reason.

Here are some examples and explanations for
    infeasible trimming instructions:

[start of examples]

[In context examples flag 2]

[end of examples]

You can invoke a viewer multiple times to
    acquire the video content (partial content
    each time), which viewer is a downstream
    module prepared to assist you.

Note that the viewer is capable to deal with two
     type of questions:

1. snippet rough scanning, e.g. "what is the
    video about from xxx second to xxx second"?
2. localizing, e.g. "which segment contains xxx
    (event/object)?"

Therefore, if [your decision] is "view", [your
    message] should follow one of the above two
    example templates.

Here is the trimming instruction you need to
    validate: [plan_flag].
The video length is [video_length_flag] seconds,
     and its frame rate is [frame_rate_flag] fps
    .

Now, in each following turn of this conversation,
     you need to give your response in this json
     format: {"decision": [your decision], "
    message": [your message]}. This works as
    follows:
[your decision]: either "succeeded", "failed",
    or "view". "succeeded" means that the plan
    is successfully implemented, "failed" means
    that it is not, and by "view" you invoke the
     viewer to provide partial video content for
     you. If you choose "view", the viewer will
    take your message and return as you
    requested, and the conversation will
    continue. If you choose "succeeded" or "
    failed", it will be your final decision and
    the conversation will end.
[your message]: if you choose "succeeded" as
    your decision, then it should be the two-
    layer list as mentioned before, as the video
     edit result of the plan. If you choose "
    failed", this should be a brief reason on
    the failure (e.g. requested timestamp
    exceeds video length, video doesnt have the
    object/event needed, etc.). If you choose "
    view", this should be the question to ask
    the viewer about the video content.

Hint: it is not always necessary to invoke the
    viewer. [In context example flags 3]

For your ease of decision, here are some initial
     frame captions and their timestamps for you
     (in the form of key value pairs, where the
    value is the frame caption and the key is
    its corresponding timestamp, in the unit of
    second): [initial_captions_flag].
[sys_usr_split]

Now let's start!
```

Prompt for Viewer module:

```
You are a helpful and smart assistant that can
    respond to an upstream request a video by
    invoking tools. The length of this video is
    [video_length_flag].

Here is the content of the request: [
    validator_request_flag].

Here are the tools you can access (you might
    access them multiple times if you want):

1. scan(start, end): Return the overall caption
    of the video snippet (clip) between start
    and end timestamp, which are the parameters
    with the unit of second.
2. localize(query): Return the video location,
    in the form of a timestamp range given by
    the start and end timestamp, which contains
    the visual content of the query (which query
     might be an object, event, etc.)
3. get_image_cap(timestamp): parameter "
    timestamp" is an integer in the unit of
    second. Return the caption of the video
    frame at the given timestamp (regard the
    frame as an image).

Now, in each turn of the following conversation,
     your response should be in the following
    json format: {"decision": [your decision], "
    message": [your message]}. This works as
    follows:
[your decision]: either "tool" (if you wish to
    call tool in this round) or "respond" (if
    you feel that you are able to respond to the
     upstream module's request by comprehending
    your current knowledge acquired about the
```

```
    video.).
[your message]: if your decision is "tool", then
     this should only contain tool calling
    following given format, e.g. 'get_image_cap
    (10)', 'scan(21, 35)', 'localize("kicking
    the ball")'. If your decision is "respond",
    then this should be your response to the
    upstream request.

[sys_usr_split]

Now let's start!
```

E.3 Prompt used for ReSimplifyIt-simple framework

```
You are a assistant for the video question
    answering process, in which a candidate is
    presented with a video and a question for
    them to answer.\
Your objective is to help the candidate so that
    they will be able to give the answer with
    watching the shortest posible sub-clip(s) of
     the video. \
Your task is to cut the video to acquire this
    sub-clip(s) and also to modify the question,
     so that the candidate directly answering
    your modified question with presented only
    this sub-clip(s) of the video would be
    equivalent to answering the original
    question with presented the original whole
    uncut video. \

[In context examples flag 1]

You will need to cut the video in the form of
    providing me the timestamps, which is a list
     of [start, end] unit clips in the unit of
    second. \
A tool (python function) will be helping you to
    get the frame caption of at a certain
    timestamp (in the unit of second). Whenever
    you need to call this tool, send a message
    in this json format: {"decision": "tool", "
    parameter": [timestamp you need]}. [In
    context example flag 2] \
Whenever you think you are confident enough to
    provide the timestamp, return {"decision": "
    end", "timestamps": [your result timestamps
    ], "revised_question": [your revised
    question]}. [In context example flag 3].

Before we formally begin, here is a set of
    original captions with their timestamps
    provided for you to have an overall rough
    understanding of the video: [
    initial_captions_flag].
Also, the frame rate of this video is [
    frame_rate_flag] frames per second, and the
    total duration is [duration_flag].

[sys_usr_split]Now let's begin! and the original
     question is "[original_question_flag]".
```

E.4 Prompt used for ReSimplifyIt-blind framework

```
You are a assistant for the video question
    answering process, in which a candidate is
    presented with a video and a question for
    them to answer.\
Your objective is to help the candidate so
    thattheywill be able to give the answer with
     watching the shortest posible sub-clip(s)
    of the video. \
Your task is to cut the video to acquire this
    sub-clip(s) and also to modify the question,
     so that the candidate directly answering
    your modified question with presented only
    this sub-clip(s) of the video would be
    equivalent to answering the original
    question with presented the original whole
    uncut video. \

[In context examples flag 1]

You will be provided with a list of tools to
    process the video, and the original question
     to be answered by the candidate based on
    which to select the frames. Here is the list
     of tools you have access to, with the
    description (content in the brackets are the
     arguments needed):
[1]: get_duration(): return the duration of the
    video as a floating point value.
[2]: get_resolution(): return the resolution of
    the video, as a tuple.
[3]: get_total_frame_num(): return total number
    of the frames of the video, as an integer.
[4]: grounding_select(obj_name,
    concerned_indices_input): return, in the
    form of a list of integers, the indices of
    all frames containing the object given by
    obj_name, after taking the intersection of
    indices provided by the argument '
    concerned_indices_input'. '
    concerned_indices_input' is also a list of
    indices, and will be set to indices of all
    frames in the video if 'None' is passed.
[5]: indices_list_intersect(frame_indices_list_1,
     frame_indices_list_2): return, in the form
    of a list of integers, the intersection of
    the two arguments as list. Both argument '
    frame_indices_list_1' and '
    frame_indices_list_2' are a list of indices.
[6]: indices_list_union(frame_indices_list_1,
    frame_indices_list_2): return, in the form
    of a list of integers, the union of the two
    arguments as list. Both argument '
    frame_indices_list_1' and '
    frame_indices_list_2' are a list of indices.
[7]: indices_concat_and_fill(
    frame_indices_list_1, frame_indices_list_2):
      first take the sorted union of the two
    lists given by the arguments, and then fill
    in all the missing values so that every two
    adjacent element only differ by 1. Both
    argument 'frame_indices_list_1' and '
    frame_indices_list_2' are a list of indices.
[8]: indices_concat(frame_indices_list_1,
    frame_indices_list_2): return, in the form
    of a list, the concatenation of the two
    lists provided by the arguments.
[9]: timestamp_to_single_index(timestamp):
    return a list with a single integer, which
    integer is the index of the frame at the
```

```
    given timestamp. The argument timestamp is a
     floating point value, whose unit is second.
[10]: single_timestamp_to_index_range(timestamp):
     return, in the form of a list, the indices
    of 60 consecutive frames, the midpoint of
    which is at the given timestamp. The
    argument timestamp is a floating point value
    , whose unit is second.
[11]: range_timestamp_to_index_range(start, end):
     return, in the form of a list, the indices
    of all frames which are between the two
    timestamps which are provided by the
    arguments. The argument start and end are
    both floating point values, whose unit are
    both second.

Above are all the tools to have access to.
    Please note that selecting frames out of all
     the frames of the original video is being
    cut and clipped, therefore you will also
    need to modify the aforementioned prompt, to
     make it align well with the reduced video
    frames.
[sys_usr_split]
Now the original question is: [question_flag].
    Having access to the information of all the
    tools mentioned above, provide me the python
     code which could achieve the selection of
    frames. You may define variables to store
    intermediate result, and determine the value
     of some arguments when necessary, but you
    should not require the downstream task
    operator to replace any of your assumption
    on arguments, as no more information but the
     original video is provided to the
    downstream task. Please use the variable
    name 'final_frames' to store your final list
     of frame index. Please only provide the
    code and revised question in this format: {"
    Code:[your whole paragraph of code] Revised
    question:[your revised prompt]"}, where [
    your whole paragraph of code] should be an
    empty string if you think no tools need to
    be called and the whole original video
    should be passed to the downstream task.
```

F More on Related Work

Video-LLMs for VideoQA Video-LLMs have spurred a wave of models aimed at enhancing video understanding by leveraging the language capabilities of large language models (LLMs) (Lin et al., 2023; Ma et al., 2024; Li et al., 2024b, 2023b; Liu et al., 2024b; Xu et al., 2024a; Wang et al., 2025; Bai et al., 2025b,a; Li et al., 2024a). Some approaches (Chen et al., 2023; Li et al., 2024c, 2023a) utilize dedicated video encoders such as video transformers (Bertasius et al., 2021) or convolution networks. However, these designs often demand large-scale annotated video-text data and significant computational resources. To address this, alternative methods adapt pre-trained image-domain MLLMs to video inputs (Liu et al., 2024c; Maaz et al., 2024), offering improved practicality. Nonetheless, the sparsity and query-invariant nature of video encoding in existing models limits their ability to capture fine-grained spatial-temporal details effectively, especially under token budget constraints. This work addresses such inefficiencies by introducing a query-adaptive processing mechanism inspired by the principle of information screening, aiming to reconcile the trade-off between token efficiency and representational fidelity.

LLM-assisted Agentic Reasoning for VideoQA Another line of research for video question answering lies in building pure-text LLM assisted frameworks or multi-agent systems for VideoQA (Wang et al., 2024c; Shang et al., 2024; Wang et al., 2024b). Compared to Video-LLMs, which represents end-to-end single-pass approaches, these methods adopt traditional or LLM-based methods to proactively sample relevant video frames. VideoAgent (Wang et al., 2024b) and TravLER (Shang et al., 2024) utilized LLM's planning ability to conduct iterative keyframe searching, while VideoTree (Wang et al., 2024c) presented query-adaptive hierarchical tree-based keyframe selection.

Temporal Sentence Grounding for Videos Temporal sentence grounding (TSG) aims to localize the video segment best matching a language query. Early sliding-window methods (e.g., TALL (Gao et al., 2017), MCN (Hendricks et al., 2017)) were costly, while later proposal-based (Xu et al., 2019; Chen and Jiang, 2019) and proposal-free methods (Yuan et al., 2019; Zhang et al., 2020) improved efficiency via query-guided proposals or direct boundary prediction. All these methods take a video and a query as input and predict a matching temporal span. In contrast, our proposed TVS generalizes beyond TSG by supporting inter-frame reasoning and joint adaptation over both video and query, rather than retrieving only frames directly mentioned in the query.

Grounded videoQA Another seemingly similar line of research lies in integrating grounding techniques into VideoQA pipelines (Xiao et al., 2024; Chen et al., 2024a). Grounded VideoQA seeks to enhance model faithfulness by explicitly linking a model's textual output to "visual cues" or "evidence" within the video. This approach is effective at mitigating hallucinations for descriptive queries

by enforcing *perceptual* alignment. While effective for descriptive tasks, the approach suffers from a fundamental conceptual incongruity with the nature of complex video reasoning. The semantics of a query and its corresponding video segment are often holistically entangled; for example, a high-level question about intent, causality, procedure, or even a simple temporal relational reasoning does not map to an atomic piece of visual evidence but is inferred from a continuous temporal context. The attempt to impose a discrete justification framework onto this intertwined semantic space leads to an inherently brittle and ill-defined notion of a "visual cue".

TVS naturally sidesteps this ambiguity by fundamentally reframing the objective, turning from ***"what to answer"*** to ***"what to ask"***. More fundamentally, TVS introduces a more principled paradigm that respects this intrinsic entanglement. Rather than seeking to atomize evidence for a given answer, TVS reformulates the task itself through a priori contextual simplification. It isolates the minimal, self-contained $(video, query)$ sub-problem through an **endomorphic transformation** that keeps with the original task space unchanged and preserves the necessary holistic context for reasoning. This approach is not merely a circumvention of the definitional challenge; it establishes a more fundamentally sound and cognitively aligned task. By first reducing the problem space to a manageable and semantically coherent unit—mirroring the human strategy of simplifying a problem before attempting to solve it—TVS offers a more robust foundation for complex video-language understanding.

G Cost Driver Details

This section provides per-dataset cost-driver statistics for both TVS instantiations, complementing the headline CostRatio analysis of Section 6. Table 8 reports the high-level cost drivers (caption count, tool calls, LLM turns, output tokens) and screening effectiveness (DurAll, DurScrn, Screen%) for ReSimplifyIt-simple. Table 9 reports the same high-level statistics for the full ReSimplifyIt. Table 10 further decomposes the captions of the full ReSimplifyIt into passive (pre-loaded) versus active (tool-fetched) sources; column definitions follow the list below.

- **Pasv**: passive captions per sample (= VaIn + LoIn), embedded in prompts without explicit tool calls.
- **VaIn**: Validator initial captions, uniformly sampled frames embedded in the Validator's prompt ($\sim$10 per trial).
- **LoIn**: Localize initial captions; each `localize()` call samples k=10 uniform frames for its sub-agent prompt.
- **Actv**: active captions per sample (= #TotalCap − Pasv), fetched during tool execution via `scan`, `localize`, and `get_image_cap`.
- **V→Vw**: Viewer sessions spawned per sample (number of times the Validator invokes the Viewer as a tool).
- **#scan**, **#localize**, **#get_cap**: the breakdown of total Viewer tool calls (VwTl) into its three constituent tool types.

Dataset	#TotalCap	#Tool	#LLM	#OutTok	DurAll	DurScrn	Screen%
ActivityNet-QA	21.4	1.9	3.2	341	9.8%	9.0%	99.2%
YouCookII-TVS	21.9	1.9	3.3	366	6.4%	6.4%	100.0%
NExT-OE	23.4	3.9	5.0	433	20.2%	16.7%	95.7%
EgoSchema	22.1	2.1	3.5	446	26.0%	26.0%	100.0%
LVBench	27.0	7.0	9.6	1482	2.2%	1.9%	99.7%
MLVU	24.9	4.9	6.9	1122	12.2%	7.0%	94.4%
Video-MME	23.9	3.9	5.7	806	8.3%	8.0%	99.7%

Table 8: Per-dataset cost drivers and screening effectiveness for ReSimplifyIt-simple.

Dataset	#TotalCap	V→Vw	VwTl	DurAll	DurScrn	Screen%
ActivityNet-QA	67.5	2.7	6.8	51.7%	16.2%	57.6%
YouCookII-TVS	73.9	2.4	5.8	6.8%	6.8%	97.9%
NExT-OE	18.3	0.4	1.3	86.9%	32.8%	19.5%
EgoSchema	50.6	2.3	4.5	75.0%	25.6%	33.6%
LVBench	83.8	2.6	11.3	57.5%	4.8%	44.6%
MLVU	64.5	1.7	5.2	81.4%	7.2%	20.1%
Video-MME	76.2	2.4	6.7	64.0%	10.4%	40.1%

Table 9: Per-dataset cost drivers and screening effectiveness for the full ReSimplifyIt. VwTl denotes total Viewer tool calls per sample.

Dataset	Pasv	VaIn	LoIn	Actv	#scan	#loc	#get_cap
ActivityNet-QA	34.0	17.4	16.6	33.5	2.6	1.7	2.4
YouCookII-TVS	38.2	22.8	15.4	35.7	3.0	1.5	1.1
NExT-OE	5.3	2.7	2.6	13.0	0.4	0.3	0.4
EgoSchema	24.5	13.6	10.8	26.1	2.5	1.1	0.8
LVBench	48.9	17.5	31.3	94.9	4.8	3.1	3.0
MLVU	22.1	9.8	12.3	42.4	2.1	1.2	1.6
Video-MME	32.8	15.0	17.8	43.4	3.3	1.8	1.5

Table 10: Full per-source caption breakdown for ReSimplifyIt. #loc abbreviates #localize. Passive captions are pre-loaded into agent prompts; active captions are fetched during tool execution.

H Additional Discussion

H.1 TVS vs. Test-Time Reasoning Strategies

TVS and test-time reasoning strategies such as Chain-of-Thought (Wei et al., 2022), Tree-

of-Thoughts (Yao et al., 2023), and Graph-of-Thoughts (Besta et al., 2024) address different bottlenecks and are not substitutes for one another. Test-time reasoning strategies primarily modify *how* an LLM explores or restructures its thought process given a fixed input; they do not directly address the sparsity of evidence in long videos, where the dominant failure mode is that a fixed downstream frame budget under-covers the relevant segment to begin with. By contrast, TVS is a front-end endomorphic problem transformation that reshapes the *evidence distribution* itself: by screening $(V, Q) \to (v, q)$ under answer invariance, it concentrates the same downstream frame budget onto a short relevant segment, yielding the density gains quantified by our $\mathrm{DensAmp}$ and $\mathrm{CostRatio}$ metrics in Section 6.

These two directions are therefore orthogonal and complementary: adding reasoning steps does not automatically reproduce TVS's benefit unless it is coupled with explicit evidence selection at the sampling stage, while TVS can be composed with any downstream reasoning strategy. Our framework further exposes a performance–cost spectrum rather than a single operating point (ReSimplifyIt, ReSimplifyIt-simple, and ReSimplifyIt-blind; Appendix C), with progressively lower amounts of inference compute. The fact that screening gains persist even in the flattened and blind variants supports that TVS's benefit stems from the evidence-screening mechanism rather than from the sheer amount of LLM compute invested.

H.2 Minimality as a Desideratum and Boundary Expansion

The minimality condition in our formulation (Section 2.1) describes a *target property* of the task formalization—that the output segment v should contain the minimal sufficient visual evidence for answering q—rather than a strict guarantee enforced by any particular implementation. In practice, event boundaries in real-world videos are often ambiguous or gradual, and strict minimal cropping is a known failure mode: small boundary errors can easily exclude critical visual evidence and lead to disproportionately large downstream reasoning failures. Our Localizer (Appendix A) therefore applies a conservative temporal expansion of $\pm p$ seconds around the predicted boundary as a safety margin. This expansion is a tunable engineering parameter that trades a small amount of strict compactness for robustness against localization noise, and does not contradict the minimal-sufficiency objective.

Sensitivity to the expansion margin. To validate that $p{=}5$ is not arbitrary, we vary $p \in \{1, 3, 5, 7\}$ and report the resulting average mIoU on YouCookII-TVS:

p (seconds)	Average mIoU
1	0.51
3	0.53
5	**0.56**
7	0.49

We observe a clear trade-off: too small a margin (1–3s) risks missing event boundaries and lowers overlap with the ground truth, while too large a margin (7s) unnecessarily enlarges the segment and reduces localization precision. A moderate margin of 5 seconds achieves the best balance, and we adopt it as the default in all reported results. An adaptive, confidence-aware or boundary-aware expansion strategy would be an interesting direction for future work.