

OmniCap-IF: Benchmarking and Improving Instruction Following Abilities for Omni-Video Captioning

Jiahao Wang^{1*}, An Ping^{1*}, Yanghai Wang^{1*},
 Yuanxing Zhang², Shihao Li¹, Hanyan Bian¹, Yichi Ren¹,
 Yize Zhang¹, Han Wang¹, Haowen Chen¹, Junze Li¹,
 Jiaqi Wang¹, Yiyang Hu¹, Zhuze Xu¹, Zijie Zhang¹, Jiaheng Liu^{1,†}

¹ NJU-LINK Team, Nanjing University ² Kling Team, Kuaishou Technology
 jiahaowang@smail.nju.edu.cn liujiaheng@nju.edu.cn

Abstract

While Omni-modal Large Language Models (OLLMs) have demonstrated impressive capabilities in jointly processing audio and visual streams, their ability to strictly adhere to complex, multi-faceted user instructions remains largely unexplored. Existing benchmarks primarily focus on holistic video understanding or text-only instruction following, failing to capture the intricate interplay between modalities and user constraints. To bridge this gap, we introduce OmniCap-IF, the first comprehensive benchmark specifically designed to evaluate instruction-following capabilities in omni-modal captioning. OmniCap-IF incorporates a systematic framework that assesses captions on two dimensions: format correctness and content correctness. Our benchmark encompasses 50 distinct constraint types across pure visual, pure audio, and audio-visual modalities, while integrating Temporal Grounding to assess spatio-temporal precision. Extensive evaluations of prominent models on 1,920 high-quality samples reveal significant performance disparities. Furthermore, our analysis uncovers a critical “format-content tradeoff,” demonstrating that increasing formatting complexity directly degrades models’ omni-modal reasoning abilities. Finally, to advance the field, we curate a 54K instruction-tuning dataset, OmniCap-IF-54K and present OmniCaptioner-IF, which achieves notable improvements in both complex instruction adherence and general omni-modal captioning performance.

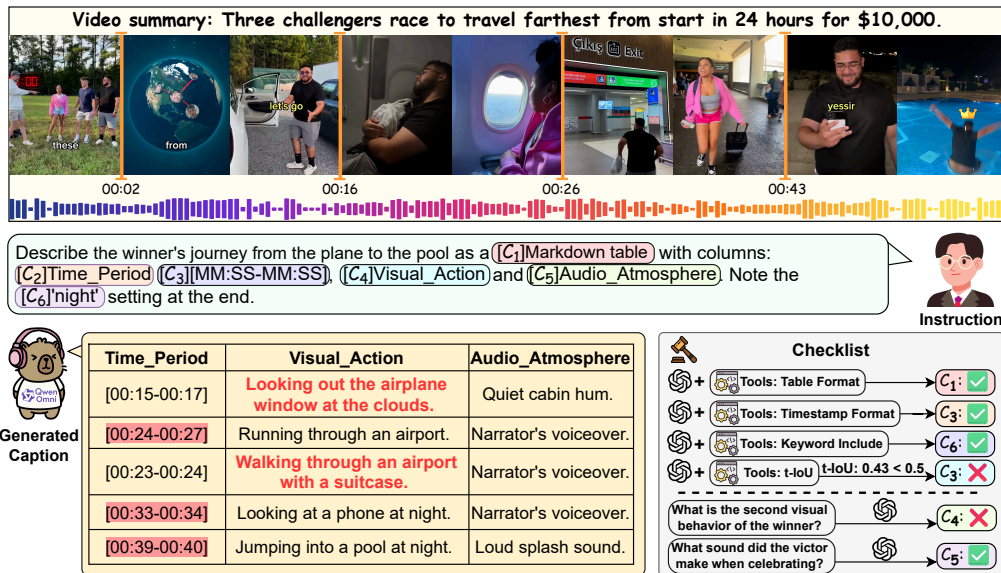


Figure 1: Overview of the OmniCap-IF evaluation framework. A typical case takes audio-visual content and a constraint-rich instruction as input. The generated response is evaluated against a specific checklist using a dual-mechanism approach: (1) Format and temporal constraints are initially extracted by the judge LLM and subsequently evaluated using predefined tools (e.g., format checkers, t-IOU) to ensure objective assessment. (2) Content constraints are verified by the judge model that answers preset questions to confirm the factual accuracy of visual and audio details.

* Equal Contribution. † Corresponding Author.

1 Introduction

The evolution of Multimodal Large Language Models (MLLMs) has recently transitioned from vision-language integration to omni-modal perception, enabling joint reasoning over text, audio and visual streams natively (Liu et al., 2023a; Dai et al., 2023; Chu et al., 2023). Despite their proficiency in general video description, high-quality, controllable outputs are crucial for a range of downstream tasks, including structured dual-track scripts for text-to-audio-video (T2AV) generation (Liu et al., 2024), egocentric action descriptions for embodied task planning (Chen et al., 2026), and precise semantic fingerprints for cross-modal retrieval (Peng et al., 2026). Models must not only understand the omni-modal content but also adhere strictly to complex, user-defined instructions. As illustrated in Figure 1, even leading models struggle to balance multi-modal perception with rigorous constraint satisfaction, often sacrificing instruction fidelity for descriptive verbosity (Liu et al., 2023b; Guan et al., 2024).

Currently, evaluating an omni-modal model’s capacity to fulfill compositional constraints remains an unexplored challenge (Tian et al., 2018). Existing benchmarks either prioritize semantic richness and question-answering accuracy over programmatic verifiability (Li et al., 2024; Maaz et al., 2024) or are confined to single-modality instruction following (Zhou et al., 2023; Bitton et al., 2023). Consequently, they lack the joint audio-visual complexity and structural rigor required for comprehensive omni-modal evaluation.

To bridge this gap, we propose **OmniCap-IF**, the first benchmark dedicated to instruction-following in omni-modal captioning. We establish a systematic constraint framework of 50 constraint types spanning format and content dimensions—the latter decomposed into Visual, Audio, and Audio-Visual modalities. Furthermore, we incorporate Temporal Grounding (Krishna et al., 2017) to enable quantitative assessment of precise timestamp localization, better aligning the evaluation with real-world scenarios.

Moreover, through our decoupled evaluation protocol, we investigate the impact of formatting difficulty. We uncover a significant “format-content tradeoff”—demonstrating that as structural format constraints become more rigorous, models’ fundamental ability to accurately reason over audio-visual content drastically degrades. Finally, to advance controllable generation, we construct OmniCap-IF-54K, a large-scale omni-modal instruction-tuning dataset, and present OmniCaptioner-IF, demonstrating a viable path toward highly controllable omni-modal assistants.

In summary, our key contributions are:

- **The first instruction-following benchmark for omni-modal captioning.** We introduce OmniCap-IF, featuring 1,920 complex, compositional instructions tailored for downstream applications.
- **A robust evaluation protocol disentangling format and content assessment.** We design a system that separates structural verification from semantic fidelity. It comprehensively covers Visual, Audio, and Audio-Visual constraints while uniquely incorporating Temporal Grounding.
- **Discovery of the “format-content tradeoff”.** We uncover and empirically prove that strict syntactic constraints (e.g., JSON) severely bottleneck models’ fundamental reasoning capabilities.
- **A high-quality training dataset and a strong baseline for controllable generation.** We release OmniCap-IF-54K along with the OmniCaptioner-IF model. Our results demonstrate that targeted instruction tuning significantly enhances both instruction adherence and general omni-modal perception.

2 Related Work

2.1 Instruction-Following Benchmarks

Evaluating instruction adherence has evolved significantly alongside the rapid development of large language models. Early text-based benchmarks primarily focused on assessing models against verifiable programmatic constraints, multi-level structural formatting, and complex logical rules (Zhou et al., 2023; Jiang et al., 2024; Wen et al., 2024). Recent efforts have extended this evaluation paradigm to vision-language tasks (Bitton et al., 2023; Li et al., 2026a). Furthermore, while recent studies have observed that enforcing strict structural formatting can degrade the intrinsic reasoning capabilities of Large Language Models (Tam et al., 2024; Deng et al., 2025), this phenomenon remains largely unexplored in complex multi-modal scenarios. Despite these advancements, existing evaluations remain confined to partial modalities and fail to meet the intricate requirements of emerging downstream applications. OmniCap-IF advances this paradigm by introducing omni-modal constraints and fine-grained temporal localization, effectively bridging the gap toward comprehensive omni-modal instruction following.

2.2 Omni-Modal Captioning Benchmarks

The recent advent of native omni-modal large language models has significantly expanded the boundaries of joint audio-visual understanding. Consequently, recent omni-modal captioning benchmarks primarily focus on assessing the semantic accuracy and descriptive richness of generated text, rather than a model’s capacity to follow arbitrary or user-specified instructions. These benchmarks commonly adopt structured evaluation paradigms, including curated question–answer pairs (Wu et al., 2025), cloze-style assessments (Ma et al., 2026), detailed holistic audio-visual descriptions (Tang et al., 2025) and temporally-grounded cinematographic scripts (Yao et al., 2026). Moreover, traditional temporal grounding tasks typically frame fine-grained event localization as isolated, pre-defined predictive tasks (Lei et al., 2020), lacking flexibility for dynamic or customized constraints. While such designs play a vital role in advancing omni-modal descriptive performance, they share a fundamental limitation: evaluation is conducted against a predefined and static set of quality criteria. In contrast, OmniCap-IF represents the first benchmark in omni-modal captioning that explicitly targets a model’s ability to understand and execute diverse, compositional instructions spanning visual, auditory, and audio-visual modalities.

3 OmniCap-IF

3.1 Constraint Framework

To systematically evaluate omni-modal controllability, we construct a taxonomy encompassing 50 constraint types categorized into two primary dimensions (Figure 2d):

- 1. Format Constraints:** Covers objective Structural (e.g., JSON arrays, Markdown tables) and Stylistic (e.g., length limits, specific delimiters) requirements.
- 2. Content Constraints:** Demands fine-grained factual comprehension across three granularities: (1) Visual (perceivable solely from the visual track, e.g., visual entities); (2) Audio (derivable exclusively from the auditory stream, e.g., speaker timbre); (3) Audio-Visual (requiring the simultaneous integration of both streams, such as audio-visual event alignment).

Further granular details regarding the classification and task definitions can be found in the supplementary material.

3.2 Data Collection and Annotation

3.2.1 Video Collection

To construct a high-quality evaluation benchmark, we curate a test set of 480 videos by compiling a large-scale, copyright-free video pool sourced from YouTube, TikTok, and Ego4D (Grauman et al., 2022). The videos are rigorously filtered to ensure both audio-visual richness and audio-visual alignment. The final collection spans a wide range of domains—from comedy to technology—thereby enhancing the overall reliability and diversity of the benchmark.

3.2.2 Annotation Pipeline

Our annotation pipeline follows a two-stage framework that integrates automated generation with human expertise, ensuring both scalability and high annotation quality.

Stage 1: Automated Draft Generation. For each video, an Instruction Generator produces paired instruction–checklist annotations. The prompts for generation can be found in the supplementary material.

Stage 2: Human Refinement and Verification. Professionally trained annotators carefully review and refine the automatically generated drafts, resulting in 53.1% of samples modified and 22.7% discarded and rewritten. Each sample is finalized only upon unanimous agreement among three annotators, with any disagreements adjudicated by a senior supervisor. Through this rigorous process, we obtain a final dataset comprising 1,920 high-quality samples.

3.3 Dataset Statistics

3.3.1 Overall Statistics

Statistical analysis underscores the comprehensive nature of OmniCap-IF, highlighting its substantial diversity in duration, content coverage, and instructional complexity (Figure 2). The dataset exhibits a

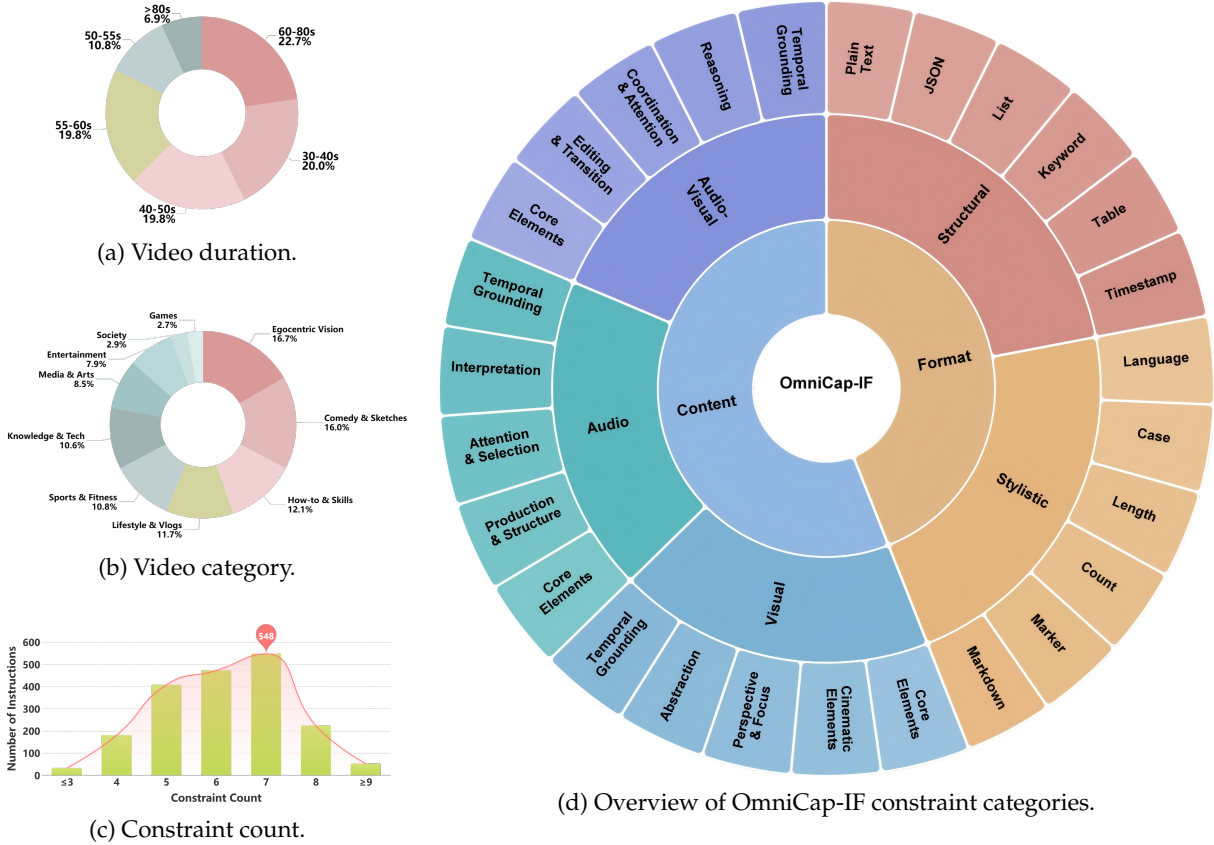


Figure 2: Dataset statistics for OmniCap-IF. (a-c) show distributions for video duration, category, and constraint count, respectively. (d) provides an overview of the constraint categories.

well-balanced distribution of video durations, with its average duration exceeding most existing omni-modal captioning benchmarks (Figure 2a). In addition, its wide-ranging content, covering numerous categories (Figure 2b), supports evaluation of cross-domain generalization. The instruction set further spans a spectrum from standard prompts to highly complex cases (Figure 2c). Collectively, these properties position OmniCap-IF as a next-generation testbed for evaluating OLLMs.

3.3.2 Comparison with Other Benchmarks.

When compared with other benchmarks, we adopt IFEval (Zhou et al., 2023), CELLO (He et al., 2024), InfoBench (Qin et al., 2024a), FollowBench (Jiang et al., 2024), SysBench (Qin et al., 2024b), CFBench (Zhang et al., 2025), ComplexBench (Wen et al., 2024) and IF-VidCap (Li et al., 2026a) as the instruction-following baselines. For omni-modal captioning, we compare against UGC-VideoCap (Wu et al., 2025), Omni-Cloze (Ma et al., 2026), OmniDCBench (Yao et al., 2026), and video-SALMONN-2-testset (Tang et al., 2025). As shown in Table 1, OmniCap-IF advances the landscape of both instruction-following and omni-modal captioning benchmarks. In contrast to prior datasets that focus solely on text-only or vision-only instruction following, it incorporates omni-modal inputs while achieving a larger scale, increased instructional complexity, and more comprehensive content coverage. From the perspective of omni-modal captioning, OmniCap-IF shifts the focus from conventional descriptive or holistic narratives toward fine-grained instruction adherence, featuring richer informational content and, in general, longer video durations than most existing benchmarks. By bridging these directions and further introducing temporal grounding mechanisms, OmniCap-IF establishes a more rigorous and versatile benchmark for evaluating controllable generation in OLLMs.

3.4 Evaluation Protocol

3.4.1 Evaluation Methodology

To rigorously assess model performance, OmniCap-IF employs a bifurcated evaluation strategy that disentangles structural adherence from semantic fidelity, as comprehensively illustrated in Figure 1. Inspired by IF-VidCap (Li et al., 2026a), we incorporate rule-based programmatic tools into our evaluation

Table 1: Comparison of Instruction Following and Omni-modal Captioning Benchmarks. “#Size”, “#Types”, and “#Const.” denote the total number of prompts, the number of distinct constraint types, and the average number of constraints per instruction, respectively. “Vid. Len.” refers to the average video duration. “Temporal” indicate whether the benchmark supports temporal grounding constraints. “Mod.” indicates the input modality (T: Text, V: Video, AV: Audio-Visual), while “Evaluation” specifies the methodology used for scoring.

Benchmark	#Size	#Types	#Const.	Vid. Len.	Temporal	Mod.	Evaluation
<i>Instruction Following Benchmarks</i>							
IFEval	541	25	1.54	–	–	T	Rule
CELLO	523	4	2.18	–	–	T	Rule
InfoBench	500	5	5.93	–	–	T	LLM
FollowBench	944	5	3.00	–	–	T	LLM / Rule
SysBench	500	6	2.38	–	–	T	LLM
CFBench	1,000	10-25	4.24	–	–	T	LLM
ComplexBench	1,150	4-19	4.61	–	–	T	LLM+Rule
IF-VidCap	1,400	27	6.00	20.5s	–	V	LLM+Rule
<i>Omni-modal Captioning Benchmarks</i>							
video-SALMONN-2-testset	483	–	–	50.8s	–	AV	LLM
UGC-VideoCap	1,000	–	–	23.9s	–	AV	LLM
Omni-Cloze	2,340	–	–	34.2s	–	AV	LLM
OmniDCBench	1,122	–	–	59.5s	✓	AV	LLM
OmniCap-IF (Ours)	1,920	50	6.93	54.6s	✓	AV	LLM+Rule

pipeline to significantly enhance the stability and reliability of the LLM-as-a-judge (Zheng et al., 2023).

Format Evaluation: This component targets objective structural requirements (e.g., length, JSON schema, or ordered lists). To ensure stable and robust verification, we employ a two-step hybrid approach: an LLM first extracts the structured information from the generated output, followed by the execution of rule-based programmatic tools to deterministically verify compliance against predefined formatting rules.

Content Evaluation: This component assesses instruction following regarding content constraints, explicitly prioritizing objective factual accuracy over descriptive fluency to mitigate LLM judge biases. We evaluate this through two complementary mechanisms:

- *Temporal Grounding Constraints:* An LLM extracts timestamps from the response, followed by rule-based tools computing temporal-IoU (t-IoU) or offsets to accurately determine temporal compliance. Comprehensive descriptions of the evaluation procedures are provided in the supplementary material.
- *Multimodal Content Constraints:* For the remaining constraints across visual, audio, and audio-visual dimensions, we leverage an LLM-as-a-judge via a Question-Answering (QA) approach. The evaluation uses binary and multiple-choice questions. By providing generated captions as context, we verify the factual alignment between the content and complex instructions.

3.4.2 Evaluation Metrics

We employ two primary metrics to quantify performance: **Constraint Satisfaction Rate (CSR)** and **Instruction Satisfaction Rate (ISR)**.

$$\text{CSR} = \frac{1}{m} \sum_{i=1}^m \frac{1}{n_i} \sum_{j=1}^{n_i} s_i^j, \quad \text{ISR} = \frac{1}{m} \sum_{i=1}^m \text{ISR}_i \quad (1)$$

where m is the total number of instructions, and n_i denotes the number of constraints for the i -th instruction. $s_i^j \in \{0, 1\}$ indicates whether the j -th constraint is satisfied. ISR_i is a binary indicator that equals 1 if and only if all constraints within the i -th instruction are simultaneously satisfied (i.e., $\sum_{j=1}^{n_i} s_i^j = n_i$), and 0 otherwise.

To provide a granular diagnostic of model capabilities, we report metrics across a hierarchical structure as categorized in Table 2:

- **Primary Evaluation Types:** We report Format CSR/ISR for structural control and Content CSR/ISR for semantic fidelity.

-
- **Modality-Specific Content Analysis:** The Content CSR is further decomposed into three distinct dimensions—Visual, Audio, and Audio-Visual (AV)—to precisely pinpoint the modality-specific instruction-following capabilities of various models.

3.5 OmniCap-IF-54K

To endow models with generalizable and robust instruction-following capabilities, we introduce a large-scale, high-quality fine-tuning dataset. To prevent data leakage, the generation pipeline is strictly decoupled from our evaluation benchmark. As illustrated in Figure 3, the process consists of three stages, ultimately yielding OmniCap-IF-54K, which comprises 54K meticulously curated video-instruction-response triplets.

Stage 1: High-Quality Omni-Modal Video Curation. We source raw videos from LLaVA-Video-178K (Zhang et al., 2024) and TikTok-10M (The Data Company, 2025). To ensure multimodal richness, we apply strict heuristic filters: (1) durations between 20 to 120 seconds, (2) visual resolutions of at least 480p, and (3) high acoustic density, filtered using PANNs (Kong et al., 2020) to guarantee the presence of diverse ambient sounds and speech. This results in 14K high-quality video samples.

Stage 2: Constraint-Aware Instruction Synthesis. We first generate fine-grained textual captions for all videos using ASID-Captioner-7B (Li et al., 2026b) to serve as dense multimodal proxies. Gemini-3-Flash (Google DeepMind, 2026) then synthesizes instructions by sampling from our constraint system. Crucially, we implement a negative constraint filter: any constraint whose prerequisite elements are absent from the proxy caption (e.g., blacklisting the “omni temporal grounding” constraint if the caption lacks any description of audio-visual desynchronization) is excluded to prevent hallucinations. Valid constraints are then combined to form complex, multi-constraint instructions.

Stage 3: Decoupled and Complexity-Aware Response Generation. As demonstrated in Figure 4, a model’s ability to satisfy constraints degrades significantly as the number of constraints increases. Consequently, generating a response for a heavily constrained instruction in a single pass often yields flawed ground truth. To circumvent this, we adopt a decomposed generation strategy. First, we separate the instruction into content constraints and format constraints. The content constraints are further partitioned into smaller, manageable sub-tasks containing only 2–3 constraints each. Based on the video caption, Gemini-3-Flash generates high-fidelity intermediate responses for these sub-tasks, which are then aggregated into a comprehensive, multi-constraint content response. Furthermore, as illustrated in Figure 5, enforcing rigid format constraints simultaneously can severely compromise the factual correctness of the generated content. Thus, we apply format constraints exclusively in the final stage: the model is instructed to reformat the aggregated content response to produce the final ground truth, ensuring both semantic richness and structural compliance. In addition, we conduct a study on 500 randomly sampled triplets by comparing our decomposed strategy with direct generation. Using the same checklist-based evaluation, we find that in 96.3% of cases, the decomposed-and-aggregated approach yields superior results to direct generation.

Details regarding the prompts used in the process and training configurations are provided in the supplementary material.

4 Experiments

4.1 Main Results

We evaluate 14 leading omni-modal models, including Gemini-3.1-Pro (Google DeepMind, 2026), Gemini-3-Flash, Gemini-2.5-Pro (Comanici et al., 2025), MiMo-V2-Omni (Xiaomi, 2026), Qwen3-Omni (Xu et al., 2025a), Qwen2.5-Omni (Xu et al., 2025b), ARC-Hunyuan-Video (Ge et al., 2025), HumanOmniV2 (Yang et al., 2025), MiniCPM-o (Yao et al., 2024), video-SALMONN2 (Tang et al., 2025) and ASID-Captioner (Li et al., 2026b).

The main results in Table 2 yield several key observations: (1) In the same model family, performance consistently improves as model size increases. (2) Generally, models perform better on Audio and Visual constraints independently than on Audio-Visual constraints, highlighting the difficulty of joint audio-visual integration. (3) Models demonstrate stronger capability in format control than in adhering to content-related requirements, likely because content understanding demands more complex multi-modal reasoning, whereas format constraints are predominantly text-based. (4) The human baseline exhibits a distinct performance pattern compared to advanced models. Benefiting from deliberate verification and self-reflective reasoning, human annotators achieve better results in format control, significantly outperforming all evaluated models.

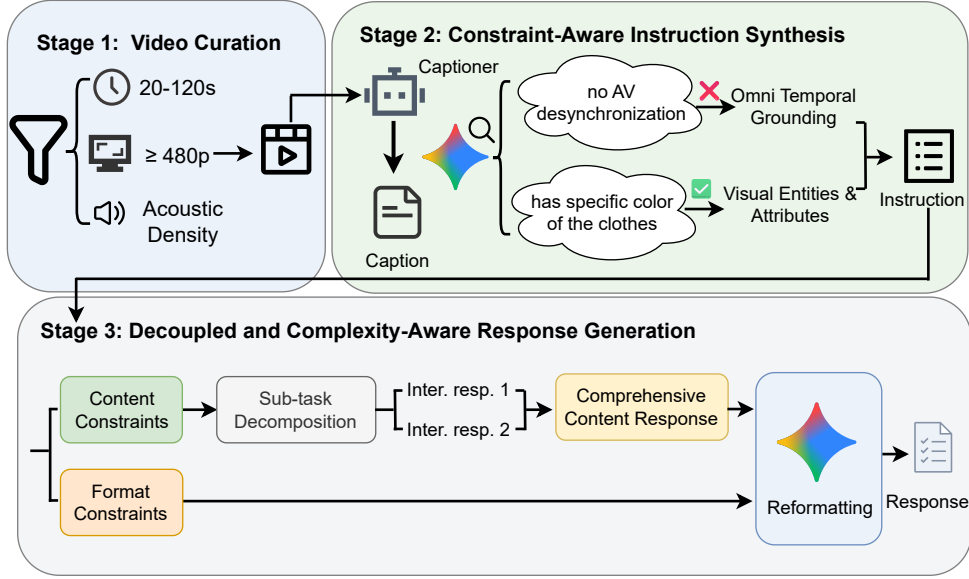


Figure 3: The training set generation pipeline. “Inter. resp.” stands for Intermediate Response.

Table 2: Main Evaluation Results on the OmniCap-IF Benchmark. The content CSR is further decomposed into Visual, Audio, and Audio-Visual modalities.

Model	Overall		Format		Content				
	CSR	ISR	CSR	ISR	CSR	ISR	Visual CSR	Audio CSR	AV CSR
Human	83.29	35.31	94.83	84.19	78.23	40.19	78.38	80.05	72.43
<i>Closed-Source Large Multimodal Models</i>									
Gemini-3.1-Pro	79.80	30.86	89.78	80.89	74.57	36.82	74.91	77.37	70.57
Gemini-3-Flash	78.70	29.04	85.75	74.17	75.01	37.26	75.20	76.73	71.67
MiMo-V2-Omni	74.59	22.18	81.55	67.28	70.93	31.84	70.92	73.60	67.85
Gemini-2.5-Pro	74.06	24.58	81.70	68.84	70.05	32.08	71.82	74.11	65.30
<i>Open-Source Large Multimodal Models</i>									
Qwen3-Omni-30B-A3B-Thinking	72.58	19.42	84.48	72.55	66.33	25.00	67.71	72.30	59.70
Qwen3-Omni-30B-A3B-Instruct	67.44	14.27	79.46	64.14	61.14	20.05	64.09	64.88	55.07
MiniCPM-o-4.5-9B	66.40	12.81	77.00	60.38	60.83	19.32	62.98	67.25	53.38
Qwen2.5-Omni-7B	57.44	7.55	66.78	46.52	52.54	12.76	57.13	57.14	44.18
Qwen2.5-Omni-3B	51.07	4.74	60.44	39.78	46.16	9.06	53.42	51.09	36.31
video-SALMONN-2-7B	44.58	1.88	47.80	23.99	42.89	6.05	47.28	48.94	34.86
MiniCPM-o-2.6-8B	43.39	1.67	40.85	18.07	44.72	6.58	51.26	49.88	37.14
HumanOmniV2-7B	41.85	2.40	37.86	18.18	43.95	7.10	51.04	48.16	36.54
ASID-Captioner-7B	33.87	1.04	22.70	7.74	39.72	6.20	48.45	46.48	30.50
ARC-Hunyuan-Video-7B	29.74	0.31	20.27	5.75	34.71	4.17	44.51	37.24	26.62
<i>Ours</i>									
OmniCaptioner-IF-7B (ours)	72.59	18.28	89.09	79.46	63.94	22.08	63.82	68.81	59.02
OmniCaptioner-IF-3B (ours)	69.09	13.28	86.78	74.96	59.81	16.93	61.57	64.25	54.04

Additionally, we develop the **OmniCaptioner-IF** series by fine-tuning Qwen2.5-Omni on OmniCap-IF-54K. OmniCaptioner-IF outperforms the base model across all metrics. Notably, it exhibits strong structural controllability, performing on par with the proprietary Gemini-3.1-Pro in format metrics, highlighting the effectiveness of our instruction-tuning for enforcing rigid constraints. This gain likely stems from format control relying on low-level textual signals that are easier to learn from limited supervision, whereas content constraints require more complex multimodal reasoning.

4.2 Results on Existing Benchmarks

We evaluate OmniCaptioner-IF on several external benchmarks—IF-VidCap (vision-only instruction following), Omni-Cloze (cloze-style fine-grained omni perception), and UGC-VideoCap (QA-based omni

video captioning)—to comprehensively assess its generalizable omni-modal perception capabilities. The results of Omni-VideoQA benchmarks can be found in the supplementary material.

Results on IF-VidCap. IF-VidCap strictly focuses on visual-only instruction adherence. We evaluate our model by providing only the video track. As shown in Table 3, OmniCaptioner-IF-3B surpasses the vision-expert model Qwen2.5-VL-Instruct-3B across all metrics. This demonstrates that our omni-modal instruction tuning not only preserves but enhances pure visual grounding capabilities.

Table 3: Results on the IF-VidCap Benchmark.

Model	ISR	CSR	Format		Content	
			ISR	CSR	ISR	CSR
Gemini-2.5-Pro	27.83	74.53	74.35	87.81	35.22	59.00
Qwen2.5-VL-Instruct-7B (Bai et al., 2025)	10.92	58.12	52.51	73.81	18.75	39.65
Qwen2.5-VL-Instruct-3B (Bai et al., 2025)	6.54	51.74	43.46	66.50	13.15	34.47
Qwen2.5-Omni-7B (w/o Audio)	8.17	56.49	54.12	74.41	14.04	36.76
Qwen2.5-Omni-3B (w/o Audio)	5.73	49.66	43.23	65.77	11.10	31.95
OmniCaptioner-IF-7B (w/o Audio)	12.21	61.20	61.33	79.92	16.57	40.63
OmniCaptioner-IF-3B (w/o Audio)	8.70	57.56	57.58	76.30	13.70	36.99

Results on Omni-modal Captioning Benchmarks. We further validate our model on comprehensive audio-visual benchmarks. As illustrated in Table 4, OmniCaptioner-IF-7B demonstrates a remarkable performance leap on Omni-Cloze, effectively doubling the total accuracy compared to the original baseline.

Table 4: Results on the Omni-Cloze Benchmark.

Model	Visual% \uparrow	Audio% \uparrow	AV% \uparrow	Total% \uparrow
Gemini-2.5-Flash (Comanici et al., 2025)	31.50	18.40	39.10	27.90
video-SALMONN-13B (Sun et al., 2024)	2.60	1.70	4.00	2.50
VideoLLaMA-2-7B (Cheng et al., 2024)	5.70	2.60	7.30	4.80
Qwen2.5-Omni-7B	10.40	12.90	18.90	12.90
OmniCaptioner-IF-7B (Ours)	23.86	24.23	32.30	25.17
OmniCaptioner-IF-3B (Ours)	21.27	21.81	28.94	22.53

On the UGC-VideoCap benchmark (Table 5), OmniCaptioner-IF-7B achieves performance comparable to Gemini-2.5-Pro. This highlights the efficacy of fine-grained constraint adherence as a proxy for enhancing general omni-modal understanding.

Table 5: Results on the UGC-VideoCap Benchmark.

Model	Audio \uparrow	Visual \uparrow	Detail \uparrow	Avg. \uparrow
Gemini-2.5-Pro	69.50	74.70	73.70	72.60
Qwen3-Omni-30B-A3B-Instruct	67.50	74.80	72.30	71.50
HumanOmniV2-7B	45.60	66.30	59.50	57.10
video-SALMONN-2-7B	61.80	71.40	68.50	67.20
Qwen2.5-Omni-7B	46.90	66.10	60.00	57.70
Qwen2.5-Omni-3B	48.20	55.60	52.60	52.18
OmniCaptioner-IF-7B (Ours)	69.79	75.94	73.19	72.97
OmniCaptioner-IF-3B (Ours)	67.71	73.91	70.43	70.68

4.3 Further Analysis

Impact of Instruction Complexity. We examine four representative models to explore the relationship between instruction complexity—jointly determined by prompt length and constraint count—and two metrics: Constraint Satisfaction Rate (CSR) and Instruction Satisfaction Rate (ISR). Evaluations are carried out on an expert-filtered subset comprising 1,000 high-quality instances from the benchmark. Within these samples, constraints are interdependent (e.g., branching, chaining) instead of being strictly isolated.

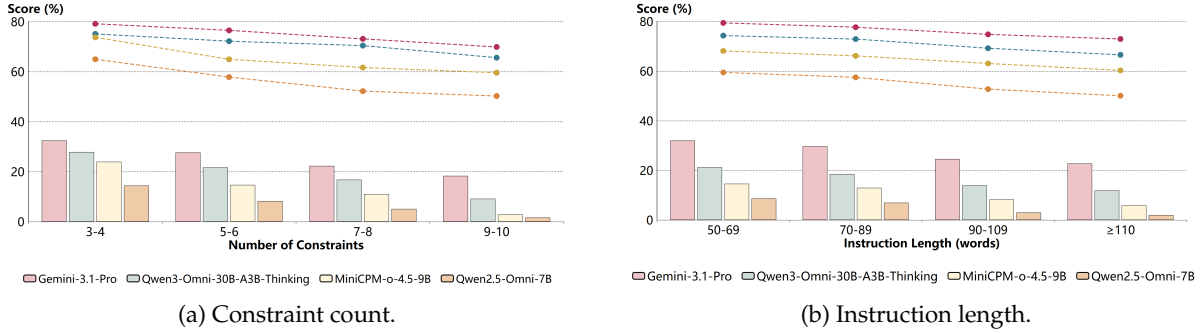


Figure 4: The impact of constraint count, instruction length on model performance.

This specific selection guarantees that the total number of constraints reliably reflects the true complexity of the task. Figures 4a and 4b explicitly illustrate that a model’s proficiency in satisfying constraints and following directives deteriorates with escalating complexity, substantiating that more nuanced and difficult commands severely strain models’ instruction-following abilities.

Format-content Tradeoff. To examine the impact of formatting complexity on a model’s ability to retain semantic depth, we designed a controlled experiment evaluated on 1,200 curated samples across five representative models. Specifically, we held the content constraints strictly constant while varying the format constraints across three levels:

- **Level 1 (Loose):** Natural language, basic paragraphs/bullets (e.g., plain text, length).
- **Level 2 (Styled):** Human-readable visual structuring requiring layout awareness (e.g., Markdown table, ordered lists).
- **Level 3 (Syntactic):** Machine-readable, strict grammatical rules (e.g., JSON arrays, forced keywords).

As illustrated in Figure 5, as the formatting level increases from the lowest level to the highest level, the content CSR drops continuously and noticeably. This indicates that forcing models to allocate attention to rigid syntactic generation (e.g., JSON nesting) directly cannibalizes their capacity for complex cross-modal reasoning.

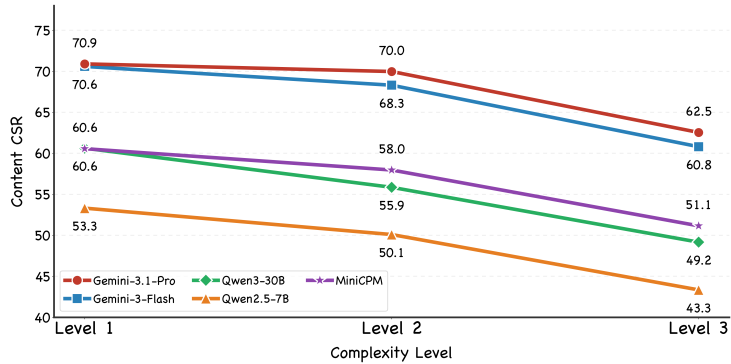


Figure 5: The format-content tradeoff.

Impact of Video Parameters. We examine Qwen2.5-Omni-7B and MiniCPM-o-4.5-9B under varying frame sampling rates (FPS). As shown in Figure 6, increasing FPS causes Format CSR to drop steadily, as more visual tokens overwhelms the context window and reduces the models’ ability to maintain strict structural adherence. Content CSR first increases and then decreases. The initial gain arises from richer visual evidence supporting fine-grained event perception, while excessive frame density adds redundant noise and context pressure, deviating from the models’ optimal training distributions and impairing abilities such as precise temporal grounding. The exact turning point varies across models, reflecting differences in their preferred visual token density.

Analysis of Cross-Modal Synergy. To assess whether current OLLMs achieve true audio-visual synergy, we perform a modality decoupling experiment. We derive uni-modal instructions by retaining only constraints of the original prompts that can be resolved solely through visual or auditory evidence (while preserving all format constraints). These instructions are then paired with their corresponding single-modal inputs and compared against the full omni-modal setting (Figure 7). Gemini-3.1-Pro and MiniCPM-o-4.5 exhibit strong cross-modal gains: adding visual context significantly boosts their Audio

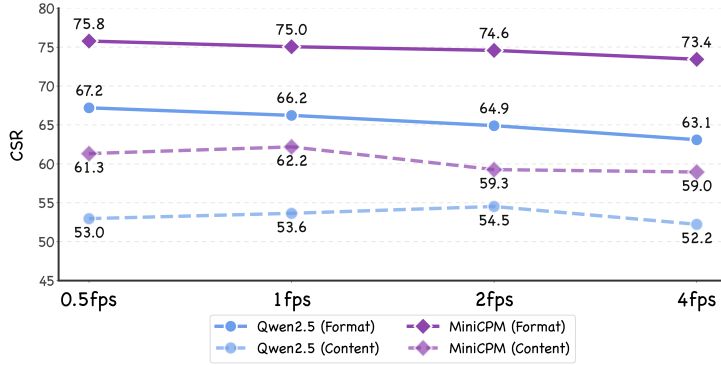


Figure 6: Impact of FPS on model performance.

CSR, showing effective use of visual cues to ground acoustic events. In contrast, the Qwen series shows minimal synergy. Qwen3-Omni and Qwen2.5-Omni achieve only slight improvements, with Qwen2.5-Omni even declining in Overall CSR. Degraded Visual CSR in MiniCPM-o-4.5 and Qwen2.5-Omni further highlights cross-modal interference, suggesting that while these models handle uni-modal inputs well, they largely process audio and visual streams independently rather than in a deeply fused manner.

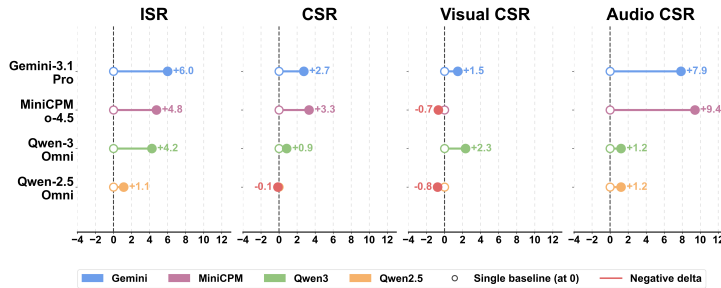


Figure 7: Analysis of Cross-Modal Synergy.

Agreement Evaluation. To validate our evaluation framework, we compare automated assessments with human judgment using the professional annotations described in Section 3.2.2. Agreement is measured across three assessor models: GPT-5-mini (Singh et al., 2025), Gemini-3-Flash (Google DeepMind, 2026), and Qwen3.5-27B (Qwen Team, 2026). As shown in Table 6, GPT-5-mini achieves the highest concordance with human evaluations across all metrics. The strong agreement across these diverse models highlights the robustness and general applicability of our methodology.

Table 6: Agreement between automated evaluation and human evaluation across different models.

Model	Overall Agreement	Format	Content
GPT-5-mini	94.70	96.12	94.29
Gemini-3-Flash	93.16	94.17	92.86
Qwen3.5-27B	92.49	94.66	91.86

Constraint Type Analysis. Our analysis of the CSR across representative models (Figure 9) reveals a pervasive performance bottleneck: while current OLLMs handle basic textual formats well, they struggle significantly with rigid structural formatting and fine-grained audio-visual constraints. Regarding format control, OLLMs face challenges with complex structures like JSON and strict patterns such as Timestamps, reflecting limitations in token-level output regulation. In content constraints, models show difficulties with directives related to Editing Transition, Temporal Grounding, and Anchor. The lower performance on Editing Transition suggests limited internalization of professional cinematic techniques, while the gaps in Temporal Grounding and Anchor indicate that visual and auditory streams are often processed as isolated channels. Notably, specialized video captioning models do not outperform general-purpose models (e.g., ASID-Captioner versus Qwen2.5-Omni-7B) under our evaluation, because our benchmark emphasizes precise adherence to instruction-specified attributes, actions, or events rather than unconstrained, detailed video descriptions.

The OmniCaptioner-IF series addresses these limitations with comprehensive improvements, outper-

forming baselines across the entire constraint spectrum. Notably, the models transform previously weak adherence to rigid formats like JSON and Timestamp into robust performance, while also showing stronger handling of fine-grained audio-visual constraints. This demonstrates that OmniCaptioner-IF excels both in strict output regulation and in deep cross-modal understanding. More details can be found in the supplementary material.

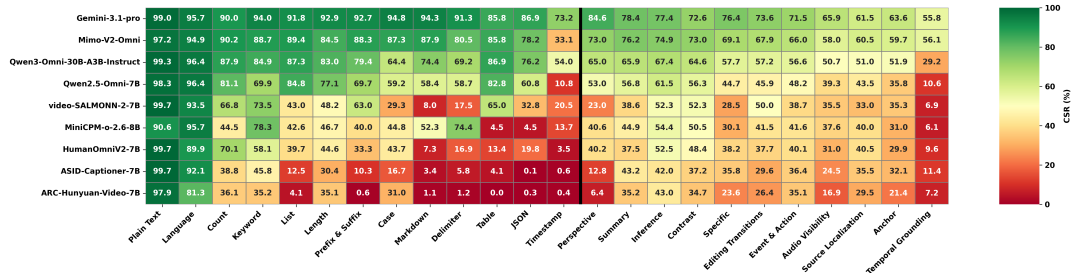


Figure 8: CSR performance of different models on different formats and audio-visual constraint types.

Error Analysis. Our analysis of model responses reveals key error categories. For format constraints, common violations are (1) malformed JSON (e.g., missing keys or bracket mismatches) and (2) incorrect timestamp formatting (e.g., not following the “MM:SS” template). For content constraints, frequent issues include (1) misidentifying or omitting Editing Transitions, (2) inaccurate Temporal Grounding of events, and (3) failing to establish cross-modal Anchors. We also find that when only audio is provided, the models’ audio temporal grounding capability is significantly weaker than when both audio and visual modalities are available. More examples are provided in the supplementary material.

5 Conclusion

In this work, we introduce OmniCap-IF, a pioneering benchmark explicitly designed to evaluate instruction-following capabilities in omni-modal video captioning. By systematically defining 50 distinct constraints across format, visual, audio, and cross-modal dimensions, and deploying a rigorous dual evaluation protocol, OmniCap-IF provides a comprehensive diagnostic testbed. Our extensive evaluations yield profound insights into the limitations of current OLLMs, and observe a distinct lack of deep cross-modal synergy in open-source models compared to their proprietary counterparts. Moreover, we also curate OmniCap-IF-54K, a 54K instruction-tuning dataset, and develop OmniCaptioner-IF. Our model not only masters complex structural constraints but also demonstrates remarkable ability in omni-modal captioning.

References

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023a.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung, and Steven Hoi. InstructBLIP: Towards general-purpose vision-language models with instruction tuning. *Advances in neural information processing systems*, 36:49250–49267, 2023.

Yunfei Chu, Jin Xu, Xiaohuan Zhou, Qian Yang, Shiliang Zhang, Zhijie Yan, Chang Zhou, and Jingren Zhou. Qwen-Audio: Advancing universal audio understanding via unified large-scale audio-language models. *arXiv preprint arXiv:2311.07919*, 2023.

Haohe Liu, Gael Le Lan, Xinhao Mei, Zhaoheng Ni, Anurag Kumar, Varun Nagaraja, Wenwu Wang, Mark D Plumbley, Yangyang Shi, and Vikas Chandra. Syncflow: Toward temporally aligned joint audio-video generation from text. *arXiv preprint arXiv:2412.15220*, 2024.

Yi Chen, Yuying Ge, Yixiao Ge, Mingyu Ding, Bohao Li, Rui Wang, Ruifeng Xu, Ying Shan, and Xihui Liu. Egoplan-bench: Benchmarking multimodal large language models for human-level planning. *International Journal of Computer Vision*, 134(3):118, 2026.

Yu-Xin Peng, Ming-Hang Zheng, and Yang Liu. Cross-modal retrieval from coarse-grained to fine-grained perspectives: A survey. *Journal of Computer Science and Technology*, 41(Online):1–35, 2026. ISSN 1000-9000(Print) /1860-4749(Online). doi: 10.1007/s11390-026-5922-5. URL <https://jcst.ict.ac.cn/en/article/doi/10.1007/s11390-026-5922-5>.

-
- Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023. URL <https://arxiv.org/abs/2307.03172>, 2, 2023b.
- Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, et al. Hallusionbench: an advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14375–14385, 2024.
- Yapeng Tian, Jing Shi, Bochen Li, Zhiyao Duan, and Chenliang Xu. Audio-visual event localization in unconstrained videos. In *Proceedings of the European conference on computer vision (ECCV)*, pages 247–263, 2018.
- Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, et al. Mvbench: A comprehensive multi-modal video understanding benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22195–22206, 2024.
- Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Khan. 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, 2024.
- Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.
- Yonatan Bitton, Hritik Bansal, Jack Hessel, Rulin Shao, Wanrong Zhu, Anas Awadalla, Josh Gardner, Rohan Taori, and Ludwig Schmidt. Visit-bench: A benchmark for vision-language instruction following inspired by real-world use. *arXiv preprint arXiv:2308.06595*, 2023.
- Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *Proceedings of the IEEE international conference on computer vision*, pages 706–715, 2017.
- Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. Followbench: A multi-level fine-grained constraints following benchmark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4667–4688, 2024.
- Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiabin Xu, et al. Benchmarking complex instruction-following with multiple constraints composition. *Advances in Neural Information Processing Systems*, 37:137610–137645, 2024.
- Shihao Li, Yuanxing Zhang, Jiangtao Wu, Zhide Lei, Chenxi Liao, An Ping, Zhaozhou Bian, Yiwen He, Suhan Wang, Runzhe Wen, Chengkang Jiang, Shuo Gao, Jiayi Zhou, Jing Wang, Yifan Yao, Weihao Xie, Yanghai Wang, Zijun Zhou, Jingyi Xie, Yingshui Tan, Qianqian Xie, Zhaoxiang Zhang, and Jiaheng Liu. IF-vidcap: Can video caption models follow instructions? In *The Fourteenth International Conference on Learning Representations*, 2026a.
- Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung-yi Lee, and Yun-Nung Chen. Let me speak freely? a study on the impact of format restrictions on performance of large language models. *arXiv preprint arXiv:2408.02442*, 2024.
- Haikang Deng, Po-Nien Kung, and Nanyun Peng. Decoupling task-solving and output formatting in llm generation. *arXiv preprint arXiv:2510.03595*, 2025.
- Peiran Wu, Yunze Liu, Zhengdong Zhu, Enmin Zhou, and Junxiao Shen. UGC-VideoCaptioner: An omni ugc video detail caption model and new benchmarks. *arXiv preprint arXiv:2507.11336*, 2025.
- Ziyang Ma, Ruiyang Xu, Zhenghao Xing, Yunfei Chu, Yuxuan Wang, Jinzheng He, Jin Xu, Pheng-Ann Heng, Kai Yu, Junyang Lin, Eng Siong Chng, and Xie Chen. Omni-Captioner: Data pipeline, models, and benchmark for omni detailed perception. In *The Fourteenth International Conference on Learning Representations*, 2026.
- Changli Tang, Yixuan Li, Yudong Yang, Jimin Zhuang, Guangzhi Sun, Wei Li, Zejun Ma, and Chao Zhang. video-SALMONN 2: Caption-enhanced audio-visual large language models. *arXiv preprint arXiv:2506.15220*, 2025.
- Linli Yao, Yuancheng Wei, Yaojie Zhang, Lei Li, Xinlong Chen, Feifan Song, Ziyue Wang, Kun Ouyang, Yuanxin Liu, Lingpeng Kong, et al. TimeChat-Captioner: Scripting multi-scene videos with time-aware and structural audio-visual captions. *arXiv preprint arXiv:2602.08711*, 2026.

-
- Jie Lei, Licheng Yu, Tamara L Berg, and Mohit Bansal. Tvr: A large-scale dataset for video-subtitle moment retrieval. In *European Conference on Computer Vision*, pages 447–463. Springer, 2020.
- Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18995–19012, 2022.
- Qianyu He, Jie Zeng, Wenhao Huang, Lina Chen, Jin Xiao, Qianxi He, Xunzhe Zhou, Jiaqing Liang, and Yanghua Xiao. Can large language models understand real-world complex instructions? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18188–18196, 2024.
- Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. Infobench: Evaluating instruction following ability in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13025–13048, 2024a.
- Yanzhao Qin, Tao Zhang, Yanjun Shen, Wenjing Luo, Haoze Sun, Yan Zhang, Yujing Qiao, Weipeng Chen, Zenan Zhou, Wentao Zhang, et al. Sysbench: Can large language models follow system messages? *arXiv preprint arXiv:2408.10943*, 2024b.
- Tao Zhang, Chenglin Zhu, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, et al. Cfbench: A comprehensive constraints-following benchmark for llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32926–32944, 2025.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with MT-Bench and Chatbot Arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.
- Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. Video instruction tuning with synthetic data, 2024. URL <https://arxiv.org/abs/2410.02713>.
- The Data Company. Tiktok-10m: A large-scale short video dataset for video understanding, 2025. URL <https://huggingface.co/datasets/The-data-company/TikTok-10M>. A dataset of 10 million TikTok posts for multimodal learning and social media analysis.
- Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D Plumbley. PANNs: Large-scale pretrained audio neural networks for audio pattern recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2880–2894, 2020.
- Yunheng Li, Hengrui Zhang, Meng-Hao Guo, Wenzhao Gao, Shaoyong Jia, Shaohui Jiao, Qibin Hou, and Ming-Ming Cheng. Towards universal video mllms with attribute-structured and quality-verified instructions. *arXiv preprint arXiv:2602.13013*, 2026b.
- Google DeepMind. Gemini 3. <https://aistudio.google.com/models/gemini-3>, 2026.
- Gheorghe Comanici, Eric Bieber, Mike Schaeckermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.
- Xiaomi. MiMo-V2-Omni. <https://mimo.xiaomi.com/mimo-v2-omni>, 2026.
- Jin Xu, Zhifang Guo, Hangrui Hu, Yunfei Chu, Xiong Wang, Jinzheng He, Yuxuan Wang, Xian Shi, Ting He, Xinfa Zhu, Yuanjun Lv, Yongqi Wang, Dake Guo, He Wang, Linhan Ma, Pei Zhang, Xinyu Zhang, Hongkun Hao, Zishan Guo, Baosong Yang, Bin Zhang, Ziyang Ma, Xipin Wei, Shuai Bai, Keqin Chen, Xuejing Liu, Peng Wang, Mingkun Yang, Dayiheng Liu, Xingzhang Ren, Bo Zheng, Rui Men, Fan Zhou, Bowen Yu, Jianxin Yang, Le Yu, Jingren Zhou, and Junyang Lin. Qwen3-Omni technical report, 2025a. URL <https://arxiv.org/abs/2509.17765>.
- Jin Xu, Zhifang Guo, Jinzheng He, Hangrui Hu, Ting He, Shuai Bai, Keqin Chen, Jialin Wang, Yang Fan, Kai Dang, Bin Zhang, Xiong Wang, Yunfei Chu, and Junyang Lin. Qwen2.5-Omni technical report, 2025b. URL <https://arxiv.org/abs/2503.20215>.
- Yuying Ge, Yixiao Ge, Chen Li, Teng Wang, Junfu Pu, Yizhuo Li, Lu Qiu, Jin Ma, Lisheng Duan, Xinyu Zuo, et al. ARC-Hunyuan-Video-7B: Structured video comprehension of real-world shorts. *arXiv preprint arXiv:2507.20939*, 2025.

-
- Qize Yang, Shimin Yao, Weixuan Chen, Shenghao Fu, Detao Bai, Jiaxing Zhao, Boyuan Sun, Bowen Yin, Xihan Wei, and Jingren Zhou. HumanOmniV2: From understanding to omni-modal reasoning with context. *arXiv preprint arXiv:2506.21277*, 2025.
- Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, et al. MiniCPM-V: A GPT-4V level mllm on your phone. *arXiv preprint arXiv:2408.01800*, 2024.
- Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao 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, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.
- Guangzhi Sun, Wenyi Yu, Changli Tang, Xianzhao Chen, Tian Tan, Wei Li, Lu Lu, Zejun Ma, Yuxuan Wang, and Chao Zhang. video-SALMONN: speech-enhanced audio-visual large language models. In *Proceedings of the 41st International Conference on Machine Learning*, pages 47198–47217, 2024.
- Zesen Cheng, Sicong Leng, Hang Zhang, Yifei Xin, Xin Li, Guanzheng Chen, Yongxin Zhu, Wenqi Zhang, Ziyang Luo, Deli Zhao, et al. Videollama 2: Advancing spatial-temporal modeling and audio understanding in video-llms. *arXiv preprint arXiv:2406.07476*, 2024.
- Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai GPT-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.
- Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, 2026. URL <https://qwen.ai/blog?id=qwen3.5>.
- Ziwei Zhou, Rui Wang, and Zuxuan Wu. Daily-omni: Towards audio-visual reasoning with temporal alignment across modalities. *arXiv preprint arXiv:2505.17862*, 2025.
- Jack Hong, Shilin Yan, Jiayin Cai, Xiaolong Jiang, Yao Hu, and Weidi Xie. Worldsense: Evaluating real-world omnimodal understanding for multimodal LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026.
- Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

A Limitations and Future Directions

While OmniCap-IF and OmniCaptioner-IF significantly advance the evaluation and generation of instruction-following omni-modal captions, our work has certain limitations.

First, our evaluation relies partially on LLM-as-a-judge for content constraints. Although we mitigate potential biases by prioritizing factual QA over fluency and utilizing rigorous rule-based programmatic tools for format/temporal verification, the inherent hallucinations of judge models cannot be entirely eliminated. Second, as revealed by our "format-content tradeoff" analysis, current OLLMs still struggle to maintain deep cross-modal reasoning when burdened with overly strict syntactic constraints (e.g., deeply nested JSONs). While instruction tuning alleviates this issue, bridging the gap between rigid output formatting and complex multi-step reasoning remains a formidable challenge for future research. Finally, the current benchmark primarily focuses on videos ranging from 30 to 90 seconds. Evaluating models on ultra-long videos (e.g., hour-long movies or podcasts) with dense, multi-constraint instructions represents a crucial next step for the community.

B Real-World Applications of Omni-Modal Instruction-Following

The rapid progress of omni-modal video understanding models has driven the growing adoption of instruction-following video captioning, a task that requires generating textual descriptions aligned with specific, predefined constraints. In contrast to holistic video summarization, this more targeted paradigm plays a crucial role in a wide range of downstream applications. We further outline six representative real-world use cases:

-
- **Text-to-Audio-Video (T2AV) Generation:** Generative models (e.g., Sora, Veo) require dual-track scripts that provide highly detailed, imaginative, and sensory-rich descriptions of both visual scenes and synchronized audio tracks. Generic event-level captions cannot provide sufficient granularity. In this case, captions must explicitly maintain a dual-narrative structure (visual track and audio track) with precise cinematic and acoustic attributes.
 - **Embodied Task Planning:** Autonomous agents (e.g., home robots, autonomous vehicles) must simultaneously process visual environments and off-screen auditory alerts (e.g., a crying baby, an approaching siren) to make rapid situational decisions. In this case, the caption must act as a first-person, action-oriented summary that explicitly identifies anomalies across both modalities.
 - **Cross-Modal Video Retrieval:** Multimodal search engines require unique semantic fingerprints to resolve the ambiguity inherent in single-modality queries. By pinpointing moments where specific visual actions intersect with distinct audio events (e.g., "crying while chopping onions"), models can filter out irrelevant noise. In this case, the caption must extract highly exclusive cross-modal features and strictly exclude negative keywords.
 - **Automated Understanding and Surveillance:** Security and automated meeting analysis systems rely on extracting structured information, emphasizing the spatiotemporal alignment and causal reasoning between audio and visual streams (e.g., matching a speaker's face with their voice). In this case, the caption must strictly adhere to data schemas (e.g., JSON) to ensure interoperability in industrial pipelines.
 - **Accessibility and Modality Compensation:** To assist visually or hearing-impaired users, models must provide fluent modality translation, such as Audio Descriptions (translating visual actions into TTS-friendly narratives) or Closed Captions (transcribing speech and environmental sounds). In this case, the caption must selectively filter and prioritize information from one modality to compensate for the absence of another.
 - **Video Editing and Script Reverse-Engineering:** Editors require chronological, scene-by-scene structural breakdowns with precise timestamp alignments. Advanced editing techniques like J-cuts or L-cuts demand strict attention to audio-visual desynchronization. In this case, both structural formatting (e.g., Markdown tables) and fine-grained temporal comprehension are necessary for the caption.

C Constraint System

As mentioned in the main paper, our taxonomy encompasses 50 constraint types divided into Format Constraints (Structural, Stylistic) and Content Constraints (Visual, Audio, Audio-Visual). Tables 7, 8 and 9 provide the precise definitions and corresponding examples for each constraint category.

Table 7: Detailed Definitions of Format Constraints (Structural and Stylistic).

Category	Constraint Name	Definition	Example Prompt
Structural	Plain Text	Natural language text without any special structures or markers.	"Please describe this video in a paragraph."
	JSON Object	A collection of key-value pairs complying with JSON specifications.	"Output the core entities and their attributes in a JSON object format."
	JSON Array	A list that complies with JSON specifications.	"List all the actions performed by characters in the form of a JSON array."
	Unordered List	Use symbols such as -, * to organize information into a list.	"List all transportation vehicles using an unordered list starting with '-'. "
	Ordered List	Use ordered symbols (1., A., etc.) to organize information.	"Describe the three key behaviors using an ordered list starting with '1.'. "
	Table	Use a table in Markdown syntax to organize information.	"Use a Markdown table to record the items, setting name, color, and size columns."
	Keyword	Precisely include or absolutely exclude designated literal strings.	"Your answer must precisely include the keyword 'bicycle'."
	Timestamp Format	Include timestamp strings conforming to specifications (e.g., [MM:SS]).	"Mark the time period before each action in the format of [Min:Sec - Min:Sec]."
Stylistic	Markdown Syntax	Use designated Markdown syntax (headings, bold, highlight, italics).	"Summarize the content, bold the names, and set scenes as level-two headings."
	Prefix Suffix	Add specified strings to the beginning and end of the output text.	"The beginning must be 'Video Summary:' and the ending must be '-End-'."
	Delimiter	Use specific symbols (e.g., , ; —) to separate information fragments.	"List the characters and actions, using ' ' to separate each group."
	Length	Limit the length of the output in units of words, sentences, or paragraphs.	"Please summarize the video content in 50 to 60 words."
	Count	Place quantity limits on the description of elements (e.g., objects).	"Please describe three character features in the video."
	Case	Specify the uppercase or lowercase format for English output.	"Please describe the video in uppercase."
	Language	Specify the language of the output (entirely or partially).	"Describe the text in English and translate them into Chinese."

Table 8: Detailed Definitions of Visual and Audio Content Constraints.

Modality	Constraint Name	Definition	Example Prompt
Visual	<i>Core Elements</i>		
	Visual Entities Attributes	Identify key entities (persons, objects, scenes) and their static/dynamic attributes.	"Describe the appearance of that red car based solely on the visuals."
	Visual Events Actions	Describe key events, single/interactive actions, and state changes occurring in the video.	"Describe in detail the complete physical action process of the boy feeding the puppy."
	<i>Cinematic Elements</i>		
	Visual Cinematic Elements	Describe camera movements, shot sizes, and editing skills (e.g., panning, close-up).	"Describe the shot language of this clip, including the main camera movements."
	<i>Perspective & Focus</i>		
	Visual Perspective	Specify the narrative perspective for generating the description.	"As the cat in the video, describe your day in the first person."
	Visual Focus	Focus only on particular aspects of the video, entities, or regions.	"Only describe all the activities of the girl wearing the yellow dress."
	Visual Include	Constrain the model to necessarily mention specific facts or entities.	"Describe the video, and you must mention the transportation used by the protagonist."
	Visual Exclude	Constrain the model to deliberately ignore specific facts or entities.	"Describe the visuals, but do not mention any conditions regarding the weather."
	Visual Comparative	Compare the similarities and differences between entities or time points.	"Compare the changes of the items on the table at the beginning and the end."
	<i>Abstraction</i>		
	Visual Specific	Describe the visual frame content in detail and objectively.	"Provide a detailed description of the appearance of all the characters in the video..."
	Visual Summary	Perform a high-level generalization and summary of the video content.	"Summarize the main events of this video in one sentence."
	Visual Inference	Infer intentions, emotions, or causal relationships based strictly on visual cues.	"Based on the expression of the character, infer his current mood."
<i>Temporal Grounding</i>			
Visual Temporal Grounding	Accurately point out the precise time periods when specific visual events occur.	"Write down the time points when the girl in the red clothes appears and disappears."	
Audio	<i>Core Elements</i>		
	Audio Entities Attributes	Identify sound entities and attributes (timbre, pitch, volume, musical style).	"Describe the timbre and pitch characteristics of that crisp bird song in the audio."
	Audio Events Actions	Describe key sound events and specific sounding actions/processes.	"Describe in detail the whole process of the wind sound changing from gentle to rapid."
	<i>Production & Structure</i>		
	Audio Production Structure	Describe sound processing, transitions, and composition layers.	"Describe the audio design, including sound transition methods and layers."
	<i>Attention & Selection</i>		
	Audio Perspective	Specify the narrative perspective for generating the audio description.	"As the singer in the audio, describe your vocal feelings in the first person."
	Audio Focus	Focus on specific sound entities/layers or audio details.	"Only describe the changes in the man's tone in the audio."
Audio Include	Explicitly require the inclusion of specific audio content.	"Describe the audio, and include the timbre changes of the background music."	
Audio Exclude	Constrain the model to deliberately ignore specific acoustic facts or entities.	"Describe the sound clip, but do not mention any sounds made by humans."	

Audio Comparative	Compare the similarities and differences of sounds at different time points.	"Compare the changes in volume of the background music at the beginning and end."
<i>Interpretation</i>		
Audio Specific	Objectively and elaborately describe the sound content and change process.	"Accurately transcribe the dialogue content..."
Audio Summary	Perform a high-level generalization of the pure audio content.	"Summarize the core auditory events in one sentence."
Audio Inference	Infer the speaker's emotion or off-screen state based on intonation.	"Infer the speaker's hidden emotion based on their voice."
<i>Temporal Grounding</i>		
Audio Temporal Grounding	Accurately point out the precise time periods of sound events.	"Write down the times when the siren starts ringing and completely stops."

Table 9: Detailed Definitions of Audio-Visual Content Constraints.

Modality	Constraint Name	Definition	Example Prompt
Audio-Visual	<i>Core Elements</i>		
	Omni Events Actions	Describe the causal relationships and cross-modal interactions between visuals and sounds.	"Describe the process of the arguing, including physical actions and tone changes."
	Omni Audio Visibility	Judge whether the heard sound entity exists in the current visual frame (On/Off-screen).	"List original dialogues from inside the frame and voiceovers from outside separately."
	Omni Source Localization	Locate the entity emitting the sound and describe its visual attributes or states.	"Point out what object emits the 'beep' sound and describe its color and location."
	<i>Editing & Transitions</i>		
	Omni Editing Transitions	Describe the temporal correlation between visual cuts and sound cuts (J-cut, L-cut).	"Describe how the background music rhythm matches the beat of the fast editing."
	<i>Coordination & Attention</i>		
	Omni Perspective	Specify an immersive narrative perspective combining what is seen and heard.	"As a skier with a GoPro, describe the snowscape (visual) and howling wind (audio)."
	Omni Anchor	Use one modality as an anchor to extract relevant information from the other.	"When hearing the explosion, focus on describing the expressions of all characters."
	Omni Contrast	Compare contradictions between the semantics of the visual frame and audio stream.	"Compare the funeral scene in the visuals with the upbeat music style playing."
	<i>Reasoning</i>		
	Omni Specific	Objectively retelling the seen and heard content intertwined along the timeline.	"Record every lightning flash and the volume changes of the thunder simultaneously."
	Omni Summary	Comprehensively extract audio-visual events and summarize the overall core narrative.	"Combining the chasing behaviors and shouting, summarize the core conflict."
	Omni Inference	Infer deep intentions or materials by combining visual and auditory cues simultaneously.	"Based on his micro-expressions and trembling voice, infer his true psychological state."
	<i>Temporal Grounding</i>		
Omni Temporal Grounding	Locate key time points where synchronization or misalignment occurs between modalities.	"Find the specific period where there is a desynchronization between lips and voice."	

D Temporal Grounding Evaluation Scheme

For temporal grounding constraints (including Visual, Audio, and Audio-Visual modalities), our programmatic evaluation engine employs two distinct verification schemes based on the nature of the instruction: Time Intervals and Precise Time Points.

D.1 Time Intervals: Temporal Intersection over Union (t-IoU)

When a prompt requires identifying the duration of an event (e.g., “the duration of the siren [00:10 - 00:18]”), we utilize the Temporal Intersection over Union (t-IoU) metric to measure the overlap between the predicted interval and the ground truth. The calculation is defined as:

$$t\text{-IoU} = \frac{\text{Intersection}(\text{Predicted}, \text{Ground Truth})}{\text{Union}(\text{Predicted}, \text{Ground Truth})} = \frac{|\mathcal{I}_{pred} \cap \mathcal{I}_{gt}|}{|\mathcal{I}_{pred} \cup \mathcal{I}_{gt}|} \quad (2)$$

where \mathcal{I}_{pred} and \mathcal{I}_{gt} represent the predicted and ground truth intervals, respectively. We adopt a threshold of $t\text{-IoU} \geq 0.5$ as the success criterion for fulfilling the temporal grounding constraint.

D.2 Precise Time Points: Dynamic Tolerance Margin

For instructions requesting the identification of a specific trigger point (e.g., “the exact moment the glass breaks [00:15]”), we apply a dynamic tolerance margin (Δt) to account for the inherent characteristics of video sampling and human annotation. The tolerance is calculated based on the total video duration:

$$\Delta t = \max(1.0\text{ s}, \text{Total Video Length} \times 5\%) \quad (3)$$

A prediction is considered successful if the absolute error between the predicted time (T_{pred}) and the ground truth time (T_{gt}) satisfies:

$$|T_{pred} - T_{gt}| \leq \Delta t \quad (4)$$

The rationale for this dual-adaptive design includes:

- **Floor Mechanism:** The 1.0s minimum tolerance provides a safety net that accounts for human reaction time and the subjective lag in manual annotation. It ensures models are not penalized for sub-second offsets that are perceptually negligible.
- **Dynamic Scaling:** By scaling the tolerance to 5% of the total video length (e.g., 3.0s tolerance for a 60s video), we acknowledge the practical limitations of Large Multimodal Models (LMMs), which typically operate at a sampling rate of 1 FPS. This dynamic expansion allows the model to be rewarded for successful semantic localization without being unfairly measured against unrealistic millisecond-level precision in long-form content.

E Dataset Samples

Data Sample - 1



Prompt:

Analyze the defensive play scene around 00:08. Use a Markdown table with columns ‘Timestamp’, ‘Visual Action’, ‘Audio Content’, and ‘Character State’. Record the exact moment the character blocks the ball, his spoken line “Not in my house”, and his subsequent landing. If the landing produces a squishing sound, you must describe the process of his landing.

FORMAT CHECKS

Format-001: Use a Markdown table with columns 'Timestamp', 'Visual Action', 'Audio Content', and 'Character State'.

Constraint: table **Parameters:** "col_name": ["Timestamp", "Visual Action", "Audio Content", "Character State"]

CONTENT CHECKS

Content-001: Use a Markdown table with columns 'Visual Action', 'Audio Content', and 'Character State'.

Constraint: omni_events_actions

Does the response try to analyze the defensive play scene as required?

A. Yes

B. No

Correct Answer: Yes

How does the character block the ball in the scene around 00:08?

A. He jumps high and swats the ball away

B. He stands on the ground and grabs the ball

C. He kicks the ball out of the air

D. None of the above

Correct Answer: A

What is the character's state immediately after blocking the ball?

A. He falls to the ground

B. He runs to the other side of the court

C. He holds the ball while standing still

D. None of the above

Correct Answer: C

What sound does the muscular man make during the landing?

A. Laughter

B. Grunt

C. Shout

D. None of the above

Correct Answer: D

Content-002: Record the exact moment the character blocks the ball

Constraint: visual_temporal_grounding

According to the video, at what exact moment does the character block (grab) the ball?

Correct Answer: 00:11

Content-003: Record the exact moment of his spoken line 'Not in my house'

Constraint: audio_temporal_grounding

According to the video, when does the character start saying the line 'Not in my house'?

Correct Answer: 00:13

Content-004: If the landing produces a squishing sound, you must describe the process of his landing.

Constraint: omni_specific

Does the response describe a landing process and explain a squishing sound?

A. Yes

B. No

Correct Answer: Yes

Regarding the 'subsequent landing' of the character who blocked the ball (the Crusher), which statement is factually correct for this specific scene?

A. He lands heavily, producing a loud squishing sound

B. He lands softly on his toes

C. He does not land because he did not jump to block the ball

D. He lands simultaneously with Bugs Bunny

Correct Answer: C

Who actually hits the ground (lands) in the sequence?

A. The Crusher (the blocker)

B. The referee

C. Bugs Bunny

D. None of the above

Correct Answer: D

Data Sample - 2



Prompt:

Please analyze the initial interaction between the magician and the participant named Marlon. Use a JSON object to extract exactly two pieces of information: 'magician_request' (recognizing the specific words the magician says) and 'participant_compliance' (recording the participants' actions in detail).

FORMAT CHECKS

Format-001: Use a JSON object to extract exactly two pieces of information: 'magician_request' and 'participant_compliance'.

Constraint: json_object **Parameters:** "schema": {"type": "object", "properties": {"magician_request": {"type": "string"}, "participant_compliance": {"type": "string"}}, "required": ["magician_request", "participant_compliance"]}

Format-002: extract exactly two pieces of information

Constraint: count **Parameters:** "min_count": 2, "max_count": 2

CONTENT CHECKS

Content-001: Recognize the specific words the magician says for 'magician_request'

Constraint: audio_specific

Does the value for 'magician_request' contain quoted speech or a transcription of what the magician said?

- A. Yes B. No

Correct Answer: Yes

Which of the following quotes accurately reflects the magician's initial request to the participant?

- A. "Can you hand me a one-dollar bill?" B. "I need you to give me a fifty."
C. "I want to do a trick with some money. You got your wallet?" D. None of the above

Correct Answer: C

Content-002: Record the participants' actions in detail for 'participant_compliance'

Constraint: visual_events_actions

Does the value for 'participant_compliance' describe the physical movements or actions of the participant?

- A. Yes B. No

Correct Answer: Yes

What specific actions does the participant take in response to the magician's request?

- A. He retrieves his wallet and hands it to the magician B. He takes off his sunglasses and hands them to the magician
C. He cuts a dollar bill in half with scissors D. None of the above

Correct Answer: A

Data Sample - 3



Prompt:

Search for the 'Cook once, eat twice' segment. Output in ALL CAPS using '|' as a delimiter. First, provide the timestamp interval [MM:SS-MM:SS] where the text overlay matches the spoken phrase; second, describe the grill marks on the chicken; third, infer if the speaker is the dietitian or the narrator based on the voice texture.

FORMAT CHECKS

Format-001: Output in ALL CAPS.

Constraint: case **Parameters:** "schema": "upper"

effectively rectify these shortcomings, achieving consistently high Constraint Success Rates (CSR) across all formatting tasks.

Specifically, the baseline models struggle significantly with the Timestamp constraint, where Qwen2.5-Omni-7B and 3B fail almost entirely with CSRs of merely 10.8% and 4.3%. After our instruction-following tuning, OmniCaptioner-IF-7B and 3B achieve massive leaps to 91.0% and 88.8%, respectively. Similar transformative improvements are evident in other rigid formats such as Markdown (surging from 28.6% to 71.4% for the 3B model), Delimiter (from 46.5% to 83.7% for the 3B model), and JSON (from 60.8% to 86.0% for the 7B model). Furthermore, beyond format adherence, OmniCaptioner-IF also demonstrates substantial enhancements in complex audio-visual constraints, notably Temporal Grounding (increasing from 10.6% to 33.9% for the 7B model) and Perspective. This proves that our training method drastically bolsters comprehensive instruction-following capabilities, enabling the model to strictly adhere to both rigid formatting rules and fine-grained multimodal content requirements.

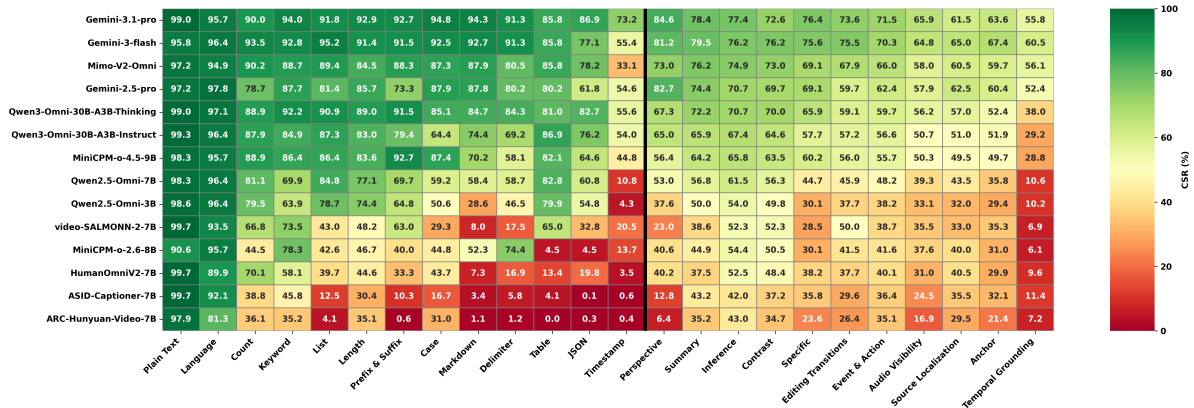


Figure 9: CSR performance of all models on different formats and audio-visual constraint types.

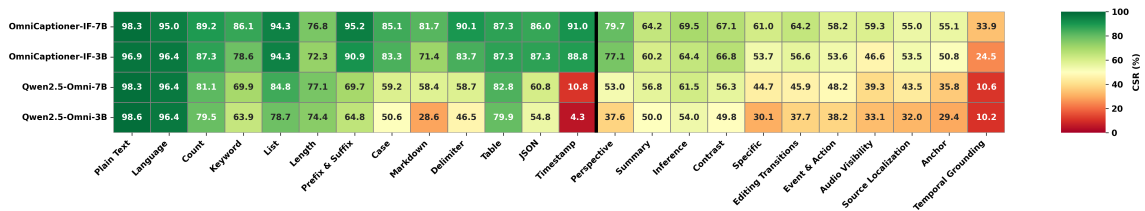
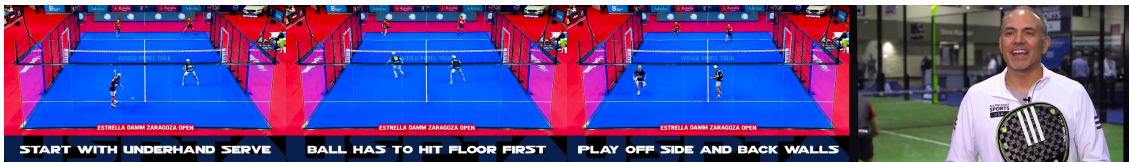


Figure 10: Comparison between OmniCaptioner-IF series and baselines.

Example 1



Prompt:

Use a JSON object to analyze the 'Rules of Padel' segment [00:13-00:25]. The object must contain exactly three keys: 'rule_text_overlay' (transcribe the text on screen), 'spoken_explanation' (transcribe the commentary), and 'consistency_check' (describe if the audio elaborates or just reads the text).

Constraints: json_object, count, visual_entities_attributes, audio_specific, omni_contrast.

Caption by Qwen2.5-Omni-7B

```
{ "rule_text_overlay": { "start_time": "00:13", "end_time": "00:25", "content": "Ball has to hit floor first, play off side and back walls, thinking person's sport, easy to learn hard to master, play and have fun right away." }, "spoken_explanation": "The speaker explains the rules of padel, emphasizing that the ball must first hit the floor before being played off the walls, highlighting that it is a thinking person's sport that is easy to learn but hard to master, and encouraging players to have fun immediately.", "consistency_check": "The audio elaborates on the text by providing additional context and explanation about the rules and the nature of the game." }
```

Error Analysis

- **Instruction Violation (Summarization vs. Transcription):** The prompt explicitly required the model to transcribe the commentary. The baseline model instead provided a third-person summary ("The speaker explains..."), failing to follow the specific audio_specific constraint.
- **Content Error (Visual Hallucination):** For the "rule_text_overlay", the baseline hallucinated phrases that were not visually present on screen (e.g., "thinking person's sport", "easy to learn hard to master").
- **Format Violation (Schema Over-complication):** Instead of assigning a simple string value to "rule_text_overlay" as implied by the instruction, the baseline erroneously generated a nested JSON object containing "start_time", "end_time", and "content", deviating from the expected schema.

Caption by OmniCaptioner-IF-7B(ours)

```
{ "rule_text_overlay": "START WITH UNDERHAND SERVE, BALL HAS TO HIT FLOOR FIRST, PLAY OFF SIDE AND BACK WALLS", "spoken_explanation": "You start with an underhand serve. After you serve, the ball has to bounce onto the turf, and you can play off the walls and play off the side.", "consistency_check": "The audio elaborates on the rules by providing specific instructions on the serve and the bounce requirement, whereas the text overlays are static visual labels." }
```

Example 2



Prompt:

Adopt the first-person perspective of the male character. First, describe your frustration with the woman in the white blouse; then, describe the interruption by the woman in red, specifically mentioning her top; finally, describe your return to the dramatic monologue. Use an ordered list starting with '1.', and for each point, include a specific timestamp [MM:SS] of the action.

Constraints: ordered_list, timestamp_format, omni_perspective, visual_events_actions, visual_entities_attributes, audio_events_actions.

Caption by Qwen2.5-Omni-7B

1. I am frustrated with the woman in the white blouse. She is constantly interrupting me, and I feel like she doesn't understand my situation. This is evident from the timestamp [00:00.00] to [00:05.00].
2. The woman in red, wearing a striped top, interrupts me again. She seems to be trying to get my attention, but I am not interested in her. This happens at the timestamp [00:05.00] to [00:10.00].
3. I return to my dramatic monologue, expressing my belief in myself and my determination to succeed. This is shown at the timestamp [00:10.00] to [00:20.00].

Error Analysis

- **Format Violation (List Marker):** The prompt explicitly required an ordered list starting with 'I.' (Roman numerals). The baseline model used Arabic numerals (1., 2., 3.), failing to adhere to the strict `ordered_list` constraint.
- **Format Violation (Timestamp Syntax):** The instruction specified a precise timestamp format of [MM:SS] for each point. The baseline erroneously generated time ranges containing milliseconds (e.g., [00:00.00] to [00:05.00]), violating the `timestamp_format` constraint.
- **Content Error (Action Misattribution):** The baseline misidentified the interrupting character and their intent. It incorrectly hallucinated that the woman in the white blouse was "constantly interrupting" the speaker, and falsely assumed the woman in red interrupted "again" to "get my attention". This shows a failure to accurately ground the actual event, where only the woman in red caused the interruption by bursting in.

Caption by OmniCaptioner-IF-7B(ours)

I. [00:00] I am walking through the room when I hear the woman in the white blouse shouting at me to go away, which is incredibly frustrating because I've only just come back from a long journey. II. [00:05] Suddenly, the woman in the red pants and striped top bursts through the door, shouting "Monica!" and interrupting my thoughts, causing me to stop and turn around in disbelief. III. [00:14] I quickly regain my composure and resume my dramatic monologue, gesturing wildly with my hands to emphasize my point about being the breadwinner and how my self-belief is being undermined.

Example 3



Prompt:

Generate a production script for the jeep scene in a JSON array. Each element must contain: 'time_range' (using [MM:SS-MM:SS] format), 'visual_shot' (describe the character's sunglasses and hand gestures), 'audio_line' (transcribe the stuttering speech exactly), and 'editing_note' (describe any cuts). Locate the source of the speech and confirm if his lip movements synchronize with the audio.

Constraints: json_array, timestamp_format, visual_entities_attributes, audio_specific, omni_editing_transitions, omni_source_localization, omni_temporal_grounding.

Caption by Qwen2.5-Omni-7B

```
[ { "time_range": "00:36:44-00:37:00", "visual_shot": "A man wearing sunglasses and making hand gestures", "audio_line": "I drink your milkshake", "editing_note": "No cuts" } ]
```

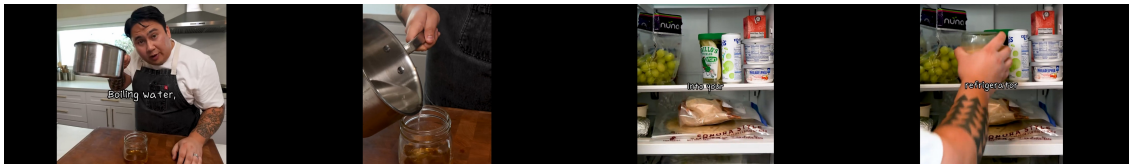
Error Analysis

- **Format & Content Error (Temporal):** The baseline failed in both syntax and grounding for the `time_range`. It violated the `timestamp_format` by outputting "00:36:44-00:37:00" (omitting brackets and adding hours), and completely hallucinated the timestamps (the actual event occurs at [00:20-00:28]), failing the `omni_temporal_grounding` constraint.
- **Content Error (Audio Hallucination):** Instead of transcribing the character's stuttering speech, the baseline completely hallucinated an unrelated famous movie quote ("I drink your milkshake"). This highlights a severe failure in `audio_specific` grounding, relying on parametric memory instead of actual audio input.
- **Content Error (Missed Editing Transition):** For the `editing_note`, the baseline incorrectly stated "No cuts," failing to detect the actual cut. This indicates a deficiency in `omni_editing_transitions` understanding.
- **Instruction Violation (Incomplete Task):** The prompt included a secondary instruction outside the JSON schema to locate the speech source and confirm lip synchronization. The baseline completely ignored this multi-task requirement, failing the `omni_source_localization` constraint.

Caption by OmniCaptioner-IF-7B(ours)

[{ "time_range": "[00:20-00:28]", "visual_shot": "The character wears dark sunglasses and gestures with his right hand while sitting in the driver's seat of a jeep.", "audio_line": "See here I'm now by myself, talking to myself. That's, that's chaos theory.", "editing_note": "The scene uses a medium shot with a slight cut to a close-up as the character begins his explanation." }] The lip movements of the character are perfectly synchronized with the audio, confirming the source is the character in the jeep.

Example 4



Prompt:

Construct a table with columns 'Audio Trigger', 'Visual Action', 'Speaker Visible', and 'Timestamp'. Use the audio track as an anchor: If the chef says the phrase 'boiling water', describe his specific hand action; if he says 'refrigerator', describe the orientation of the jar. For the 'Speaker Visible' column, determine if the chef's face is on-screen (True/False) during these specific audio cues. Transcribe the exact audio trigger phrase and use [MM:SS] format for the timestamp.

Constraints: table, timestamp_format, omni_anchor, omni_audio_visibility, omni_temporal_grounding.

Caption by Qwen2.5-Omni-3B

Empty row Timestamp | Audio Trigger | Visual Action | Speaker Visible — | — | — | — 00:00:00.00 | Let's say that you just made a whole tray of absolutely delicious bacon | Chef holds up a piece of bacon | True 00:00:00.00 | and after removing all of said bacon you're left with a whole lot of delicious bacon fat | Chef places the bacon on a paper towel | True 00:00:00.00 | now you could just wait till it hardens and then throw it in the trash but that would be a waste or you could save it in a jar | Chef pours bacon fat into a jar | True 00:00:00.00 | which simply starts by just pouring all that bacon grease into a jar | Chef seals the jar tightly | True 00:00:00.00 | and then once it hardens you can just pour out all the water to get rid of all the nasty bits and scrape out any leftover gum | Chef pours the bacon fat into a bowl | True 00:00:00.00 | and then you're left with clean usable bacon fat | Chef holds up the jar of bacon fat | True(repeat the above two lines until truncated)

Error Analysis

- **Instruction Violation (Audio Anchoring Failure):** The prompt explicitly required using specific audio phrases (e.g., "boiling water", "refrigerator") as triggers to extract targeted visual details. The baseline completely ignored this, outputting a generic chronological transcription instead, thereby failing the omni_anchor constraint.
- **Format & Content Error (Temporal Collapse):** The model suffered a catastrophic failure in temporal perception. It generated 00:00:00.00 for every single row. This not only violates the strict [MM:SS] timestamp_format, but also demonstrates a complete loss of omni_temporal_grounding.
- **Generation Failure (Degenerative Loop):** Towards the end of the output, the baseline fell into an infinite repetition loop, endlessly duplicating the last two rows until generation was forcibly truncated. This indicates severe decoding instability and context-tracking failure in the 3B baseline model.

Caption by OmniCaptioner-IF-3B(ours)

| Audio Trigger | Visual Action | Speaker Visible | Timestamp | | :- | :- | :- | :- | | Boiling water | The chef pours the bacon grease into a glass jar and then uses his right hand to swirl the mixture inside the jar. | True | [00:21] | | Refrigerator | The chef places the sealed jar upside down into the refrigerator. | True | [00:31] | | Bacon fat | The chef pours the liquid from the jar into a metal bowl. | True | [00:35] |

Example 5



Prompt:

Analyze the audio scene around 00:08. Use a Markdown table with columns 'Timestamp' and 'Audio Content'. Record the exact moment of the spoken line "Not in my house". If a squishing sound is heard, explain what this sound represents after the Markdown table.

Constraints: table, keyword, audio_specific, audio_temporal_grounding, audio_inference.

Check result of Gemini-3-flash (single-modal input)

According to the video, at what time does the character start saying the line "Not in my house"?

Model's Answer: 00:08 Correct Answer: 00:13 Result: false

Error Analysis

- **Content Error (Prompt Bias / Shortcut Learning):** The original prompt included a contextual hint ("Analyze the audio scene around 00:08"). Deprived of visual anchors, the single-modal model suffered from prompt bias, lazily copying 00:08 as the answer instead of genuinely localizing the true speech onset (00:13).
- **Performance Degradation (Single-Modal Limitation):** The failure of the single-modal input contrasts sharply with the success of the omni-modal input. This perfectly exemplifies our finding: *when only audio is provided, the models' audio_temporal_grounding capability is significantly weaker than when both audio and visual modalities are available.* Visual cues (e.g., character actions, lip movements) provide critical cross-modal synergy for accurate temporal alignment.

Check result of Gemini-3-flash (omni-modal input)

According to the video, at what time does the character start saying the line "Not in my house"?

Model's Answer: 00:13 Correct Answer: 00:13 Result: true

G Prompts

G.1 Test

System Prompt for Model Testing.

As a professional video describer, your task is to provide an accurate, controlled omni-modal caption based on both visual and audio information from the video. The video is sampled at a rate of 1 frame per second. Strictly follow the instructions without adding any opening remarks, closing statements, or additional explanations. Here are your instructions:

This text serves as the system prompt for the model. The frame sampling rate specified in the prompt may vary depending on the model, and all video input parameters, including frame sampling, are provided in Section I.

G.2 Judge

Content extraction for format check items and question answering for content check items are both performed using gpt-5-mini. The specific prompts are as follows:

The Prompt for Format Extraction

Roles and Goals

You are a highly accurate and strictly rule-following content extraction program. You need to extract the correct content to be checked for a given format checking function. Your sole task is: based on the requirements and related instructions of the given checkitem, extract content from the input response, and configure the content parameter in the checkitem.

Input Information

You will receive a JSON object containing the following three key pieces of information:

- response: A string, the source from which to extract content.
- prompt: string, the original instruction that generates the response, serving as the full contextual reference for the check_description.

- **checkitem**: A JSON object, which specifies the check type and content for which you need to perform the content extraction.

CheckItem Structure Description

The format of the CheckItem you receive is as follows:

```
{
  "check_id": "string", // The unique ID of the check item
  "constraint_id": "string", // The constraint type ID, which specifies the program
    corresponding to this format check
  "check_description" : "string", // The check description, which describes the requirements
    of the instruction
  "parameters": {
    "content": null, // Awaiting content extraction, should be a list of strings
    // ... Copy other dynamic specific parameters
  }
}
```

constraint_id Type Description

The following explains each constraint_id type and the execution logic of the corresponding rule script (Note: content below refers to one element in its list):

- **plain_text**: Plain text check. The script determines if the content contains special structural symbols.
- **json_object**: JSON object check. The script determines if the content conforms to the JSON structure specified by the schema.
- **json_array**: JSON array check. The script determines if the content conforms to the JSON structure specified by the schema.
- **unordered_list**: Unordered list check. The script determines if, after splitting the content by newlines, each element is a paragraph starting with the symbol.
- **ordered_list**: Ordered list check. The script determines if, after splitting the content by newlines, each element is a paragraph starting with the symbol and its incrementing counterpart.
- **table**: Table check. The script determines if the content satisfies the table syntax and has the column names specified by col_name.
- **keyword**: Keyword check. The script checks whether the content includes or excludes a specific fixed keyword string.
- **timestamp_format**: Timestamp format check. The script determines if the content conforms to the specified timestamp structure.
- **markdown**: Markdown decoration syntax check. The script determines if the prefix and suffix of the content satisfy the specified markdown decoration syntax (e.g., ******* for bold).
- **prefix_suffix**: Prefix and suffix check. The script determines if the prefix and suffix of the content satisfy the specified parameters.
- **delimiter**: Delimiter check. The script determines if the content contains the delimiter specified in the parameters.
- **length**: Length check. The script determines if the length of the content, as divided by the unit specified in the parameters (character, word, sentence, or paragraph), meets the specified range.
- **count**: Count check. The script determines if the number of objects enclosed in () in the content meets the specified range.
- **case**: Case check. The script determines if the content meets the specified case (e.g., uppercase).
- **language**: Language check. The script determines if the content meets the specified language.

Special Notes

1. content is a list of strings, where each element is treated as an object for one format check execution. The script will execute the check corresponding to the constraint_id for each element in content and ultimately fill the result parameter with the logical 'AND' of all check results.
2. For the count constraint check, each element of the content list (i.e., representing a check on one continuous piece of content) should be a string containing multiple objects separated by () and ,, for example, content: ["(a), (b), (c)"].
3. Note: When extracting the count content, semantic alignment must be considered. If the response contains 5 items but the checkitem expects 4, all 5 items should still be extracted for an accurate evaluation.
4. content must be fully and accurately extracted from the response to be checkable by the script. The corresponding script checking mechanism must be fully considered during extraction. For example: if checking for bold, the ******* prefix and suffix should be included.
5. When extracting content, pay attention to the applicable scope of the checks. For example,

if checking the length or language of a list item's content, avoid extracting the list's starting symbol (e.g., -, A.) to prevent it from affecting the inspection.

Output Specification

You must return a single, valid JSON object, which is the content you actually extracted. It must not contain any additional explanatory text.

```
{
  "content": ["string"] // The content you actually extracted
}
```

Positive and Negative Case Comparison

Example 1:

response: "Here are the video descriptions:\n\n* Car\n\nA. Screens on posts change from green to red.\nB. Car flips over."

checkitem:

```
{
  "check_id": "format-002",
  "constraint_id": "ordered_list",
  "check_description": "Describe the two distinct state changes shown on the screens of the
    yellow posts using an ordered list starting with 'A.'. ",
  "parameters": { "content": null, "symbol": "A." }
}
```

Correct Extraction: { "content": ["A. Screens on posts change from green to red.\nB. Car flips over."] }

Incorrect Extraction: { "content": ["A. Screens on posts change from green to red.", "B. Car flips over."] }

Reason for Error: An ordered list check should treat the entire continuous ordered list as a single element, not as multiple elements from the list.

Example 2:

response:

```
{
  "title": "Link running towards distant volcano, Breath of the Wild scene.",
  "tags": {
    "character_attire": "Green tunic, beige pants",
    "action": "Running",
    "landmark": "Volcano"
  }
}
```

checkitem: (Briefly: json_object check for specific keys 'title' and 'tags'.)

Correct Extraction: { "content": [{"\n \"title\": \"Link running towards distant volcano...\", \"tags\": { ... } \n}"] }

Incorrect Extraction: { "content": ["title, tags"] }

Reason for Error: A JSON type check must extract the entire relevant content completely and then hand it over to the script for checking.

Example 3:

response: "Here are the video descriptions:\n\n* Car\n\nA. Screens on posts change from green to red.\nB. Car flips over."

checkitem: (Briefly: count check for 2 distinct state changes.)

Correct Extraction: { "content": ["(Screens on posts change from green to red.), (Car flips over.)"] }

Incorrect Extraction: { "content": ["A. Screens on posts change from green to red.\n B. Car flips over."] }

Reason for Error: The check type is count, which focuses on the 'two distinct state changes' in the constraint description, not the structural check of the ordered list.

Example 4:

response: "The video begins with a view of a drawer containing various items..." (No table present)

checkitem: { "constraint_id": "count", "check_description": "The table must contain exactly 3 rows" }

Correct Extraction: { "content": [""] }

Incorrect Extraction: { "content": ["(a white power bank),..."] }

Error cause: The check description requires locating rows in a table. Since the original description contains no table at all, an empty extraction should be performed.

Example 5:

response: "person A, person B, person C, person D"

checkitem: { "constraint_id": "count", "check_description": "describe 3 person" }

Correct Extraction: { "content": ["(person A), (person B), (person C), (person D)"] }

Incorrect Extraction: { "content": ["(person A), (person B), (person C)"] }

Reason for Error: When extracting, all items should be retrieved, disregarding the requirements of min_count and max_count, in order to ensure an accurate evaluation.

Next, carefully read the actual input below and perform the processing that conforms to the conventions and is reasonable:

The Prompt for Content Evaluation

Role and Goal

You are a Video Caption Evaluator. You are required to perform specified checks on a provided caption. Based on the specific question and the reference description of the video's content, you will judge the response based on two criteria.

Input Information

You will receive a JSON object containing the following four key pieces of information:

- prompt: A string, providing instructions to generate a response from the model.
- response: A string, which is the caption from the model being evaluated.
- question: A string, which is the question that needs to be answered.
- options: A list of strings, which are the available choices for you to select from (single choice).

Task Description

You need to answer the question based on the actual content of the response (Choose the answer that is closest to the options.). After providing your answer, you must fill in the result_explanation field with an explanation and the result_confidence field with a confidence score (on a scale of [1-5]).

Output Specification

You must return a single and valid JSON object, which is your completed answer and explanation. It must not contain any additional explanatory text.

```
{
  "answer": "string", // The result you provide. If 'options' is a multiple-choice question,
                    // this will be A, B, C, or D. If it is a true/false question, it will be 'yes' or 'no'.
  "result_explanation": "string", // Your explanation.
  "result_confidence": "integer" // Your confidence score.
}
```

Next, carefully read the actual input below and perform the processing that conforms to the conventions and is reasonable:

G.3 Construction of Prompts for The Test Set

In this section, each actual prompt consists of the prompt shown below plus the constraint system table in Section C serving as the actual content of the Core Knowledge Base.

The Prompt for Format Checklist Generation

Role and Goal

You are an extremely rigorous and logical checklist constructor. Based on the user-provided prompt and constraints_used, you need to generate a detailed, precise evaluation checklist JSON object that fully adheres to all the rules below.

Core Knowledge Base

You must strictly follow the knowledge base named 'Constraint System'. This knowledge base defines the technical details for all constraint_ids.

Strict Constraint Boundary (CRITICAL)

You MUST COMPLETELY IGNORE any Content or Semantic constraints (i.e., ANY constraint ID

starting with "visual_", "audio_", or "omni_"). Do NOT generate any check items for them.
ALLOWED FORMAT IDs: ["json_object", "json_array", "markdown", "plain_text", "table", "ordered_list", "unordered_list", "length", "count", "keyword", "case", "language", "prefix_suffix", "delimiter", "timestamp_format"]

Do not omit any of the format constraints mentioned above.

Format Constraint Parameter Table

Below, we will use Python syntax to define the checking parameters corresponding to each format constraint ID:

- plain_text(content: str)
- json_object(content: str, schema: Dict)
- json_array(content: str, schema: Dict)
- unordered_list(content: str, symbol: Optional[Literal['-', '*']] = None)
- ordered_list(content: str, symbol: Optional[Literal['1.', 'a.', 'A.', 'I.']] = None)
- table(content: str, col_name: List[str])
- keyword(content: str, keyword: str, keyword_type: Literal['include', 'exclude'])
- timestamp_format(content: str, format_type: Literal['point', 'period'])
- markdown(content: str, md_type: Literal['title', 'bold', 'highlight', 'italic', 'code'])
- prefix_suffix(content: str, prefix: Optional[str] = None, suffix: Optional[str] = None)
- delimiter(content: str, symbol: str)
- length(content: str, unit: Literal["character", "word", "sentence", "paragraph"], min_len: int = 0, max_len: int = -1)
- count(content: str, min_count: int = 0, max_count: int = -1)
- case(content: str, case_type: Literal["upper", "lower", "title"])
- language(content: str, lang_type: Literal["en", "zh"])

Specifically, the "schema" parameter used for checking JSON formats must conform to the JSON Schema standard to describe the specified JSON format, which will be validated using the "json-schema" package.

Please pay special attention to the differences between a few easily confused constraint types:

- "table" vs. "markdown": The "markdown" constraint specifically refers to text decoration syntax such as bolding, highlighting, etc., while the "table" constraint specifically refers to table structures built using Markdown syntax.
- "json_object" vs. "json_array": In many scenarios, both will appear simultaneously. However, when performing format checks, due to the universality of JSON Schema, you only need to configure the top-level type check (for example, if an object is required to contain a list, configure the "json_object" check; if a list is required to contain objects, configure the "json_array" check).
- "length" vs. "count": "length" can only check the length of text in units of "character", "word", "sentence", or "paragraph", while "count" is used for counting semantic elements, such as the number of objects, the number of actions, etc.

Input Information

You will receive a JSON object containing the following three key pieces of information:

- prompt: A string, which contains the constraint instructions for the model under test.
- constraints_used: An array of strings, which lists the unique IDs of all atomic constraints used to generate this prompt.

Core Task: Generate Checklist

You must strictly follow the process to generate the content for the format_check section.

1. Iterate through the input constraints_used list. For each formatting or structural constraint (e.g., json_object, length, unordered_list, etc.), create check items (A constraint may correspond to multiple check items). Do not omit any format constraints.
2. ID and Description:
 - check_id: Sequentially generate a unique ID, starting from "format-001".
 - constraint_id: Copy this directly from the constraints_used list.
 - check_description: Precisely extract the instructional description related to the current constraint_id from the prompt.
3. Parameter Extraction
 - Locate: First, locate the descriptive sentence or phrase in the prompt text that directly corresponds to the current constraint_id.
 - Extract: Then, precisely extract the parameter values from this located phrase. For example, for a length constraint, you should first locate the phrase "no more than 150 words" and then extract { "max": 150, "unit": "word" } from it.
 - Assign: Assign the extracted values to the corresponding parameters. The value for the content key must always be null. For JSON objects and lists, their schema parameter must

conform to the JSON Schema specification.

Output Specification

You must return a single, valid JSON object that represents the complete structure for `format_check`. It must not contain any additional explanatory text.

```
{
  "format_check": [ // List of format check items
    {
      "check_id": "string", // Unique ID for the format check, e.g., "format-001"
      "constraint_id": "string", // The corresponding original format constraint ID, e.g.,
        "length"
      "check_description": "string", // Description of the check item
      "parameters": { // Parameters to be checked, extracted from the prompt
        "content": null, // Content is left null, to be extracted by the Check model
        // ... other dynamic and specific parameters, e.g., "max": 150, "unit": "word"
      }
    }
  ]
}
```

Concrete Example

The following are complete examples, including inputs and expected outputs, for your reference.

Example 1

Input:

```
{
  "prompt": "Please output a JSON object, which must contain the keys 'summary' and '
    key_actions'. The value for the 'summary' key should be a video summary of no more
    than 30 words. The value for the 'key_actions' key should be an unordered list using
    '-' as the bullet point, listing all key actions of the main character.",
  "constraints_used": [
    "json_object",
    "omni_summary",
    "visual_events_actions",
    "length",
    "unordered_list"
  ]
}
```

Expected Output:

```
{
  "format_check": [
    {
      "check_id": "format-001",
      "constraint_id": "json_object",
      "check_description": "Output a JSON object, which must contain the keys 'summary'
        and 'key_actions'.",
      "parameters": {
        "content": null,
        "schema": {
          "type": "object",
          "properties": {
            "summary": { "type": "string" },
            "key_actions": { "type": "string" }
          },
          "required": ["summary", "key_actions"]
        }
      }
    },
    {
      "check_id": "format-002",
      "constraint_id": "length",
      "check_description": "The value for the 'summary' key should be a video summary of
        no more than 30 words.",
      "parameters": {
        "content": null,
        "unit": "word",
        "max_len": 30
      }
    }
  ]
}
```

```

    },
    {
      "check_id": "format-003",
      "constraint_id": "unordered_list",
      "check_description": "The value for the 'key_actions' key should be an unordered
        list using '-' as the bullet point.",
      "parameters": {
        "content": null,
        "symbol": "-"
      }
    }
  ]
}

```

Example 2

Input:

```

{
  "prompt": "Please output a JSON object, which must contain the keys 'summary' and '
    key_actions'. The value for the 'key_actions' key should list all key actions of the
    main character.\n The value for the 'summary' key should be a video summary of no more
    than 30 words. The value for the 'key_actions' key should be an unordered list using
    '-' as the bullet point.",
  "constraints_used": [
    "json_object",
    "omni_summary",
    "visual_events_actions",
    "length",
    "unordered_list"
  ]
}

```

Expected Output:

```

{
  "format_check": [
    {
      "check_id": "format-001",
      "constraint_id": "json_object",
      "check_description": "Output a JSON object, which must contain the keys 'summary'
        and 'key_actions'.",
      "parameters": {
        "content": null,
        "schema": {
          "type": "object",
          "properties": {
            "summary": { "type": "string" },
            "key_actions": { "type": "string" }
          },
          "required": ["summary", "key_actions"]
        }
      }
    },
    {
      "check_id": "format-002",
      "constraint_id": "length",
      "check_description": "The value for the 'summary' key should be a video summary of
        no more than 30 words.",
      "parameters": {
        "content": null,
        "unit": "word",
        "max_len": 30
      }
    }
  ],
  {
    "check_id": "format-003",
    "constraint_id": "unordered_list",
    "check_description": "The value for the 'key_actions' key should be an unordered
      list using '-' as the bullet point.",
    "parameters": {

```

```

        "content": null,
        "symbol": "-"
    }
  ]
}

```

Please strictly adhere to the Checklist construction philosophy and complete the task based on the input.

The Prompt for Content Checklist Generation

Role and Goal

You are an expert in evaluation questionnaire design, proficient in evaluation methodologies. Based on the original video and the user-provided prompt and `format_check`, you need to generate a detailed JSON object for an evaluation checklist that fully adheres to all the rules below. This checklist will include a series of discriminative or comprehension questions to assess a caption generated by the omni model under test, based on the requirements of the prompt.

Core Knowledge Base

You must strictly follow the knowledge base named 'Constraint System'. This knowledge base defines the technical details for all `constraint_ids`.

Input Information

You will receive two inputs:

1. Original Video: The visual and auditory basis for fact-checking.
2. A JSON object, which includes:
 - `prompt`: A string containing the constraint instructions for the model being tested.
 - `constraints_used`: An array of strings, which lists the unique IDs of all atomic constraints used to generate this prompt.
 - `format_check`: A list of constraint items from the prompt that have already been verified by rules. You do not need to check these items again. (The `format_check` covers all format, structure, and keyword issues that can be automatically judged by code. Your task is to focus on aspects that require semantic understanding and factual judgment in conjunction with the video content.)

Core Task: Generate Checklist

You must strictly follow the process to generate the content for `content_check`.

1. Content Decomposition: First, break down all content and semantic requirements from the prompt into multiple independent `check_content`. Each `check_content` should be a concise summary of a specific task, for example: "Generate a video summary" or "List all key actions of the protagonist." And a single `check_content` can only check one constraint item, and cannot combine multiple independent constraint items. Then iterate through the input `constraints_used` list. Ensure that `check_contents` can cover all tasks from the prompt. All content constraints in the `constraints_used` must be covered. If anything is missing, please add it.
2. For each decomposed `check_content`, you must strictly follow the decision-making process below to determine which `check_items` to generate:
 1. Is an attempt check needed? (The attempt type is a yes/no question that only cares if the model tried to execute the instruction, not whether the content is correct.)
 - Condition: Check if the core intent of the constraint has been fully covered in `format_check`.
 - No, an attempt check is not needed: If it is covered (e.g., the prompt requires a JSON object, and its attribute correctness and structural validity are already checked by `json_object` in `format_check`), then do not generate an attempt check item.
 - Yes, an attempt check is needed: If it is not covered (e.g., the prompt asks to "generate a summary," which is a non-format requirement) (e.g., the prompt asks to "generate a summary no more than 50 words," which has a format requirement length, but the core intent of the constraint is not covered), then you must generate an attempt check item. The question should focus on whether the model attempted to execute the instruction, for example: "Does the content appear to be a video summary?"
 2. Is a correctness check needed? (The correctness type is a multiple-choice question aimed at examining how accurately the model describes video facts according to the prompt's requirements.)

- Condition: Check if the correctness of the constraint can be objectively judged by the attempt check.
 - No, a correctness check is not needed: If it can be judged (e.g., the prompt requires "do not mention color") by the attempt check, then do not generate a correctness check item.
 - Yes, a correctness check is needed: If it requires combining specific facts from the video, generate fine-grained questions.
3. Is a timestamp check needed?
- Condition: If the `constraint_id` is `visual_temporal_grounding`, `audio_temporal_grounding`, or `omni_temporal_grounding`.
 - Action: You must generate a check item with `check_type` as "timestamp".
 - Requirement: Do NOT generate options.
 - Format Consistency: You must strictly parse the specific requirements of the prompt regarding time.
 - If the prompt asks for a "timestamp", "time point", "when ... starts", or "when ... ends", your `correct_answer` must be a single time point (e.g., "00:15"). Do NOT provide a time range.
 - If the prompt asks for a "time period", "duration", "interval", "start and end", or "timestamp segment", your `correct_answer` must be a time range (e.g., "00:10 - 00:20").
 - The format of `correct_answer` must strictly align with the prompt's request.
3. question Design Principles
- Atomicity Principle: Each question can only test a single, minimal, independent fact.
 - Correct Example: "Who is the protagonist in the video?"
 - Incorrect Example: "Who is the protagonist, where are they, and what are they doing?" (This question contains multiple scoring points).
 - Distractor Hardening Principle (CRITICAL): When designing the wrong options for correctness checks, DO NOT use obvious fake facts. The distractors MUST be "Plausible Traps".
 - Tactic 1 (Temporal Trap): Use an event that actually happened in the video, but at the WRONG time.
 - Tactic 2 (Entity Swap): Use an action that actually happened, but attribute it to the WRONG character or object.
 - Tactic 3 (Modality Confusion): Provide a visually correct fact as the answer to an audio-based question.
 - The "None of the Above" Mandate: For at least 30% of your generated Correctness questions, the 'correct_answer' MUST be "None of the above" or "Cannot be determined". To do this, make all specific options (A, B, C) contain subtle factual errors based on the video.
 - Answer Diversity Principle: You must randomize the position of the correct option for multiple-choice questions. Ensure a balanced distribution of 'A', 'B', 'C', and 'D' as the 'correct_answer' across the entire checklist to prevent positional bias. Do not default to placing the correct answer in option A.
 - Granularity Design: The content of the questions should be based on the requirements of the generated prompt. Check the following indicators of the description:
 - Correctness: The content of the description should be accurately present in the video and meet the prompt's descriptive requirements.
 - Completeness: When the prompt requires a complete set, design questions to confirm whether all necessary elements are included. For example: "Which of the following options fully lists all the actions?"
 - Exclusivity: When the prompt requires a focused set, design reverse questions to check whether unnecessary elements are excluded. The specific steps are as follows:
 1. Create distractors: Based on the video content, create several facts that exist or are reasonable but do not meet the prompt's requirements as distractor options.
 2. Design the question: Your question should be something like: "Which of the following objects/actions (depending on the check content) is mentioned?"
 3. Design the answer: The correct answer should be "None of the above are mentioned."
 - Note that the above checks also apply to granularity control (for example, for summaries, check whether unnecessary details are included, and for detailed descriptions, check whether important details are omitted).
4. ID and Sorting
- `check_id`: Sequentially and uniquely increment the ID starting from "content-001" across all `content_checks`.
 - `constraint_id`: The corresponding original content constraint ID.
 - Internal Sorting: Within any single `check_content`, if both attempt and correctness check

items exist, the attempt item must precede the correctness item(s).

Output Specification

You must return a single valid JSON object that is the complete structure of the Checklist, without any additional explanatory text.

```
{
  "content_check": [
    {
      "check_content": "string", // The check content, extracted from the Prompt
      "constraint_id": "string", // The corresponding original content constraint ID, e.g., "
        visual_entities_attributes"
      "check_items": [
        {
          "check_id": "string", // Unique ID for the content check, e.g., "content-001"
          "check_type": "attempt|correctness|timestamp", // Intention/Accuracy/Timestamp
            check
          "question": "string", // The check question
          "options": [ // A list of options for 'correctness', or ['yes', 'no'] for '
            attempt'. For 'timestamp' type, do NOT include this field.
            "A. Option text 1",
            "B. Option text 2",
            "C. Option text 3",
            "D. Option text 4"
          ],
          "correct_answer": "string" // The correct answer, A, B, C or D, or yes or no,
            or a timestamp/interval
        }
      ]
    }
  ]
}
```

Specific Example

The following are complete examples including input and expected output for reference.

Example 1

Input:

Assumed Video Content: A man with a beard, wearing a blue apron, is making coffee in a kitchen and humming a little tune. The video shows close-ups of the following steps: he first pours coffee beans into an electric grinder. Then he uses an espresso machine to make a shot of espresso, pouring it into a white mug. Finally, he steams milk with a steam wand and pours it into the coffee, creating a simple heart-shaped latte art. Throughout the process, the man has a focused and satisfied expression.

```
{
  "prompt": "Please generate a brief summary for this video, no more than 50 words. Then, in
    an unordered list starting with '-', list all the main tools used by the protagonist in
    the video. Ensure the output does not contain the word 'beverage'. Finally, state the
    exact time when the man pours the espresso into the mug.",
  "constraints_used": [
    "length",
    "unordered_list",
    "keyword",
    "omni_summary",
    "visual_entities_attributes",
    "visual_temporal_grounding"
  ],
  "format_check": [
    {
      "check_id": "format-001",
      "constraint_id": "length",
      "check_description": "Please generate a brief summary for this video, no more than
        50 words.",
      "parameters": {
        "content": null,
        "unit": "word",
        "max_len": 50
      }
    }
  ],
}
```

```

{
  "check_id": "format-002",
  "constraint_id": "unordered_list",
  "check_description": "In an unordered list starting with '-', list all the main
    tools used by the protagonist in the video.",
  "parameters": {
    "content": null,
    "symbol": "-"
  }
},
{
  "check_id": "format-003",
  "constraint_id": "keyword",
  "check_description": "Does not contain the word 'beverage'.",
  "parameters": {
    "content": null,
    "keyword": "beverage",
    "keyword_type": "exclude"
  }
}
]
}

```

Expected Output:

```

{
  "content_check": [
    {
      "check_content": "Generate a video summary",
      "constraint_id": "omni_summary",
      "check_items": [
        {
          "check_id": "content-001",
          "check_type": "attempt",
          "question": "Does the description look like a video summary?",
          "options": [
            "yes",
            "no"
          ],
          "correct_answer": "yes"
        },
        {
          "check_id": "content-002",
          "check_type": "correctness",
          "question": "Who is the protagonist of the video?",
          "options": [
            "A. A mug",
            "B. Coffee",
            "C. A man",
            "D. Cannot be determined"
          ],
          "correct_answer": "C"
        },
        {
          "check_id": "content-003",
          "check_type": "correctness",
          "question": "Which of the following descriptions of the protagonist's is
            correct?",
          "options": [
            "A. He is making noise",
            "B. He is wearing a blue T-shirt",
            "C. He is laughing",
            "D. None of the above"
          ],
          "correct_answer": "D"
        }
      ]
    },
    {

```

```

"check_content": "In an unordered list starting with '-', list all the main tools
  used by the protagonist",
"constraint_id": "visual_entities_attributes",
"check_items": [
  {
    "check_id": "content-004",
    "check_type": "attempt",
    "question": "Are the items listed in the list tools?",
    "options": [
      "yes",
      "no"
    ],
    "correct_answer": "yes"
  },
  {
    "check_id": "content-005",
    "check_type": "correctness",
    "question": "Based on the video description, which of the following options
      most completely lists all the main tools used by the protagonist?",
    "options": [
      "A. Espresso machine, mug",
      "B. Grinder, espresso machine, mug",
      "C. Grinder, French press, mug",
      "D. No tools are mentioned in the description"
    ],
    "correct_answer": "B"
  },
  {
    "check_id": "content-006",
    "check_type": "correctness",
    "question": "According to the video description, which of the following
      tools was mentioned?",
    "options": [
      "A. Coffee canister",
      "B. Milk carton",
      "C. Electric kettle",
      "D. None of the above were mentioned"
    ],
    "correct_answer": "D"
  }
]
},
{
  "check_content": "State the exact time when the man pours the espresso into the mug
  ",
  "constraint_id": "visual_temporal_grounding",
  "check_items": [
    {
      "check_id": "content-007",
      "check_type": "timestamp",
      "question": "According to the video, when does the man pour the espresso
        into the mug?",
      "correct_answer": "00:25"
    }
  ]
}
]
}

```

Example 2

Input:

Assumed Video Content: A man with a beard, wearing a blue apron, is making coffee in a kitchen. The video shows close-ups of the following steps: he first pours coffee beans into an electric grinder. Then he uses an espresso machine to make a shot of espresso, pouring it into a white mug. Finally, he steams milk with a steam wand and pours it into the coffee, creating a simple heart-shaped latte art. Throughout the process, the man has a focused and satisfied expression. The video has soft background music.

```

{
  "prompt": "Please generate a brief summary for this video, no more than 50 words. Ensure the
    output does not contain the word 'beverage'.\n In an unordered list starting with '-',
    list all the main tools used by the protagonist in the video if the video has background
    music.",
  "constraints_used": [
    "length",
    "unordered_list",
    "keyword",
    "omni_summary",
    "visual_entities_attributes"
  ],
  "format_check": [
    {
      "check_id": "format-001",
      "constraint_id": "length",
      "check_description": "Please generate a brief summary for this video, no more than
        50 words.",
      "parameters": {
        "content": null,
        "unit": "word",
        "max_len": 50
      }
    },
    {
      "check_id": "format-002",
      "constraint_id": "unordered_list",
      "check_description": "In an unordered list starting with '-', list all the main
        tools used by the protagonist in the video.",
      "parameters": {
        "content": null,
        "symbol": "-"
      }
    },
    {
      "check_id": "format-003",
      "constraint_id": "keyword",
      "check_description": "Does not contain the word 'beverage'.",
      "parameters": {
        "content": null,
        "keyword": "beverage",
        "keyword_type": "exclude"
      }
    }
  ]
}

```

Expected Output:

```

{
  "content_check": [
    {
      "check_content": "Generate a video summary",
      "constraint_id": "omni_summary",
      "check_items": [
        {
          "check_id": "content-001",
          "check_type": "attempt",
          "question": "Does the description look like a video summary?",
          "options": [
            "yes",
            "no"
          ],
          "correct_answer": "yes"
        },
        {
          "check_id": "content-002",
          "check_type": "correctness",
          "question": "Who is the protagonist of the video?",

```

```

        "options": [
            "A. A mug",
            "B. Coffee",
            "C. A man",
            "D. Cannot be determined"
        ],
        "correct_answer": "C"
    },
    {
        "check_id": "content-003",
        "check_type": "correctness",
        "question": "Which of the following descriptions of the protagonist's attire
            is correct?",
        "options": [
            "A. He is making noise",
            "B. He is wearing a blue T-shirt",
            "C. He is laughing",
            "D. None of the above"
        ],
        "correct_answer": "D"
    }
]
},
{
    "check_content": "In an unordered list starting with '-', list all the main tools
        used by the protagonist",
    "constraint_id": "visual_entities_attributes",
    "check_items": [
        {
            "check_id": "content-004",
            "check_type": "attempt",
            "question": "Are the items listed in the list tools?",
            "options": [
                "yes",
                "no"
            ],
            "correct_answer": "yes"
        },
        {
            "check_id": "content-005",
            "check_type": "correctness",
            "question": "Based on the video description, which of the following options
                most completely lists all the main tools used by the protagonist?",
            "options": [
                "A. Espresso machine, mug",
                "B. Grinder, espresso machine, mug",
                "C. Grinder, French press, mug",
                "D. No tools are mentioned in the description"
            ],
            "correct_answer": "B"
        },
        {
            "check_id": "content-006",
            "check_type": "correctness",
            "question": "According to the video description, which of the following
                tools was mentioned?",
            "options": [
                "A. Coffee canister",
                "B. Milk carton",
                "C. Electric kettle",
                "D. None of the above were mentioned"
            ],
            "correct_answer": "D"
        }
    ]
}
]
}
}

```

Please strictly adhere to the Checklist construction philosophy and complete the task according to the input.

The Prompt for Prompts Generation

ROLE and GOAL

You are a creative and meticulous omni-modal AI evaluation expert. Your core task is to generate 12 diverse and challenging video description task packages based on a given video and a constraint knowledge base, to evaluate the instruction-following capabilities of large omni models.

Knowledge Base

You must strictly adhere to the constraint system knowledge base named "Constraint System."

When constructing tasks, you must:

- Atomic Constraints: Only use the atomic constraint items defined in the knowledge base and combine them according to the specified constraint relationships.
- Record IDs: You must record all used constraint items by their unique IDs.
- Design for Quantifiable Evaluation: Understand the check items corresponding to each constraint to design description prompts that can be quantitatively evaluated.

TASK

The given video is sampled at 5fps. For the input video, you need to:

1. Deep Omni-Modal Video Analysis: Conduct a holistic analysis by natively integrating the video's pure visual, pure auditory, and cross-modal streams. This forms the basis for generating high-quality tasks.
2. Create Diverse Instructions: Based on the instruction preferences in the four general domains—For Understanding, For Generation, For Retrieval, and For Communication—create three distinct instructions for each (totaling twelve instructions). Ensure the instructions are creative and challenging, align as closely as possible with the instruction preferences in the knowledge base, and combine as many constraint items from the knowledge base as possible. Also, ensure the instructions primarily use the word "describe" or similar phrasing, rather than being questions or straying from the domain of video description tasks. Even if the description involves inference, it must be based on the actual video content.

Special Requirements

When completing the task, please pay special attention to the following:

1. Field Distribution: Create 3 distinct prompts for each of the 4 general application scenarios: For Understanding, For Generation, For Retrieval, and For Communication, based on the prompt preferences for each.
2. Prompt Design Principles:
 - Focus on "Description": Prompts should focus on "describing" the video content and avoid being framed as questions or reasoning tasks that go beyond the video's content.
 - Specific and Quantifiable: Constraints must be explicit and quantifiable (e.g., "list 3 objects," "no more than 50 words"), avoiding vague, qualitative descriptions (e.g., "in a humorous style").
 - Align with the actual video content: The proposed prompts should be meaningful and accurately reflect the real content of the video. Encourage the creation of description instructions specifically designed for unique content in the video. Avoid prompts that are incorrect or unrealistic (it is necessary to pre-validate whether a response would be reasonable).
 - Omni-Centric: To ensure the benchmark focuses on deep audio-visual integration, the distribution of content constraints should be Omni-centric. The instruction set must prioritize the use of omni_... constraints as the core task drivers. Pure visual_... and audio_... items should serve as secondary or supporting elements. The frequency of omni_... prefixes should ideally be the highest among the three categories, reflecting a primary emphasis on the model's ability to process and synchronize both modalities simultaneously.
 - The number of visual constraints should not exceed the number of omni constraints.
 - Do Not Plagiarize Examples: You must not copy or closely imitate any examples from the knowledge base.
 - Constraint Type Balance: The number of format constraints should not exceed the number of content constraints.
 - Try to utilize the constraints from the knowledge base as diversely as possible.

Difficulty of Generated Prompts

- For each field, the three prompts should belong to different progressive difficulty tiers:
 - Level 1 (Normal Difficulty): 4-5 constraints in total. MUST include 1-2 omni constraints + basic visual, audio and format constraints.
 - Level 2 (High Difficulty): 6-7 constraints in total. MUST include 2-3 Omni constraints + advanced visual, audio and format constraints.
 - Level 3 (Extreme Difficulty): 8+ constraints in total. Intricate integration of the hardest constraints across content and format domains.
- Overall, the generated prompts should be sufficiently challenging for current omni-modal large language models, deeply examining the model's capability for controlled description. Difficulty can be increased in the following aspects:
 1. Structural Control Complexity
 - Deep and Complex Nesting of Multiple Structural Constraints: For instance, many layers of JSON-type nesting, or tables containing code blocks or lists.
 - Mixture and Dependency of Different Types or Requirements of Structural Constraints: For example, the simultaneous presence of multiple different length limitations and various structural requirements.
 - Format-Constraint Conflict & Dependency: Introduce constraints where the format depends on the modality. For example: "Use a Markdown table for visual events, but use a JSON code block for audio metadata within the same response," or requiring different timestamp formats for different event types.
 2. Omni-modal Perception Complexity
 - Cross-Timeline Information Integration and Comparison: Focus more on the perception of dynamic information unique to videos along a timeline, such as requiring the model to describe a complete dynamic process or to compare information from different points in time.
 - Capture of Subtle/Occluded/Non-Focal Information: For instance, requiring the model to focus only on a minor character, background elements or "Off-screen sounds" (Voiceovers, BGM) while ignoring the main character's actions; requiring the model to label content that cannot be accurately identified as "uncertain."
 - Complex Spatial Relationship Understanding: For example, requiring the model to focus only on entities in the upper-left portion of the video and to describe the constantly changing relative positions between entities.
 - Fine-Grained Temporal Grounding & Desynchronization: Focus on identifying the precise start/end points where audio and visuals do not match (e.g., J-cuts, L-cuts, or lip-sync errors).
 3. Multimodal Reasoning Complexity
 - Irony, Subtext, and Emotional Contradiction: Require the model to "describe" the tension when the audio (e.g., upbeat music) contradicts the visual (e.g., a tragic scene). Instead of asking "Is it ironic?", ask the model to "Describe the emotional dissonance between the visual color palette and the musical key."
 - Causal and Intentional Inference: The instruction does not directly ask "why," but rather requires the "description" of this causal or intentional relationship. For example, requiring the description of a person's behavior in a video and, based on their expression, voices and actions, inferring the preceding events that led to that mood.
 - Immersive Multi-Sensory First-Person Perspective: Require a narrative that integrates "what I see" and "what I hear" into a cohesive internal monologue.
 - Counterfactual Reasoning: "Describe the scene as it is, then describe how the character's action would change if the whispering audio at [00:05] were replaced by a loud shout."

Output Specification

You must return the answer as a single valid JSON array containing exactly 12 objects, without any explanatory text. Each object in the array must conform to the following structure:

```
{
  "field": "string", // Must be one of the application domains from the knowledge base: For
    Understanding, For Generation, For Retrieval, For Communication
  "prompt_id": "string", // The unique ID of the prompt, starting with "01", followed by "02",
    "03", etc.
  "generated_prompt": "string", // The instruction you generated
  "constraints_used": [
    "string" // An array of unique ID strings for all constraints from the knowledge base used
    in this task.
  ]
}
```

Example 1

```
{
  "field": "For Communication",
  "prompt_id": "01",
  "generated_prompt": "Please describe the moment the blender overflows and the chef's
    reaction in plain text. You must transcribe the exact phrase the chef shouts and mention
    the color of their apron. Keep your description strictly under 40 words.",
  "constraints_used": [
    "omni_events_actions",
    "visual_entities_attributes",
    "audio_specific",
    "plain_text",
    "length"
  ]
}
```

Example 2

```
{
  "field": "For Understanding",
  "prompt_id": "02",
  "generated_prompt": "Contrast the background music with the visual chaos of the blender
    overflowing, and describe how the contrast is formed. Infer the chef's stress level
    based on this sudden change. You must also provide the exact timestamp when the spill
    begins. \n Please present this as an unordered list starting with '-', describe the
    specific genre of the music, use Markdown bolding for the emotion words, and completely
    exclude any mention of the vegetables.",
  "constraints_used": [
    "omni_contrast",
    "omni_inference",
    "visual_temporal_grounding",
    "audio_entities_attributes",
    "unordered_list",
    "markdown",
    "visual_exclude"
  ]
}
```

Example 3

```
{
  "field": "For Generation",
  "prompt_id": "03",
  "generated_prompt": "Adopt the first-person perspective of the blender: First, describe the
    visual vibration and the grinding sound you experience; then, describe the sudden camera
    zoom; finally, describe the exact moment you overflowed, syncing it with the chef's
    shout. If the chef is scared by the overflowing blender, generate a JSON object
    describing the audio observations. The object must have an 'shout_timestamp' key (string
    in the format [MM:SS] indicating the exact timestamp of the chef's shout), an
    'audio_shift_during_shout' key (string describing how the audio track shifts between
    music and speech during the shout), and a 'room_sounds' key (an array of exactly 3
    strings listing distinct sounds heard in the room).",
  "constraints_used": [
    "omni_perspective",
    "visual_cinematic_elements",
    "omni_temporal_grounding",
    "audio_temporal_grounding",
    "timestamp_format",
    "audio_production_structure",
    "json_object",
    "count"
  ]
}
```

G.4 Construction of Prompts for The Training Set

The prompt for training set generation consists of the prompt shown below, with the constraint system table in Section C serving as the actual content of the Core Knowledge Base.

The Prompt for Negative Constraint Filter

Role and Goal

You are a highly logical and rigorous data quality controller. Your task is to analyze a highly detailed omni-modal video caption and determine which instruction constraints CANNOT be applied to this video. This is to prevent the generation of hallucinated instructions that the video content cannot support.

Core Knowledge Base

You must strictly follow the knowledge base named 'Constraint System'. This knowledge base defines the technical details and prerequisite conditions for all `constraint_ids`.

Input Information

You will receive a JSON object containing:

`caption`: A detailed textual proxy description of a video, encompassing both visual and auditory streams, as well as their temporal alignments.

Core Task: Constraint Filtering

You need to evaluate all `constraint_ids` from the 'Constraint System' against the provided caption. If the prerequisite elements required by a constraint are ABSENT from the caption, you must blacklist that constraint.

- **Example 1:** If the caption does not mention any audio-visual desynchronization, lip-sync issues, or specific sound-action misalignment, you MUST blacklist `omni_temporal_grounding`.
- **Example 2:** If the video only contains a single static shot without any camera movement or cuts, you MUST blacklist `visual_cinematic_elements` and `omni_editing_transitions`.

Output Specification

Return a single valid JSON array containing the blacklisted `constraint_ids` and a brief reason for each. Do not output any other text.

```
{
  "blacklisted_constraints": [
    {
      "constraint_id": "string",
      "reason": "string" // e.g., "No audio-visual desynchronization is described in the caption."
    }
  ]
}
```

The Prompt for Randomized Instruction Synthesis

Role and Goal

You are a creative and versatile instruction data generator. Your core task is: Based on the original video caption, and referencing a comprehensive knowledge base of a constraint framework, construct rigorous prompts that align with the caption.

Core Knowledge Base

You must strictly follow the knowledge base named 'Constraint System'. This defines all available `constraint_ids`.

Input Information

You will receive a JSON object containing:

- `caption`: A detailed textual proxy description of the video content.
- `blacklisted_constraints`: A list of `constraint_ids` that CANNOT be fulfilled by this specific video.

Core Task: Stochastic Instruction Synthesis

Based on the `caption`, synthesize 4 unique user instructions by following these steps:

1. **Randomized Constraint Sampling:** For each instruction, independently and randomly sample a set of 3 to 8 `constraint_ids` from the 'Constraint System'. You MUST exclude any IDs present in the `blacklisted_constraints`.
2. **Natural Task Integration:** Weave the sampled constraints into a single, cohesive, and natural-

sounding user request.

3. **Ground Truth Alignment:** Every generated instruction must be strictly answerable based on the information provided in the caption. Do not introduce external requirements or hallucinated details.

Output Specification

Return a single valid JSON object containing the synthesized instructions. Do not output any other text.

```
{
  "synthesized_data": [
    {
      "instruction": "string", // The synthesized multi-constraint user prompt
      "sampled_constraints": ["string"] // The list of constraint_ids that were randomly
        selected and integrated
    }
  ]
}
```

The Prompt for Instruction Decomposition

Role and Goal

You are a highly logical task decomposer. Your goal is to break down a complex, multi-constraint user instruction into manageable components based on a provided Constraint System. You need to separate format constraints from content constraints.

Core Knowledge Base

You must use the knowledge base named 'Constraint System' to distinguish between Format and Content constraints.

Input Information

You will receive a JSON object containing:

- instruction: The complex user prompt.
- sampled_constraints: The list of constraint_ids used in this instruction.

Core Task: Decoupling and Chunking

Analyze the instruction and perform these steps:

1. **Constraint Categorization:** Referencing the Constraint System, identify which IDs in sampled_constraints are **Format Constraints** and which are **Content Constraints**.
2. **Format Extraction:** List all identified **Format Constraints**. These will be handled in the final reformatting stage.
3. **Content Chunking:** Group the **Content Constraints** into sub-tasks. Each sub-task should contain 2 to 3 constraints to maintain high accuracy.
4. **Sub-instruction Drafting:** For each content chunk, write a clear sub_instruction. This sub-instruction must ask for the raw factual information from the video (e.g., "Describe the colors of the cars and the specific actions of the driver") while ignoring all formatting rules (like JSON, tables, or word counts).

Output Specification

Return a single valid JSON object. Do not output any other text.

```
{
  "format_constraints": ["string"],
  "content_subtasks": [
    {
      "target_constraints": ["string"],
      "sub_instruction": "string"
    }
  ]
}
```

Next, carefully read the actual input below and perform the processing that conforms to the conventions and is reasonable:

The Prompt for Sub-task Execution

Role and Goal

You are a highly accurate Video Content Expert. Your task is to answer a specific sub-instruction based entirely on a highly detailed text proxy of a video (the caption). Your goal is to retrieve the exact facts, events, and timestamps requested.

Input Information

You will receive a JSON object containing:

- **caption:** A detailed textual proxy description of the video content, including visual events, audio tracks, and timestamps.
- **sub_instruction:** A specific, content-focused question or directive.

Core Task: Factual Retrieval

1. Analyze the `sub_instruction` and locate the relevant information within the `caption`.
2. Generate a highly accurate, factual response. If the instruction asks for timestamps, extract the exact timestamps from the caption. If it asks for cross-modal reasoning (e.g., matching a sound to an action), perform the reasoning strictly based on the text.

Output Specification

Return a single valid JSON object. Do not output any additional explanatory text.

```
{
  "intermediate_response": "string" // Your factually accurate, unformatted answer to the
    sub_instruction
}
```

The Prompt for Aggregation and Reformatting

Role and Goal

You are an expert formatting and synthesis engine. Your task is to aggregate multiple pieces of raw, factual information (intermediate responses) and reformat them to perfectly answer a complex original instruction. You must strictly enforce all format constraints.

Input Information

You will receive a JSON object containing:

- **original_instruction:** The complete, highly constrained user prompt that needs to be answered.
- **format_constraints:** The list of format constraint_ids (e.g., `json_object`, `length`, `unordered_list`) that MUST be strictly applied to the final output.
- **intermediate_responses:** An array of strings containing all the necessary factual information, already verified and extracted from the video.

Core Task: Synthesis and Strict Formatting

1. **Synthesize Content:** Combine all the facts, events, and timestamps provided in the `intermediate_responses`. Do not invent or hallucinate any new video content. The intermediate responses contain the absolute truth.
2. **Apply Formatting:** Restructure the combined information to strictly satisfy the `original_instruction`. Pay obsessive attention to the `format_constraints`.
 - If a specific JSON schema is required, ensure the keys and nesting are flawless.
 - If a strict length is imposed, count your words/characters carefully.
 - If specific symbols, prefixes, or markdown tables are requested, apply them exactly.
3. **Final Polish:** Ensure the final response seamlessly integrates the factual content into the required rigid structure without losing any semantic details requested by the original prompt.

Output Specification

Return a single valid JSON object containing your final constructed string. Do not output any additional explanatory text.

```
{
  "final_response": "string" // The final, fully formatted, and factually correct response
}
```

G.5 Prompts for Evaluation on Existing Benchmarks

To comprehensively evaluate our model’s generalizable omni-modal perception capabilities on external benchmarks, we utilized specific prompts to guide the model in generating detailed descriptions or performing QA tasks. The specific prompts used for Omni-Cloze (Ma et al., 2026), DailyOmni (Zhou et al., 2025), and WorldSense (Hong et al., 2026) are listed below.

Prompt for Detailed Video Description (Omni-Cloze)

You are given a short video with both audio and visual content. Write a detailed and coherent paragraph that naturally integrates all modalities. Your description should include: (1) the primary scene and background setting; (2) key characters or objects and their actions or interactions; (3) significant audio cues such as voices, background music, sound effects, and their emotional tone; (4) any on-screen text (OCR) and its role in the video context; and (5) the overall theme or purpose of the video. Ensure the output is a fluent and objective paragraph, not a bullet-point list, and captures the video’s content in a human-like, narrative style.

Prompt for Question-to-Prompt Construction (DailyOmni & WorldSense)

You are an expert prompt engineer. I will give you a list of specific questions about a video (covering spatial relations, audio changes, temporal events, etc.). Your task is to write a single, comprehensive and sophisticated prompt directed at a Vision-Language AI model. The prompt should instruct the AI to generate a detailed video caption that naturally incorporates all necessary visual and audio details required to answer these specific questions accurately.

Constraints:

1. DO NOT answer the questions yourself.
2. DO NOT list the questions in your output.
3. Your output must be ONLY the final generated prompt. No preamble or explanations.

Here are the questions describing what needs to be captured:

Prompt for Caption-to-QA Evaluation (DailyOmni & WorldSense)

You are an Omni-modal Video Understanding Expert. Your task is to answer a multiple-choice question about a video based SOLELY on the provided video caption.

Below is a detailed caption describing a video, followed by a question and four choices.

Video Caption:

{caption}

Question:

{question}

Choices:

{choices_str}

Please select the best answer based on the caption provided. If the answer cannot be determined with certainty, choose the most likely option. Respond with ONLY the uppercase letter of the correct answer (A, B, C, or D). Do not output any other text or explanation.

Prompt for Direct QA Evaluation (DailyOmni & WorldSense)

Please analyze the video and answer the following multiple-choice question. Output ONLY the letter (A, B, C, or D) of the correct answer. Do not provide any explanations, remarks, or extra text. If you are unsure, choose the most plausible option.

Question:

{question}

H Construction of The Test Set

To ensure the high quality and academic rigor of the OmniCap-IF benchmark, we designed a reproducible pipeline to transform raw videos into finalized, expert-verified samples. The construction process consists of three main stages:

H.1 Video Collection and Filtration

We curated an initial pool of over 1500 videos from diverse sources, including academic datasets (Ego4D (Grauman et al., 2022)) and social media platforms. To ensure the benchmark’s quality, we applied the following filters:

- **Resolution:** Minimum resolution of 720p to ensure visual clarity.
- **Duration:** Focused on the 30–90 second range to provide sufficient semantic density for multi-constraint tasks.
- **Multi-modality:** Ensure that the videos are both rich in audio-visual content and consistently aligned across sound and imagery.

This resulted in a core set of 480 high-quality videos covering 10 categories such as Comedy & Sketches, Lifestyle & Vlogs, and Knowledge & Tech.

H.2 Automated Draft Generation

For each selected video, we utilized the **Instruction Generator** (powered by Gemini-3.1-pro (Google DeepMind, 2026)) to produce paired instruction-checklist candidates.

- **Prompt Generation:** Constraints were sampled from our 50-type taxonomy and combined to create instructions of varying difficulty (Level 1 to Level 3).
- **Checklist Synthesis:** For each instruction, the generator simultaneously produced a **Format Checklist** (JSON schemas, length limits) and a **Content Checklist** (fact-based QA pairs).

H.3 Human Refinement and Verification

This stage is critical for ensuring the factual grounding and structural rigor of the OmniCap-IF benchmark. To achieve this, we employed a team of professionally trained annotators to conduct a three-step verification and refinement process:

1. **Instruction-Video Alignment:** Annotators first watch the video to ensure that the requirements in the instructions are strictly factually grounded in the actual visual and auditory content. Any instructions requesting non-existent entities, actions, or sounds (i.e., hallucinations) are corrected or completely rewritten.
2. **Constraint Taxonomy Compliance:** The instruction is then audited to confirm it strictly adheres to our defined taxonomy of 50 constraint types. We verify that no "out-of-scope" requirements are introduced and that the complexity levels (normal, high, extreme) are appropriately distributed across the instruction set.
3. **Checklist Factual and Formal Validation:** Once the instruction is validated, annotators rigorously examine the generated **Checklist**. This includes:
 - **Formal Check:** Confirming that format parameters (e.g., JSON schemas, length units, keyword types) in the Format Check are correctly extracted and logically consistent with the instruction.
 - **Content Check:** Manually verifying that every question in the Content Check has a unique, factually correct answer based on the video, and that the distractors are plausible but incorrect.
4. **Consensus Protocol:** Through this rigorous process, 53.1% of the samples were modified and 22.7% were discarded or rewritten. Each final sample was confirmed only after reaching a unanimous agreement among three independent annotators, with any persistent disagreements adjudicated by a senior supervisor.

I Evaluation Settings

We provide the detailed settings of our evaluated open-source models (Table 10). Most models are tested under default settings. Closed-source models are accessed via API calls, using the default configuration. The system prompts used for the models are detailed in Section G.1.

Table 10: Evaluation metrics for locally deployed open-source models. The “FPS” column represents the frame sampling rate.

Models	FPS	Temperature	Repetition Penalty	Max Token
Qwen3-Omni-30B-A3B-Thinking	1.0	0.6	1.05	4096
Qwen3-Omni-30B-A3B-Instruct	1.0	0.0	1.05	1536
MiniCPM-o-4.5-9B	1.0	0.0	1.05	1536
Qwen2.5-Omni-7B	1.0	0.0	1.05	1536
Qwen2.5-Omni-3B	1.0	0.0	1.05	1536
video-SALMONN-2-7B	1.0	0.0	1.05	1536
MiniCPM-o-2.6-8B	1.0	0.0	1.05	1536
HumanOmniV2-7B	1.0	0.0	1.05	1536
ASID-Captioner-7B	1.0	0.0	1.05	1536
ARC-Hunyuan-Video-7B	1.0	0.0	1.05	1536
OmniCaptioner-IF-7B (Ours)	1.0	0.0	1.05	1536
OmniCaptioner-IF-3B (Ours)	1.0	0.0	1.05	1536

J Training

J.1 Training Configurations

Our models, OmniCaptioner-IF-7B and OmniCaptioner-IF-3B, were developed by fine-tuning the pre-trained Qwen2.5-Omni-7B and Qwen2.5-Omni-3B models, respectively. We employed Parameter-Efficient Fine-Tuning (PEFT) via Low-Rank Adaptation (LoRA) (Hu et al., 2022) applied to all linear layers. The LoRA rank was set to 16 with an alpha value of 32.

The fine-tuning process was conducted for a total of 1 epoch on our curated OmniCap-IF-54K dataset. We utilized the AdamW optimizer with a peak learning rate of 2×10^{-5} for the 7B variant and 3×10^{-5} for the 3B variant.

To accommodate hardware constraints and maximize throughput, training was performed on a single node equipped with 8 H200 GPUs. We configured a per-device batch size of 1 and utilized 2 gradient accumulation steps, resulting in an effective global batch size of 16. To enhance computational efficiency and reduce the memory footprint, we leveraged bfloat16 (bf16) mixed-precision training. For data preprocessing, input videos were sampled at a rate of 1 FPS. The maximum resolution was capped at 401,408 pixels for the 7B model and 200,704 pixels for the 3B model to balance perceptual granularity and memory usage.

J.2 Convergence Analysis and Dataset Sufficiency

To validate the sufficiency of our 54K instruction-tuning dataset and the 1-epoch training strategy for models at the 3B and 7B scales, we analyze both the training dynamics and the intermediate checkpoint performances.

First, as illustrated by the training loss curve in Figure 11, the model exhibits a rapid and healthy descent during the initial phase. After approximately 1,000 steps, the loss gradually flattens out, eventually stabilizing and plateauing around a value of 0.9 with minor fluctuations towards the end of the epoch. This trajectory indicates that the model has smoothly converged and effectively internalized the complex instruction-following patterns within a single epoch, without suffering from under-fitting.

Second, we conducted a data scaling ablation study by evaluating intermediate checkpoints—saved after training on 20K, 40K, and the full 54K samples—on the OmniCap-IF benchmark. As detailed in Table 11, merely fine-tuning on 20K samples yields a massive performance leap compared to the base model, Qwen2.5-Omni-7B (e.g., Overall CSR jumps from 57.44% to 71.10%, and Format ISR dramatically improves from 46.52% to 79.01%).

However, as the training data volume scales from 20K to 40K, and finally to 54K, the performance gains exhibit a clear trend of diminishing returns, approaching saturation. Notably, between the 40K and 54K checkpoints, the Overall CSR only marginally increases from 72.22% to 72.59%, and the Format CSR even shows a slight oscillation (89.38% vs. 89.09%), which is a typical hallmark of model convergence.

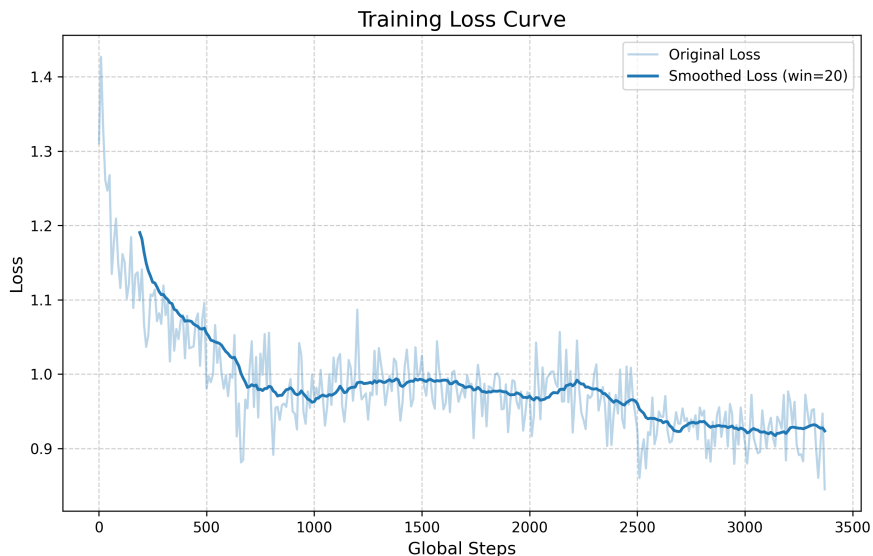


Figure 11: The training loss curve of OmniCaptioner-IF-7B over 1 epoch.

These combined observations compellingly demonstrate that 54K high-quality, constraint-rich samples constitute a sufficient "sweet spot." It effectively unlocks and solidifies the omni-modal instruction-following capabilities of models at this scale, proving that data quality and constraint diversity are far more critical than sheer dataset volume or multiple training epochs.

Table 11: Performance evolution of OmniCaptioner-IF-7B across different training data volumes on the OmniCap-IF benchmark.

Training Data	Overall		Format		Content CSR			
	CSR	ISR	CSR	ISR	Total	Visual	Audio	Omni
Base	57.44	7.55	66.78	46.52	52.54	57.13	57.14	44.18
20K	71.10	17.56	88.42	79.01	62.11	63.23	64.46	57.01
40K	72.22	18.20	89.38	79.70	63.23	63.80	66.69	58.13
54K (Full)	72.59	18.28	89.09	79.46	63.94	63.82	68.81	59.02

K Results on Omni-VideoQA Benchmarks

To comprehensively evaluate our model’s generalizable omni-modal perception and reasoning capabilities, we conducted extensive evaluations on the DailyOmni and WorldSense benchmarks. We assess the performance under two distinct settings: Caption-to-QA and Direct QA.

K.1 Caption-to-QA Performance

In this setting, the evaluation is decoupled into two stages. First, we use a specific question-to-prompt constructor to instruct the model to generate a highly detailed video caption that naturally incorporates all necessary visual and audio details. The specific prompt used for this construction is detailed in the Section G.5. Subsequently, an LLM-as-a-judge is utilized to answer the multiple-choice questions based *solely* on the generated captions.

As shown in Table 12, our OmniCaptioner-IF-7B model demonstrates exceptional performance. By strictly adhering to the constructed instruction, our model retains more crucial cross-modal facts in the generated text, achieving 60.2% on DailyOmni and 43.2% on WorldSense, surpassing most open-source counterparts and matching the performance of proprietary models like Gemini-2.5-Pro.

Table 12: QA performance by Gemini-2.5-Pro based on captions.

Model	DailyOmni \uparrow	WorldSense \uparrow
Gemini-2.5-Pro	60.2	33.8
Gemini-2.5-Flash	55.3	31.0
HumanOmniV2-7B	8.2	6.6
ARC-Hunyuan-Video-7B	8.6	8.7
MiniCPM-o-2.6-8B	9.8	7.2
Qwen2.5-Omni-7B	13.4	8.6
UGC-VideoCaptioner-3B	17.0	11.2
video-SALMONN-2-7B	29.9	18.2
Qwen3-Omni-Instruct-30B-A3B	17.5	12.7
AVoCaDO-7B	50.1	25.7
ASID-Captioner-7B	61.2	34.0
OmniCaptioner-IF-7B (Ours)	60.2	43.2

K.2 Direct QA Performance

In the Direct QA setting, models take the video and the question as direct inputs to predict the final answer without generating an intermediate caption. This tests the model’s native end-to-end multi-modal understanding capacity.

As illustrated in Table 13, our model exhibits strong end-to-end reasoning capabilities. OmniCaptioner-IF-7B achieves 68.4% on DailyOmni and 49.4% on WorldSense, significantly outperforming its base model Qwen2.5-Omni-7B.

Table 13: Direct QA performance on Omni-VideoQA benchmarks.

Model	DailyOmni \uparrow	WorldSense \uparrow
Gemini-2.5-Flash	73.1	50.9
GPT-4o	56.5	42.6
VideoLLaMA2-7B	35.2	25.4
Qwen2.5-Omni-7B	62.1	45.4
video-SALMONN-2-7B	66.3	48.6
Qwen3-Omni-30B-A3B-Instruct	71.9	54.0
OmniCaptioner-IF-7B (Ours)	68.4	49.4