ProMQA: Question Answering Dataset for Multimodal Procedural Activity Understanding

Kimihiro Hasegawa[1] **Wiradee Imrattanatrai**[2] **Zhi-Qi Cheng**[1] **Masaki Asada**[2]
Susan Holm[1] **Yuran Wang**[1] **Ken Fukuda**[2] **Teruko Mitamura**[1]
[1]Language Technologies Institute, Carnegie Mellon University
[2]National Institute of Advanced Industrial Science and Technology (AIST)
kimihiro@cs.cmu.edu

Abstract

Multimodal systems have great potential to assist humans in procedural activities, where people follow instructions to achieve their goals. Despite diverse application scenarios, systems are typically evaluated on traditional classification tasks, e.g., action recognition or temporal action segmentation. In this paper, we present a novel evaluation dataset, ProMQA, to measure system advancements in application-oriented scenarios. ProMQA consists of 401 multimodal procedural QA pairs on user recording of procedural activities, i.e., cooking, coupled with their corresponding instructions/recipes. For QA annotation, we take a cost-effective human-LLM collaborative approach, where the existing annotation is augmented with LLM-generated QA pairs that are later verified by humans. We then provide the benchmark results to set the baseline performance on ProMQA. Our experiment reveals a significant gap between human performance and that of current systems, including competitive proprietary multimodal models. We hope our dataset sheds light on new aspects of models' multimodal understanding capabilities.[1]

1 Introduction

Procedures are human knowledge of experience that enables one to obtain an expected outcome without much trial and error. Yet, following procedures (i.e., a set of instructions), itself requires skills such as, in cooking (Peddi et al., 2023), assembly (Sener et al., 2022), or surgery (Beyer-Berjot et al., 2016), among others. In supporting such user activities, current evolving multimodal foundation models like GPT-4o (OpenAI, 2024) and Claude 3.5 Sonnet (Anthropic, 2024) have great potential by monitoring the situation through the perception of a user's wearable device. Despite such diverse application scenarios, existing studies typically provide traditional, but less practical evaluation testbeds. To support an application-oriented evaluation, we present a novel multimodal question-answering (QA) dataset for understanding procedural activity, produced by our cost-effective human-LLM collaborative approach.

When supporting procedural activities, an assistant should comprehend information from multiple sources: 1) Actual process from their perception; 2) Each step and the overall flow from instructions. For instance, in cooking, answering "*What is the next step now?*" requires an assistant to recognize which steps have been completed until "*now*" from its video recording and identify what else/next should be done from its recipe. Assuming recipes are typically written in text, assistants receive multimodal information of *how one did it* as video and *how one should do it* as text. Prior work has explored the task in a text-only, unimodal setting, where a user verbalizes all of their actions (Le et al., 2023). However, it is not ideal in practice as a beginning cook might give misleading explanations that cannot be corrected by a system without raw information (video) about the actual process.

Figure 1 illustrates how one receives cooking support from a system in a reactive manner. Tailoring toward such a practical scenario, we formulate our task as QA so that multimodal capabilities can be evaluated directly on the downstream task (§2.1). In contrast, prior work traditionally tackles visual action understanding as action recognition and temporal action segmentation (Kuehne et al., 2014; Tang et al., 2019; Ding et al., 2022). We argue that these tasks are suboptimal to evaluate procedural activity assistants as they are subtasks of such procedural activity support.

In this work, we present a novel dataset, **ProMQA** (**Pro**cedural **M**ultimodal **Q**uestion **A**nswering), to evaluate models' capabilities of understanding procedural activities in multimodal settings (§2). Our work is motivated by the fact that a

[1]Code and data are available at https://github.com/kimihiroh/promqa.

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 11598–11617
April 29 - May 4, 2025

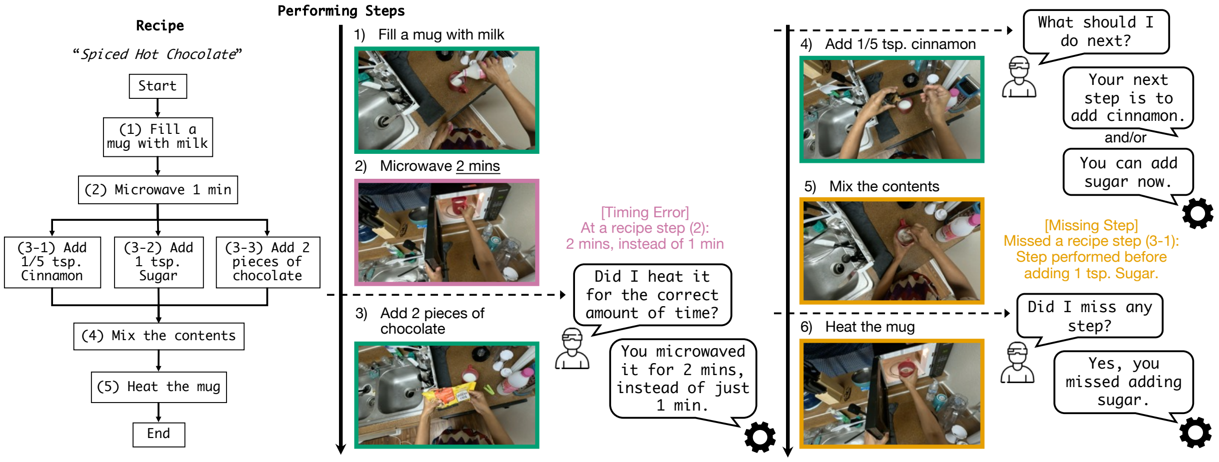

Figure 1: Illustration of a system supporting a user in a procedural activity. The left graph is the recipe and the columns of images are screenshots of the user's actions in chronological order. During the activity, the user makes two mistakes. One is *a timing error*, where the user sets a longer time than required for microwaving (red). The other is *a missing step*, where the user skips adding sugar (yellow borders for steps after the missing step). Steps with green borders do not have any errors. QAs are occurring at each divider's position.

well-adapted testbed is indispensable and can stimulate system development. In the dataset construction, we repurposed videos and recipes from the existing CaptainCook4D (Peddi et al., 2023) dataset. Then, for QA annotation, we employ a human-LLM collaborative approach, where LLMs first generate QA pairs and humans verify them to ensure the quality, inspired by the recent advances in synthetic data generation (Mangalam et al., 2023) (§3). While LLMs cost-effectively generate candidate QA pairs, the manual verification process ensures the quality of the resulting dataset. Specifically, among 500 generated QA pairs, around 80% were retained with additional human-written answers through the verification. Finally, to establish the baseline performance, we benchmark the following approaches: unimodal models, Socratic models (Zeng et al., 2022), and both open and proprietary multimodal models. Our benchmark experiments reveal that, while humans can reasonably perform the task, the dataset is challenging even for proprietary multimodal models that show strong performance on other vision-language tasks (§4).

Our contributions are three-fold. First, we define a novel multimodal QA task and present the dataset, ProMQA, for procedural activity understanding under a permissive license.[2] Second, we propose a human-LLM collaborative approach for cost-efficient QA annotation. Third, we provide benchmark results to encourage further research on this task.

[2]Apache 2.0

2 ProMQA

Our goal is to facilitate the development of procedural-activity support systems. ProMQA consists of 401 multimodal procedural QA pairs that require both recipes and video recordings to answer. It is constructed with our human-LLM collaborative approach on top of existing cooking recording and annotation (§3). In Table 1, we compare our dataset with similar multimodal datasets. Our dataset uniquely supports the assessment of multimodal procedural activity understanding as the QA task, which can serve as a testbed to advance the model's multimodal procedural activity understanding.

2.1 Task Formulation

We chose QA as our formulation to better reflect how users seek information and advice in practical situations. A model takes as input a cooking instruction $recipe$, a recording of a user's activity $video$, and a question q, and then, outputs an answer a as natural language. A $recipe$ is represented as a directed acyclic graph of recipe steps, whereas a $video$ contains a pile of frames. In this work, we treat each QA pair independently, instead of formulating it as dialogue, to focus on reasoning capability, and leave it for future work on how to extend to further practical dialogue settings. We also note that "instruction" and "recipe", and "recording" and "video", are used interchangeably.

Dataset Name	Multimodal	Video	Procedural	Explicit Instruction	QA	Open Vocab	LLM Scoring
Assembly101 (Sener et al., 2022)	✓	✓	✓	✗	✗	✗	✗
IndustReal (Schoonbeek et al., 2024)	✓	✓	✓	✗	✗	✗	✗
YouCook2 (Zhou et al., 2018)	✓	✓	✓	✓	✗	✗	✗
CaptainCook4d (Peddi et al., 2023)	✓	✓	✓	✓	✗	✗	✗
How2QA (Li et al., 2020)	✓	✓	✓	✗	✓	✗	✗
MMBench (Liu et al., 2024b)	✓	✗	✗	✗	✓	✓	✓
EgoSchema (Mangalam et al., 2023)	✓	✓	✗	✗	✓	✗	✗
GazeVQA (Ilaslan et al., 2023)	✓	✓	✓	✗	✓	✗	✗
OpenEQA (Majumdar et al., 2024)	✓	✓	✗	✗	✓	✓	✓
ProMQA (Ours)	✓	✓	✓	✓	✓	✓	✓

Table 1: Our dataset vs. similar multimodal benchmarks

Criteria & Example	Explanation
Multimodal	
✓ What is the next step now?	This is multimodal because it requires matching the completed steps from the recording to the instructions in order to identify the possible next steps.
✗ What am I supposed to do after X?	This is not multimodal because it can be answered by simply checking the instructions.
✗ What did I do after X?	This is not multimodal because it can be answered by simply checking the recording.
Procedural	
✓ Did I measure X correctly?	This is procedural because it asks specifically about a step.
✗ What is the color of the tablespoon?	This is not procedural because it asks for the static characteristic of a tool.
No External Knowledge	
✓ Did I use the correct tool to measure X?	Suppose the instructions provide sufficient details about the measurement tool, it can be answered using the instructions and the recording, without requiring external knowledge.
✗ Can I replace zucchini with cucumber?	Suppose the recipe does not mention possible replacements, it is unanswerable from the given information. External knowledge would be required to find an answer.

Table 2: Criteria of our target multimodal procedural questions with cooking-context examples. Our target questions require both instructions and recordings to answer (multimodal), which are about either the process or each step (procedural) and are answerable from given information (no external knowledge).

Question type	Target	Example question
Process-level		
Missing	Missing recipe steps	Did I miss any steps so far?
Next	Next recipe steps	What is the next step now?
Order	Errors w.r.t. recipe step ordering	Should I have done any steps in a different order?
Step-specific		
Measurement	Errors in measurement (e.g., 2 cups instead of 1 cup)	Did I measure water correctly?
Preparation	Other errors in preparation (e.g., cilantro instead of oregano)	Did I add the correct spice?
Technique	Errors in cooking technique (e.g., chop instead of slice)	Did I prepare onion correctly?
Temperature	Errors in temperature (e.g., high instead of low)	Was the heat level correct?
Timing	Errors in duration (e.g., 2 min instead of 5 min)	Did I microwave it for long enough?

Table 3: Question categories and types with their corresponding target phenomenon and example questions.

2.2 Multimodal Procedural QA

In ProMQA, we specifically target multimodal questions about procedural activities. Multimodal questions require both instructions and recordings to derive answers, while procedural questions pertain to either individual steps or multiple-step sequences. In addition, we only retain answerable questions without requiring external or inherent knowledge to emphasize multimodal reasoning capabilities over the provided information. Table 2 provides examples that distinguish our target from relevant but out-of-scope questions.

Among valid multimodal procedural questions, we categorize them into two groups, where each is further divided into specific question types, following CaptainCook4D. **Process-level questions** focus on multiple steps: *missing*, *next*, and *order*. **Step-specific questions** are questions about individual steps: *measurement*, *preparation*, *technique*, *temperature*, and *timing*. Examples of each type and their descriptions can be found in Table 3.

Answers are categorized into three groups. Suppose a user asked a question, e.g., "*What should I do next?*". **Direct answers** directly address

#example (#question)	#distinct recipe	avg. #steps/ recipe	#distinct recording	avg. length of recording	avg. #steps/ recording	avg. #answers/ question	avg. #words/ question	avg. #words/ answer
401	24	14.3	231	6m47s	6.4	1.9	8.9	11.8

Table 4: Statistics of ProMQA

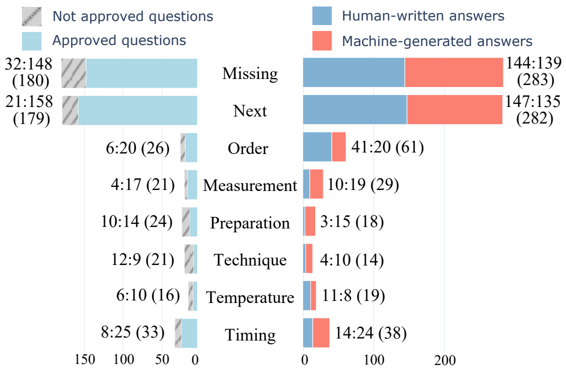

Figure 2: Question approval counts (left) and the answer counts by source (right) for each question type.

	Ours	Human (est.)
Cost / Hour	5 USD / 0.5 Hour	800 USD / 40 Hour

Table 5: Cost comparison between our human-LLM collaborative approach and a full-human approach for generating 500 QA pairs. For the latter, we asked one annotator to create 50 QAs from scratch, which took 4 hours at an assumed hourly rate of 20 USD.

the questions, e.g., "*The next step is to heat the mug*". **Suggestions** offer additional information and suggest extra actions to rectify previous errors, e.g., "*You can heat the mug after adding sugar and mix it again*," where the user forgot to add sugar. **Interventions** inform a user of irreparable situations and recommend starting over from an earlier point, e.g., "*You should start over with filing the mug with milk instead of water*," where the user mistakenly filled the mug with water.

2.3 Statistics

We show the general statistics of our dataset in Table 4. Among the 401 examples, 225 examples have no errors in previous steps (clean) and 176 examples have at least one error in previous steps. Figure 2 illustrates the high approval rate for questions, while approximately 50% of answers were added by humans through the verification process. In addition to showing the total count of each answer characteristic, we also count the number of examples with each combination of answer sources and types, as shown in Figure 3 and 4. For these

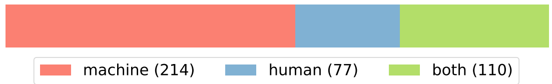

Figure 3: Answer source: The number of examples with only machine-generated answers, only human-written answers, or both types of answers (count).

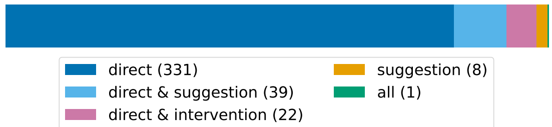

Figure 4: Answer type: The number of examples with only direct answers, direct answers and suggestions, direct answers and interventions, only suggestions, or all types of answers (count). Note that other combinations, i.e., only interventions or suggestions and interventions, are not found in our dataset.

analyses, while we used the answer source information retained through the annotation process, we obtained the answer type information by asking one annotator to categorize each answer into one of three types. We further compare the cost of our human-LLM collaborative approach and the estimate of the full-human annotation in Table 5. QA annotation for evaluation/test data typically consists of two steps: initial QA creation, followed by verification to assure the quality. We only compare the cost of the QA creation/generation part as our annotation framework replaces humans with LLMs in the initial QA creation (§3). According to the table, our approach substantially reduces the cost of the QA creation part.

3 Annotation: *Generate-then-Verify*

In this work, we take a human-LLM collaborative approach to annotate QA pairs: LLMs *generate* QA pairs with lower cost, *then* humans *verify* them to ensure quality. We hypothesize that LLMs can substantially generate valid questions when given sufficient information, inspired by synthetic data generation (Mangalam et al., 2023; Wu et al., 2024). Specifically, we leverage existing annotations of action and error labels to form textual prompts. We note that, as our annotation framework is LLM agnostic, it can plug and play LLMs, and importantly, it can benefit from ongoing LLMs' improvement.

3.1 Source & Preprocess

We chose CaptainCook4D (Peddi et al., 2023) as our data source because it includes explicit instructions and user recordings with human-annotated actions and error labels.

First, we extract video segments of various lengths using annotated action temporal segmentations. Given an original recording $video_{original}$ with n actions, we create $n+1$ video clips $video_{0:k}$ such that each clip contains the first k recording steps $S^{video}_{0:k} = \{s^{video}_0, ..., s^{video}_k\}$ $(k = 0, 1, ..., n$ and $s^{video}_0 = \varnothing)$. Each video clip with its corresponding recipe constitutes one data example d_{init}: $d_{init} = \langle recipe, video_{0:k}\rangle$. From each d_{init}, we augment 2~8 examples by adding each question type based on existing error labels: $d_{type} = \langle recipe, video_{0:k}, type\rangle$. Specifically, we create d_{type} for each, *next* and *missing*. For other six types, we create d_{type} only when the last recording step s^{video}_k in $video_{0:k}$ has a corresponding error annotation. This is based on our preliminary experiment, revealing that LLMs struggled to generate those six types of questions when no corresponding errors were annotated.

After obtaining approximately 11,000 examples from this process, we sampled 500 examples by taking the following points into account to increase diversity: (1) Sample one example for each question type from each recording; (2) Evenly sample examples with errors (*noisy*) and without errors (*clean*) in previous steps for all types; (3) Evenly sample examples that do and do not have target recipe steps for *next* and *missing* types.[3] Note that the activities in CaptainCook4D do not always result in the expected outcomes, i.e., failed procedures are included. Hence, our *noisy* examples allow us to generate QAs on top of unaddressed and/or irreparable errors from previous steps.

3.2 QA Generation

Given d_{type}, we prompt an LLM to generate a QA pair. Figure 5 shows a shortened example of our prompt, and an actual example is available in Appx. B.2. Each prompt consists of three pieces of information: (1) the textual description of $S^{video}_{0:k}$, (2) an excerpt from $recipe$ to embed what is next, missing, or incorrect, and (3) $type$, the question type to guide generation. We feed the prompts to an LLM to generate l QA pairs, from which we randomly pick one ($l = 3$). This is based on our preliminary experiment, where single pair generation often leads to monotonic question expression, e.g., "What is the next step?" across multiple $next$ examples. With GPT-4o as our QA generator, we obtain 500 examples with a pair of a machine-generated question q^m and its machine-generated answers $A^m = \{a^m_1, a^m_2, ...\}$: $d_{gen} = \langle recipe, video_{0:k}, q^m, A^m, \rangle$.

In fact, it is not trivial how to represent information in prompts and which LLMs to use to obtain better QA pairs. We conduct ablation studies to determine the prompt template and LLM.

Prompt Exploration In this ablation study, we compare the methods to embed $recipe$ and $video$ information, using small samples of d_{type} and a fixed QA generator. In a typical full-human annotation scenario, $recipe$ and $video_{0:k}$ are represented as a whole recipe and a video segment, respectively. Inspired by this, we consider the following settings: For a recipe, we compare three methods: a whole recipe as a DOT language graph (Koutsofios et al.,

[3]Example question without target recipe steps: "Did I miss any step?" "No."

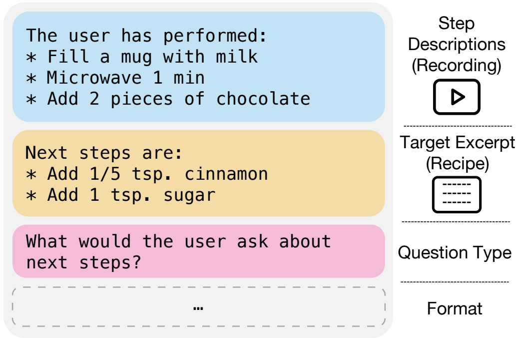

Figure 5: Example prompt with recording steps to embed recording information, an on-target excerpt from a recipe, and a question type for QA generation.

Recording	Recipe		
	DOT	image	excerpt
frames	0.43	0.53	0.65
step	0.54	0.60	0.71

Table 6: Approval rate comparison for QA generation prompts with a fixed LLM, GPT-4o.

Template	QA Generator		
	GPT-4o	Gemini 1.5 pro	Claude 3.5 Sonnet
excerpt & step	0.71	0.69	0.68

Table 7: Approval rate comparison for QA generators.

1991) ("DOT"), a whole recipe graph as an image ("image"), and only the on-target excerpt from a recipe ("excerpt"). We use DOT to accurately represent the partial graph information in a recipe. For a video segment, we feed a video segment as sampled frames ("frame") and a list of step descriptions ("step"). Actual example prompts are available in Appx B.2. We generate 80 questions for each combination using GPT-4o and ask one annotator to check if they are multimodal procedural questions. Table 6 shows the approval rate for each combination, i.e., how many generated questions passed the check. We found that feeding the combination of the excerpt from a recipe and step descriptions resulted in the most approved QA pairs.

QA Generator Selection In the second ablation study, we compare QA generators by fixing the prompt template (excerpt & step). The following LLMs are our candidates: GPT-4o, Claude 3.5 Sonnet, and Gemini 1.5 Pro (Google, 2024).[4] Similar to the prompt exploration, we use the approval rate as our metric based on the annotator's judgments. As shown in Table 7, the performance is not very different, yet, we found that GPT-4o generates slightly more valid questions.

3.3 Verification

LLM-generated questions and answers are not guaranteed to be valid. Thus, we resort to human annotators to ensure the quality of our evaluation data.

Criteria For questions, annotators check if each is a valid multimodal procedural question, as described in §2.2, and assess for naturalness, clarity, and grammatical correctness. For answers, annotators verify the correctness of each answer.

Process Our verification process involves two stages: In the first stage, two annotators independently verify each question and its answers in d_{gen}. When a question is marked as valid, its answers are shown to annotators to verify. Otherwise, annotators move on to the next example. During answer verification, annotators can add human-written answers $A^h = \{a^h_1, a^h_2, ...\}$, including suggestions and interventions, when any generated answers are incorrect or additional correct answers are missing. When two annotations for one d_{gen} conflicts or at least single a^h exists, an additional annotator (i.e., adjudicator) further verifies examples to make the final judgment or to have an additional check. More details are available in Appx B.3.

We first created an annotation guideline and hired 6 people with graduate degrees in NLP-related fields for our annotation. Among the participants, five people served as first-stage annotators, while the other, who was also involved in the guideline development, took the adjudicator's role. This adjudicator performed the answer categorization, verification, and human judgment in §2.3, §3.2, and §4.2 as well. To help familiarize the annotators with the task, we conducted a training phase in which each annotator verified 20 examples and received personalized feedback. Following the training session, we initiated the main phase. On average, judgment agreements were 0.87 for both questions and answers. After the verification, we obtained 401 verified examples $d_{ver} = \langle recipe, video_{0:k}, q^m_{ver}, A^m_{ver}, A^h_{ver} \rangle$.

4 Benchmarking

On our ProMQA, we provide the baseline results of existing models to facilitate the development of a user-support system for procedural activities. Considering that our task contains natural language answers, we employ an LLM-based metric to evaluate the performance of the baselines.

4.1 Target Models

We consider the following approaches:

Unimodal Model One baseline consists of a text-only unimodal model, which shows how many examples in ProMQA can be solved/guessed solely from textual information (i.e., instructions and questions). Vision-only unimodal models are not considered, as inputs without questions would not guide the model to generate on-target answers. We employ Llama 3.1 Instruct (Dubey et al., 2024).

Socratic Model Another baseline is a two-model pipeline: one generates captions from visual inputs, and the other generates answers based on those captions and text information. This approach demonstrates how many questions can be answered with restricted cross-modal/frame reasoning. We use LLaVA 1.5 (Liu et al., 2024a) for image captioning and Llama 3.1 Instruct for text-based reasoning.

Multimodal Model As one of our main targets, we assess open multimodal models, especially the ones tailored towards video understanding. Based

[4]These model versions were used throughout the paper: `gpt-4o-2024-08-06`, `claude-3-5-sonnet-20240620`, `gemini-1.5-pro-001`.

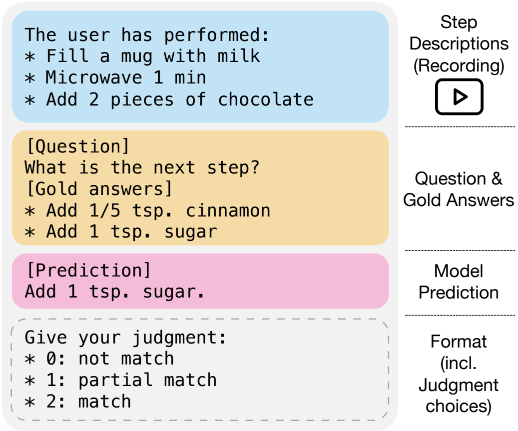

Figure 6: Example prompt for LLM Scoring with recordings as context information, a question with its gold answer(s), and a model prediction.

#choice	Context information		
	default	DOT	step
binary	0.67/0.86	0.57/0.80	0.71/0.89
ternary	0.67/0.69	0.58/0.64	0.76/0.75

Table 8: LLM-based scoring prompt comparison (Pearson/Acc.)

Template	Evaluator		
	GPT-4o	Claude 3.5 Sonnet	Gemini 1.5 Pro
ternary & step	0.83/0.82	0.79/0.77	0.66/0.68

Table 9: Evaluator comparison (Pearson/Acc.)

on the strong performance on the existing multimodal benchmarks, e.g., MMMU (Yue et al., 2024) and Video-MME (Fu et al., 2024), we evaluate VideoLLaMA2 (Cheng et al., 2024) and Qwen2-VL (Wang et al., 2024). Finally, we test proprietary multimodal models (i.e., GPT-4o, Claude 3.5 Sonnet, and Gemini 1.5 Pro) considering their strong performance in various benchmarks.

4.2 LLM-as-a-Judge

Evaluating natural language itself is a challenging task due to multiple correct answers and their possible variations for the same question. In place of string-based metrics, e.g., BLEU (Papineni et al., 2002), which often struggle with such an answer diversity, LLM-based metrics, i.e., LLM-as-a-judge (Zheng et al., 2023) are getting increasing attention. Considering possible correct answers, we also employ LLM-as-a-judge in the experiment. Figure 6 shows our shortened prompt for our LLM-based scoring. As a calibration process, we conduct ablation studies to choose which information to feed in prompts and an LLM as our evaluator.

Prompt Exploration and Evaluator Selection We aim to identify a prompt template and an LLM that yields a high correlation with human judges. We consider two key aspects in templates: 1) the number of choices in the Likert scale and 2) the context information. For choices, we consider "binary" (*match* and *unmatch*) and "ternary" (*match*, *partial-match*, and *unmatch*). For context, we examine three settings: With a question, gold answers, and a predicted answer as the fundamental elements ("default"), we then incorporate either instruction ("DOT") or step descriptions from recordings ("step"). Candidate evaluators include GPT-4o, Claude 3.5 Sonnet, and Gemini 1.5 Pro. In the experiment, we feed inputs based on the verified examples in §3.2 to LLM-evaluators to obtain predictions. Then, we obtain judgments from these LLMs with all combinations. As a comparison, we obtain human judgments, where one person judges the predictions with both binary or ternary options. We consider Pearson correlation coefficient (Pearson, 1895) and match accuracy as our metrics. Table 8 shows the average scores across three evaluator models. We found that the combination of "ternary" and "step" produces the highest correlation. With the best combination, we compare the evaluators. Table 9 shows that GPT-4o has the best correlation with the human judgments. In the benchmark experiment, we scaled judgment scores from 0-2 to 0-100 by multiplying 50.

4.3 Results

In this experiment, we also obtained human performance as a comparison. We asked five first-stage annotators (§3.3) to solve 20 samples, out of 401 total examples, which they had not previously checked during the verification process. The sampling was done due to our budget. Performers were provided only recipes and video segments with questions. In Table 10, we provide the average performance, as well as the breakdown based on previous step types and question types. It shows that all the models we benchmark lag behind human performance, even the competitive proprietary models. Among the models, Claude 3.5 Sonnet performs relatively better than others, although the differences are somewhat marginal. In general, *clean* examples are easier for models than *noisy* examples, although the gap varies depending on each model. Proprietary models are, on average,

Model	Avg.	Error		Question Type							
		clean	noisy	missing	next	order	measurement	preparation	technique	temperature	timing
Llama 3.1 70B Instruct	31.5	33.2	30.2	35.5	38.3	12.5	2.9	14.3	5.6	30.0	20.0
LLaVA 1.5 13B (50f, 288p) & Llama 3.1 70B Instruct	37.5	38.9	36.4	41.9	45.3	12.5	11.8	14.3	11.1	50.0	18.0
VideoLLaMA2 72B (8f, 336p)	39.8	49.4	32.2	46.3	49.7	20.0	0.0	21.4	11.1	25.0	8.0
Qwen2-VL 72B (100f, 336p)	31.2	32.1	30.4	34.1	37.0	45.0	0.0	10.7	0.0	20.0	14.0
GPT-4o (50f, 765p)	40.4	39.5	41.1	39.9	45.9	45.0	29.4	17.9	27.8	55.0	24.0
Gemini 1.5 Pro (50f, 765p)	25.2	27.0	23.8	27.4	29.7	15.0	17.6	7.1	16.7	20.0	12.0
Claude 3.5 Sonnet (10f, 765p)	44.1	48.9	40.4	44.6	58.2	27.5	8.8	14.3	5.6	25.0	28.0
Human*	(74.5)	(83.5)	(65.8)	—	—	—	—	—	—	—	—

Table 10: Benchmark result: The average of all the examples, the averages of examples with (noisy) and without errors (clean) in previous steps, and the averages for the same question-type examples. f and p denote the number of frames and image resolution used for each model. Note that each category contains a different number of examples. *: Human performance is based on the sampled 20 examples.

Model	Avg.	Answer Source			Answer Type				
		machine	human	both	direct	direct & suggestion	direct & intervention	suggestion	all
Llama 3.1 70B Instruct	31.5	33.4	28.0	30.5	31.7	30.8	34.1	25.0	0.0
LLaVA 1.5 13B (50f, 288p) & Llama 3.1 70B Instruct	37.5	40.7	32.9	34.8	38.8	34.6	34.1	12.5	0.0
VideoLLaMA2 72B (8f, 336p)	39.8	45.1	32.9	34.3	41.7	34.6	31.8	12.5	0.0
Qwen2-VL 72B (100f, 336p)	31.2	35.0	22.0	30.5	30.8	33.3	27.3	37.5	100.0
GPT-4o (50f, 765p)	40.4	41.4	29.3	47.1	40.2	39.7	52.3	18.7	50.0
Gemini 1.5 Pro (50f, 765p)	25.2	27.1	18.3	26.7	24.8	25.6	34.1	12.5	50.0
Claude 3.5 Sonnet (10f, 765p)	44.1	48.1	35.4	42.9	45.0	42.3	43.2	25.0	0.0

Table 11: Answer-focused benchmark result breakdown: The average of all the examples, the averages of examples with only machine-generated answer(s), human-written answer(s), and both; The averages of examples with only direct answer(s), direct and suggestion(s), direct and intervention(s), only suggestion(s), and all answer-types. f and p denote the number of frames and image resolution used for each model. Note that each category contains a different number of examples.

better on step-specific questions. Additionally, we show the breakdown based on answer sources and types in Table 11. From the table, we can see that models generally perform better on examples with only machine-generated answers, although each model exhibits different preferences. Furthermore, we investigate the effect of answer counts of examples on performance. There is a weak common trend that models perform well on examples with a single answer and with 4 answers. Considering our results do not always align with those in the public benchmarks like Video-MME, we believe our ProMQA can be complementary in evaluating models' multimodal capabilities.

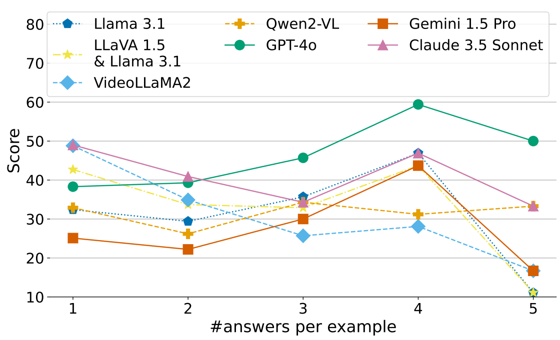

Figure 7: Model performance with different numbers of answers. Note that each category contains a different number of examples and examples with more than 5 answers are excluded due to their small counts.

5 Self-Preference Bias Analysis

Prior studies report that LLMs may introduce self-preference bias: "an LLM favors its own outputs over texts from other LLMs and humans." (Panickssery et al., 2024) This can be a critical issue when LLMs play multiple roles, as in our experiment, i.e., use LLM-generated QAs to evaluate LLMs with LLM-as-a-judge. To justify the fairness of ProMQA as a benchmark dataset, we investigate: (1) **generator-predictor** self-preference bias, where the generator's outputs harbor styles or characteristics that make it easier for the model to answer, and (2) **predictor-evaluator** self-preference

Predictor	Generator		Evaluator	
	GPT-4o	Gemini	GPT-4o	Gemini
GPT-4o	22.8	36.4	27.5	31.4
Gemini	14.0	12.7	21.6	24.5

Table 12: Result of generator-predictor and predictor-evaluator self-preference bias checks. Each number represents the score by a human evaluator. Gemini denotes Gemini 1.5 Pro.

bias, where the evaluator favors their own outputs, rather than objectively assessing the accuracy or quality of the predictions. Our experiments show no noticeable sign of biases.

5.1 Bias: Generator-Predictor

We investigate if questions generated by an LLM is easier for the same LLM to derive answers. To conduct a control experiment, we change generators and predictors, while fixing other variables, i.e., the verification person and evaluator (manual). According to Table 12, we did not find an indication that a model scores higher on its generated questions. One reason could be the modality difference between generation (text-only) and prediction (text & visual inputs), but we leave it for future work.

5.2 Bias: Predictor-Evaluator

We then examine the original self-preference bias, i.e., if an LLM favors their own predictions over others. We fix generators (i.e., verified QAs from three LLMs), and change predictors and evaluators with the same set of LLMs for each. Contrary to the previous work, Table 12 shows no sign of the bias. We believe that QA evaluation is more objective than the summarization used by Panickssery et al. (2024), resulting in less room for model-based bias. We again put deeper analysis as future work.

6 Related Work

Procedural Activity Understanding The research community constructed various datasets to improve the machine's understanding of procedural activities in videos: Breakfast (Kuehne et al., 2014), YouCook2 (Zhou et al., 2018), COIN (Tang et al., 2019), Assembly101 (Sener et al., 2022), and CaptainCook4D (Peddi et al., 2023), to name a few. With those datasets, models are typically evaluated on tasks like action recognition and temporal action localization, framed as classification tasks. In this work, we propose QA as the formulation, which aligns better with real-world scenarios.

Video QA Dataset QA as a task formulation is increasingly adopted for video QA datasets, e.g., NExT-QA (Xiao et al., 2021), EgoSchema (Mangalam et al., 2023), OpenEQA (Majumdar et al., 2024), Video-MME (Fu et al., 2024), *inter alia*. While they are multimodal, i.e., a model takes video frames and a textual question as inputs, we argue that they are still rather video-oriented as only a short question consists of the textual part, compared to a pile of images from a video. While GazeVQA (Ilaslan et al., 2023) uniquely focuses on procedural tasks as QA, instructions are yet explicitly provided to systems, hence, only a short question with multiple choices and a video are the inputs. For enhanced cross-modal comprehension, we present ProMQA where textual instructions are necessary to derive a correct answer in addition to a video and question (§2.1).

Synthetic Evaluation Data Along with the advancement of LLMs, synthetic data generation is widely explored in various phases of model development, including pretraining (Gunasekar et al., 2023; Maini et al., 2024) and instruction tuning (Wang et al., 2023; Adler et al., 2024). Compared to those phases, it is underexplored in generating evaluation data with LLMs (Wu et al., 2024), possibly because of the following two reasons: 1) The quality assurance is lacking, which can be mitigated by introducing multi-step machine and manual curation steps as EgoSchema. 2) Potential biases may be introduced (Zheng et al., 2024; Panickssery et al., 2024). Addressing these challenges, we develop our ProMQA with additional human checks (§3.3), justified by the fairness-check experiments (§5).

7 Conclusion

In this paper, we propose a human-LLM collaborative approach, *Generate-then-Verify*, and develop a novel evaluation dataset, ProMQA, for multimodal procedural activity understanding. ProMQA consists of 401 QA pairs that require understanding both instructions and videos to derive answers, queried by questions. We also provide the baseline performance of existing models, showing that there is still a large gap in performance between humans and machines, even the competitive proprietary multimodal models. We believe that ProMQA can shed light on the new aspect of multimodal capabilities to facilitate model development.

8 Limitation

We note a couple of limitations remain in this work. First, the size of the dataset is relatively small. This may affect the confidence of performance comparisons when two models receive similar scores. We plan to increase the number of examples so that the research community can present their incremental progress, i.e., a few point improvements, with higher confidence (Card et al., 2020). However, despite its limited size., ProMQA is carefully curated with a representative selection of questions and answers through our data annotation design. This enables it to serve as an effective testbed for multimodal foundation models for providing insights into model performance.

Second, the domain is restricted to a single activity, cooking. Remember our annotation framework assumes the action and error labels, explicit instructions, and procedural videos. While our source dataset, CaptainCook4D, uniquely satisfies all the prerequisites, it does not apply to other existing datasets. We leave it to future work how to extend our work to integrate other activities by making use of other datasets.

Third, the dataset is oriented toward English and Western countries, especially, the U.S. CaptainCook4D contains recipes that originate from non-English speaking regions, e.g., "Ramen" or "Bruschetta," but recipes and cooking environments are designed for people in the U.S. We believe that our dataset can support the advancement of frontier multimodal models, which can also benefit diverse and/or general models. However, considering the ubiquitous potential of our target user-support systems, we hope to contribute to the development of systems for people in non-English, non-Western countries.

Finally, we release our dataset as evaluation data, not for training data, which complies with the terms of use by OpenAI.[5]

9 Ethical Consideration

In the dataset construction, we used LLMs that are pretrained on a massive web-scraped corpus, which may contain some toxic or biased information. We do not aim to include any prejudiced, offensive, or biased content in the dataset, and we did not find any in our verification process. CaptainCook4D received IRB approval and participants provided written consent in their data collection, and no private information included.[6]

Acknowledgements

We thank the anonymous reviewers for their insightful comments. This work is partially supported by Programs for Bridging the gap between R&D and the IDeal society (society 5.0) and Generating Economic and social value (BRIDGE) / Practical Global Research in the AI × Robotics Services, implemented by the Cabinet Office, Government of Japan.

References

Bo Adler, Niket Agarwal, Ashwath Aithal, Dong H Anh, Pallab Bhattacharya, Annika Brundyn, Jared Casper, Bryan Catanzaro, Sharon Clay, Jonathan Cohen, et al. 2024. Nemotron-4 340b technical report. *arXiv preprint arXiv:2406.11704*.

Anthropic. 2024. Introducing claude 3.5 sonnet.

Laura Beyer-Berjot, Stéphane Berdah, Daniel A. Hashimoto, Ara Darzi, and Rajesh Aggarwal. 2016. A virtual reality training curriculum for laparoscopic colorectal surgery. *Journal of Surgical Education*, 73(6):932–941.

Dallas Card, Peter Henderson, Urvashi Khandelwal, Robin Jia, Kyle Mahowald, and Dan Jurafsky. 2020. With little power comes great responsibility. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9263–9274, Online. Association for Computational Linguistics.

Zesen Cheng, Sicong Leng, Hang Zhang, Yifei Xin, Xin Li, Guanzheng Chen, Yongxin Zhu, Wenqi Zhang, Ziyang Luo, Deli Zhao, and Lidong Bing. 2024. Videollama 2: Advancing spatial-temporal modeling and audio understanding in video-llms. *arXiv preprint arXiv:2406.07476*.

Guodong Ding, Fadime Sener, and Angela Yao. 2022. Temporal action segmentation: An analysis of modern techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46:1011–1030.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Chaoyou Fu, Yuhan Dai, Yondong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. 2024. Video-mme: The first-ever comprehensive evaluation benchmark

[5] `https://openai.com/policies/row-terms-of-use/`

[6] `https://github.com/CaptainCook4D/#license--consent`

of multi-modal llms in video analysis. *arXiv preprint arXiv:2405.21075*.

Google. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. 2023. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*.

Muhammet Ilaslan, Chenan Song, Joya Chen, Difei Gao, Weixian Lei, Qianli Xu, Joo Lim, and Mike Shou. 2023. GazeVQA: A video question answering dataset for multiview eye-gaze task-oriented collaborations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10462–10479, Singapore. Association for Computational Linguistics.

Eleftherios Koutsofios, Stephen North, et al. 1991. Drawing graphs with dot. Technical report, Technical Report 910904-59113-08TM, AT&T Bell Laboratories, Murray Hill, NJ.

Hilde Kuehne, Ali Arslan, and Thomas Serre. 2014. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Duong Le, Ruohao Guo, Wei Xu, and Alan Ritter. 2023. Improved instruction ordering in recipe-grounded conversation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10086–10104, Toronto, Canada. Association for Computational Linguistics.

Linjie Li, Yen-Chun Chen, Yu Cheng, Zhe Gan, Licheng Yu, and Jingjing Liu. 2020. HERO: Hierarchical encoder for Video+Language omni-representation pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2046–2065, Online. Association for Computational Linguistics.

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

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. 2024b. Mmbench: Is your multi-modal model an all-around player? In *ECCV 2024*, pages 216–233, Cham. Springer Nature Switzerland.

Pratyush Maini, Skyler Seto, Richard Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. 2024. Rephrasing the web: A recipe for compute and data-efficient language modeling. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14044–14072, Bangkok, Thailand. Association for Computational Linguistics.

Arjun Majumdar, Anurag Ajay, Xiaohan Zhang, Pranav Putta, Sriram Yenamandra, Mikael Henaff, Sneha Silwal, Paul Mcvay, Oleksandr Maksymets, Sergio Arnaud, Karmesh Yadav, Qiyang Li, Ben Newman, Mohit Sharma, Vincent Berges, Shiqi Zhang, Pulkit Agrawal, Yonatan Bisk, Dhruv Batra, Mrinal Kalakrishnan, Franziska Meier, Chris Paxton, Sasha Sax, and Aravind Rajeswaran. 2024. Openeqa: Embodied question answering in the era of foundation models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Karttikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. 2023. Egoschema: A diagnostic benchmark for very long-form video language understanding. *Advances in Neural Information Processing Systems*, 36:46212–46244.

OpenAI. 2024. Hello gpt-4o.

Arjun Panickssery, Samuel R Bowman, and Shi Feng. 2024. Llm evaluators recognize and favor their own generations. *arXiv preprint arXiv:2404.13076*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Karl Pearson. 1895. Vii. note on regression and inheritance in the case of two parents. *proceedings of the royal society of London*, 58(347-352):240–242.

Rohith Peddi, Shivvrat Arya, Bharath Challa, Likhitha Pallapothula, Akshay Vyas, Jikai Wang, Qifan Zhang, Vasundhara Komaragiri, Eric Ragan, Nicholas Ruozzi, Yu Xiang, and Vibhav Gogate. 2023. CaptainCook4D: A dataset for understanding errors in procedural activities.

Tim J Schoonbeek, Tim Houben, Hans Onvlee, Fons Van der Sommen, et al. 2024. Industreal: A dataset for procedure step recognition handling execution errors in egocentric videos in an industrial-like setting. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4365–4374.

F. Sener, D. Chatterjee, D. Shelepov, K. He, D. Singhania, R. Wang, and A. Yao. 2022. Assembly101: A large-scale multi-view video dataset for understanding procedural activities. *CVPR 2022*.

Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. 2019. Coin: A large-scale dataset for comprehensive instructional video analysis. In *Proceedings of* *the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Ian Wu, Sravan Jayanthi, Vijay Viswanathan, Simon Rosenberg, Sina Khoshfetrat Pakazad, Tongshuang Wu, and Graham Neubig. 2024. Synthetic multimodal question generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12960–12993, Miami, Florida, USA. Association for Computational Linguistics.

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

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2024. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9556–9567.

Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, and Pete Florence. 2022. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Luowei Zhou, Chenliang Xu, and Jason Corso. 2018. Towards automatic learning of procedures from web instructional videos. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

A ProMQA

Figure 8 illustrates the task formulation.[7]

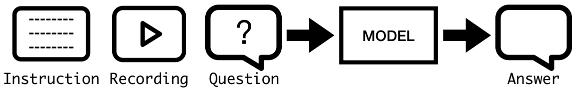

Figure 8: Task formulation of our dataset. Given recipe information, recording information, and a question, a model is to predict an answer. In our benchmark experiment, recipes and questions are fed as text, while recordings are passed as frames sampled from videos. Then, a model generates answers in text.

A.1 External Knowldge

In §2.2, we define that our target multimodal procedural questions can be solvable from the combination of instruction and recording information. Our task assumes the common sense that humans would have gained through their cooking experiences, in varying degrees. We note that this may introduce some ambiguity/subjectivity, regarding the boundary between common sense and external knowledge, as external knowledge is, to some extent, in the same spectrum as common sense. For instance, for well-experienced people, it can be too obvious (common sense) that replacing cilantro with parsley changes the flavor of a recipe, while others would think that is specialized/external knowledge. To mitigate this subjectivity, we assign two annotators for each example in verification to account for this variance.

A.2 Other Verification Criteria

Additionally, we ask annotators to check the naturalness and clarity of questions. Naturalness is to check if a question is natural/makes sense to ask. For instance, when a question like "Did I forget to do something before <stepX>?" is asked, people usually assume that <stepX> has already been passed (with or without errors). So, if the question is asked when <stepX> is yet to be performed, this question will be unnatural/nonsensical. This criterion filters out this type of nonsensical question. Clarity filters out vague/too general questions, especially questions asking about non-procedural aspects. For instance, a question like "What did I do wrong?" can target non-procedural errors, e.g., "Too many dishes are left in the sink." or "The countertop is too messy." which we encountered in our preliminary QA generation and benchmarking experiments. To focus on the procedural questions, we added this criterion.

[7] Icons in Figure 1, 5, 6, and 8 are from `https://www.flaticon.com`.

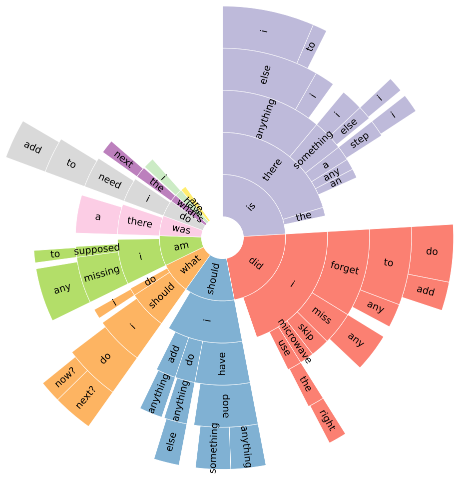

Figure 9: Count of 5 starting words in questions.

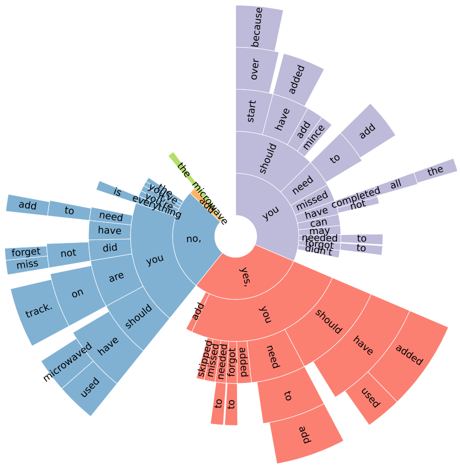

Figure 10: Count of 5 starting words in answers.

A.3 Human-written QAs

We obtain 50 *next* questions by asking one of the annotators before conducting any verification process. This provides the situation where one creates QA pairs without any prior knowledge about this work. They receive raw recipes and videos and create 50 $next$-type QAs from scratch, which took around 4 hours, as shown in Table 5.

A.4 Additional Statistics

In Figure 9 and 10, we show the counts of five starting words in questions and answers, sorted by question types.

In Table 5, we compare the cost between our approach and the full-human annotation approach. In addition, we compare machine-generated and human-written QAs in terms of question diversity using the type-token ratio, TTY ($num\_unique\_words/total\_vocab$), and cosine similarity with E5 Small (Wang et al., 2022). For human-written questions, we use the whole 50 questions to compute both numbers. For machine-generated questions, we sample 50 $next$ questions and compute the metrics. To reduce sampling variance, we sample 10 times and take the average of them. TTY and cosine similarity are 0.07 and 0.80 for human-written questions and 0.09 and 0.80 for machine-generated questions. This suggests that both approaches produce similarly diverse questions at the surface and semantic level.

A.5 Statistical Power Analysis

Following Card et al. (2020), we conduct their statistical power analysis to estimate the performance difference required to detect statistical significance between systems with confidence. We first estimate the baseline accuracy based on the performance of GPT-4o, 0.4 and the agreement rate based on GPT-40 and Claude 3.5 Sonnet, 0.65. Given these numbers, the simulation-based analysis[8] shows that at least 8.5 accuracy point difference would be needed to detect significance with 80% confidence.

B Annotation: *Generate-then-Verify*

B.1 Preprocess

Before the start of our automated preprocessing step, first, we corrected existing annotations in CaptainCook4D, especially about the orders, e.g., by checking the consistency between the order and the timestamps. In the preprocessing stage, we did not create d_{type} of *measurement*, *preparation*, *technique*, *temperature*, *timing*, and *order* from d_{init} when the last recording step did not have the corresponding error annotations. This is due to our preliminary experiments, where such cases tend to generate invalid multimodal procedural questions, i.e., the approval rate was much lower than others. This may be because not all actions can be associated with each type of question. For instance, it is harder to create a sensical *temperature* question from a step, "Peel an onion." In addition, we skip creating d_{type} in the following cases: $video_{0:k}$ is too short, i.e., less than five seconds, which occasionally happens in the case of $video_0$; $missing$ questions for $video_0$; The durations of $S^{video}_{0:k}$ overlaps with s^{video}_{k+1}, as it introduce extra step information in $video_{0:k}$. Also, as for $S^{video}_{0:k}$, we use the "modified description" available in CaptainCook4D, which combines an original step description and its error description of how a user deviates from the corresponding recipe step.

In the sampling, we sample around 200 *next*, 200 *missing*, and 20 other-type examples, approximately reflecting the total number of each type.

All the videos used in ProMQA are from CaptainCook4D, which are released under Apache license 2.0.

B.2 QA Generation

Figure 12 shows a full prompt example, and Figure 13 shows an example DOT representation of a recipe.

In the prompt exploration, the following are our findings: 1) Feeding the recipe as a whole hurts the approval rate compared to the target excerpt. This can be because the LLM needs to do extra reasoning to identify where to focus on a recipe. 2) Feeding the video as frames worsens the approval rate. Videos contain more information to contextualize the generation. However, the result suggests that even for the strong proprietary multimodal models, feeding information as text, if available, leads to better performance. In addition, feeding as frames costs is much more expensive than feeding as text, as models require more tokens to process images.

In the QA generator selection, we noticed that Gemini 1.5 Pro sometimes deviated from the specified format, e.g., additional quotations or tags like "*[question]*" or "*[answer]*".

B.3 Verification

Figure 11 shows the interface for verification. It consists of four parts: 1) A recipe graph with step status (passed as green, current step as orange, and not passed as dotted) with the triangle on the upper left corner of each step indicates that it contains errors. 2) A recording. 3) A list of step and error descriptions. And, 4) QA annotation checkbox, including a comment box for human-written answers. When a question is judged as valid, its answer checkboxes and comment box appear.

We distributed 500 examples to 5 annotators so that each example receives two annotators' judgments ($500 \times 2 = 1000$ judgments) and each pair of annotators (${}_5\mathrm{C}_2 = 10$) shares 50 examples. Based

[8] https://github.com/dallascard/NLP-power-analysis

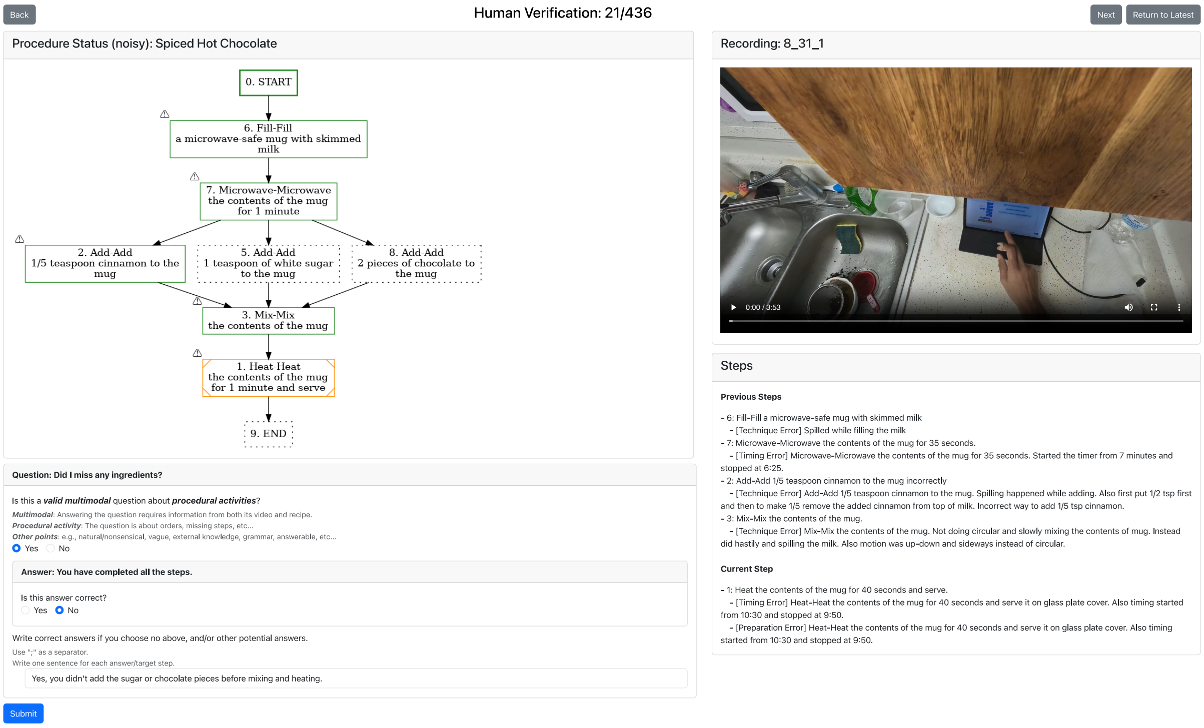

Figure 11: Verification interface.

Missing	Next	Order	Measurement	Preparation	Technique	Temperature	Timing	Avg.
0.82 (148/180)	0.88 (158/179)	0.77 (20/26)	0.81 (17/21)	0.58 (14/24)	0.43 (9/21)	0.62 (10/16)	0.76 (25/33)	0.80 (401/500)

Table 13: Approval rate (#example after/before verification) for each question type.

Annotator 1 $\langle q^m, a_q^m, a_2^m, a_1^h \rangle$	Annotator 2 $\langle q^m, a_q^m, a_2^m, a_2^h \rangle$	Adjudicator $\langle q^m, a_q^m, a_2^m, a_1^h, a_2^h, a_3^h \rangle$	Explanation
⟨ ✓, ✓, ✗, ∅ ⟩	⟨ ✓, ✓, ✗, ∅ ⟩	⟨ –, –, –, –, –, – ⟩	Majority vote & No Adjudication
⟨ ✗, –, –, – ⟩	⟨ ✗, -, -, - ⟩	⟨ –, –, –, –, –, – ⟩	Majority vote & No Adjudication
⟨ ✓, ✓, ✓, ∅ ⟩	⟨ ✓, ✓, ✗, ∅ ⟩	⟨ –, –, ✓/✗, –, –, – ⟩	Majority vote
⟨ ✓, ✓, ✗, ∃ ⟩	⟨ ✓, ✗, ✗, ∃ ⟩	⟨ –, ✓/✗, –, ✓/✗, ✓/✗, – ⟩	Majority vote for q^m, A^m Adjudicator's call for A^h
⟨ ✓, ✓, ✗, ∅ ⟩	⟨ ✗, –, –, – ⟩	⟨ ✓, ✓/✗, ✓/✗, –, –, ∃ ⟩ ⟨ ✗, –,–,–,–,– ⟩	Majority vote for q^m Adjudicator's call for A^m Adjudicator can add A^h

Table 14: Case study of adjudicator's role. Suppose a QA generator generates a question q^m and two answers a_1^m, a_2^m, and then, annotators optionally write human-written answers, a_1^h by one annotator, a_2^h by the other annotator, and a_3^h by the adjudicator. The adjudicator's role changes based on two annotators' judges. (✓: valid, ✗: invalid, ∅: no human-written answer, ∃: human-written answers exist, –: no judge added)

on the shared examples, we calculate the average of per-pair judgment agreements for both questions and answers, 0.87, as discussed in subsection 3.3. Table 13 shows the breakdown approval rate for each question type. In general, GPT-4o generates more valid process-level questions than step-specific questions. Based on our manual inspection, one reason is that some error types are not suitable for multimodal questions. As shown in the table, *preparation* and *technique* produce less valid questions than others. For instance, a step with an error description like "The user peeled the onion

Model	Avg.	w/ Error		Question Type							
		clean	noisy	missing	next	order	measurement	preparation	technique	temperature	timing
Llama 3.1 8B Instruct	25.7	25.9	25.6	35.1	22.6	25.0	5.9	0.0	20.0	16.0	14.3
Llama 3.1 70B Instruct	31.5	33.2	30.2	35.5	38.3	12.5	2.9	14.3	5.6	30.0	20.0
LLaVA 1.5 7B (50f, 288p) & Llama 3.1 8B Instruct	32.9	36.6	30.0	43.0	39.2	10.0	2.9	0.0	15.0	4.0	7.1
LLaVA 1.5 13B (50f, 288p) & Llama 3.1 70B Instruct	37.5	38.9	36.4	41.9	45.3	12.5	11.8	14.3	11.1	50.0	18.0
VideoLLaMA2 7B (8f, 336p)	39.3	45.7	34.2	47.5	47.3	22.5	0.0	0.0	40.0	8.0	14.3
VideoLLaMA2 7B (16f, 336p)	38.3	44.3	33.6	48.1	44.9	30.0	0.0	0.0	30.0	2.0	10.7
VideoLLaMA2 72B (8f, 336p)	39.8	49.4	32.2	46.3	49.7	20.0	0.0	21.4	11.1	25.0	8.0
Qwen2 VL 7B (100f, 336p)	33.8	38.6	30.0	43.4	36.8	30.0	0.0	0.0	30.0	6.0	14.3
Qwen2-VL 72B (100f, 336p)	31.2	32.1	30.4	34.1	37.0	45.0	0.0	10.7	0.0	20.0	14.0
GPT-4o (50f, 765p)	40.4	39.5	41.1	39.9	45.9	45.0	29.4	17.9	27.8	55.0	24.0
GPT-4o (100f, 288p)	38.9	39.2	38.7	37.2	44.0	37.5	32.4	21.4	22.2	50.0	34.0
GPT-4o (250f, 288p)	36.5	38.1	35.3	43.7	34.8	40.0	20.6	22.2	45.0	26.0	10.7
Gemini 1.5 Pro (50f, 765p)	25.2	27.0	23.8	27.4	29.7	15.0	17.6	7.1	16.7	20.0	12.0
Gemini 1.5 Pro (100f, 288p)	27.9	28.1	27.8	32.4	32.0	30.0	2.9	17.9	16.7	15.0	6.0
Gemini 1.5 Pro (250f, 288p)	27.7	30.4	25.6	30.1	31.8	32.5	8.8	22.2	10.0	8.0	25.0
Claude 3.5 Sonnet (10f, 765p)	44.1	48.9	40.4	44.6	58.2	27.5	8.8	14.3	5.6	25.0	28.0
Claude 3.5 Sonnet (100f, 288p)	36.8	43.8	31.3	48.4	37.5	22.5	14.7	16.7	25.0	12.0	10.7
Human	74.5	83.5	65.8	—	—	—	—	—	—	—	—

Table 15: Additional benchmark result: The average of all the examples, the averages of examples with (noisy) and without errors (clean) in previous steps, and the averages for the same question-type examples. f and p denote the number of frames and image resolution used for each model.

Model	Avg.	Answer Source			Answer Type				
		machine	human	both	direct	direct & suggestion	direct & intervention	suggestion	all
Llama 3.1 8B Instruct	25.7	23.4	22.0	33.3	26.4	30.8	9.1	6.2	100.0
Llama 3.1 70B Instruct	31.5	33.4	28.0	30.5	31.7	30.8	34.1	25.0	0.0
LLaVA 1.5 7B (50f, 288p) & Llama 3.1 8B Instruct	32.9	34.1	29.9	32.9	33.2	37.2	27.3	6.2	100.0
LLaVA 1.5 13B (50f, 288p) & Llama 3.1 70B Instruct	37.5	40.7	32.9	34.8	38.8	34.6	34.1	12.5	0.0
VideoLLaMA2 7B (8f, 336p)	39.3	43.7	31.7	36.2	40.9	39.7	25.0	12.5	0.0
VideoLLaMA2 7B (16f, 336p)	38.3	41.8	32.3	35.7	39.4	41.0	27.3	12.5	0.0
VideoLLaMA2 72B (8f, 336p)	39.8	45.1	32.9	34.3	41.7	34.6	31.8	12.5	0.0
Qwen2 VL 7B (100f, 336p)	33.8	36.7	28.0	32.4	35.3	28.2	29.5	12.5	0.0
Qwen2-VL 72B (100f, 336p)	31.2	35.0	22.0	30.5	30.8	33.3	27.3	37.5	100.0
GPT-4o (50f, 765p)	40.4	41.4	29.3	47.1	40.2	39.7	52.3	18.7	50.0
GPT-4o (100f, 288p)	38.9	38.8	27.4	48.1	38.1	38.5	56.8	31.2	0.0
GPT-4o (250f, 288p)	36.5	36.7	26.8	43.8	36.0	39.7	45.5	25.0	0.0
Gemini 1.5 Pro (50f, 765p)	25.2	27.1	18.3	26.7	24.8	25.6	34.1	12.5	50.0
Gemini 1.5 Pro (100f, 288p)	27.9	28.3	20.7	32.9	27.6	34.6	22.7	25.0	0.0
Gemini 1.5 Pro (250f, 288p)	27.7	26.6	28.0	29.5	27.9	29.5	20.5	31.2	0.0
Claude 3.5 Sonnet (10f, 765p)	44.1	48.1	35.4	42.9	45.0	42.3	43.2	25.0	0.0
Claude 3.5 Sonnet (100f, 288p)	36.8	38.8	30.5	37.6	36.6	42.3	38.6	18.7	0.0

Table 16: Additional answer-focused benchmark result breakdown: The average of all the examples, the averages of examples with only machine-generated answer(s), human-written answer(s), and both; The averages of examples with only direct answer(s), direct and suggestion(s), direct and intervention(s), only suggestion(s), and all answer-types. f and p denote the number of frames and image resolution used for each model.

improperly" is unlikely to receive a multimodal question, as it is unlikely that recipes specify the detailed instructions. Another potential reason is the quality of error descriptions in CaptainCook4D. Most of them are sensical, yet, they are not always grammatically correct, e.g., dropping subjects, or detailed enough. Although we corrected the descriptions during our preliminary experiments, they were not exhaustive.

Also, we note that, in the training session for the verification, we received consent from the annotators about the potential release of their annotations.

Adjudication scenarios We set two base principles in designing the adjudicator's role: 1) The adjudicator makes the final judgment for questions/answers when the judgments of two annotators conflict. 2) Every example receives two chances to receive human-written answers. Table 14 shows the role of the adjudicator in different cases. In the first three cases, all judgments are determined by a majority vote. For the fourth one, while a machine-generated question and answers are judged based on a majority vote, human-written answers are judged and determined by the adjudicator's call. These human-written answers are the reasons why we have the two-stage verification process, i.e., to have extra checks even for human-written answers. Also, as two annotators can independently add human-written answers, there may exist duplicate answers, and we did not remove duplicates in the adjudication process. Finally, only in the fifth case, the adjudicator can add human-written answers to comply with our second policy of two chances to receive human-written answers. We note that, as you can see from the case studies, not all examples have both machine-generated and human-written answers. In the adjudication, we shuffle the order of answers in each example to make their sources (machine or human) unclear as a source could give extra bias to the adjudicator.

C Benchmarking

C.1 LLM-as-a-Judge

Figure 14 shows one full prompt example for our LLM-based scoring.

C.2 Experimental Details

We use `vllm` for the inference of Llama 3.1 and LLaVA 1.5. For VideoLLaMA2[9] and Qwen2-VL.[10], we follow their instructions to run their respective inference code. All the weights are downloaded from *HuggingFace*[11] using `transformers`. We use 1~4 GPUs of A6000 (48GB), depending on the size of the models. Each inference took at most a few hours. For all proprietary models, we use their libraries to make API calls. Each prediction on all of our 401 examples in ProMQA costs 30~60 USD, depending on the model, the number of frames, and the resolution of each frame. All the results are based on a single run.

C.3 Additional Results

Table 15 and 16 show the additional benchmarking results by changing model size for open models and changing the number of frames and resolutions for proprietary models. The unimodal and Socratic models improve their performance as the sizes of their models increase. However, open multimodal models did not change the overall performance or even lowered their performance by a few points. As for proprietary models, under the fixed maximum input length, the number of frames trades off the resolutions. In our experiment, we found that higher resolution leads to better performance. However, the combinations we tried are rather limited, and there may exist a better combination, which we leave the exploration for future work. We also note here that we faced issues with limited maximum lengths with image-included prompts, compared to the ones listed on each API documentation or the ones when we tried text-only prompts. Presumably, this is due to the large file size of each image and the total data size of one input for each API request. We also leave it for future work on the workaround of how to feed many relatively high-resolution images in input prompts.

[9] `https://github.com/DAMO-NLP-SG/VideoLLaMA2`
[10] `https://github.com/QwenLM/Qwen2-VL`
[11] `https://huggingface.co`

```
# Instruction
A person is cooking Spiced Hot Chocolate with their friend, who is a skilled cook.
The person completed these steps:
- Fill a microwave-safe mug with whole milk but spill
- Microwave the contents of the mug for 2 minutes
- Add-Add 4 pieces of chocolate to the mug
- Add-Add 1 teaspoon of white sugar to the mug
And, the person has just performed this step:
- Mix-Mix the contents of the mug
The friend knows the following step(s) can be done next:
- Heat-Heat the contents of the mug for 1 minute and serve
The person may or may not be noticing this.
What questions would the person ask the friend about next step(s)?

Assuming the friend is watching over you throughout the cooking activity and
understand the situation, return three pairs of a question and its answers as a
list:
* <questions>
    * <answer1-1>
    * <answer1-2>
    * ...
* <question2>
    * ...

# Note
- Each question/answer should consists of one consice sentence/phrase.
- If there exist multiple correct answers, provide all correct answers for each
    question as a list so that each answer targets at one step.
- Each answer targets at one step. - Imagine a variety of a person: beginner/
    experienced, careless/careful, etc...
- It is preferable to have as diverse pairs (question/answer type, tone, wording,
    etc) as possible.
- There is a case where no missing step is performed, i.e., an answer is just no.

# Example
* What is the next step?
    * You have completed all the steps.
* What should I do next?
    * <stepY>
    * <stepZ>

# Response
```

Figure 12: Prompt example for QA generation: $next$ question

```
digraph G {
    START; "Heat-Heat the contents of the mug for 1 minute and serve";
    "Add-Add 1/5 teaspoon cinnamon to the mug";
    "Mix-Mix the contents of the mug";
    "Add-Add 1 teaspoon of white sugar to the mug";
    "Fill-Fill a microwave-safe mug with skimmed milk";
    "Microwave-Microwave the contents of the mug for 1 minute";
    "Add-Add 2 pieces of chocolate to the mug";
    END;

    "Mix-Mix the contents of the mug" -> "Heat-Heat the contents of the mug for 1
        minute and serve";
    "Add-Add 2 pieces of chocolate to the mug" -> "Mix-Mix the contents of the mug
        ";
    "Add-Add 1 teaspoon of white sugar to the mug" -> "Mix-Mix the contents of the
        mug";
    "Add-Add 1/5 teaspoon cinnamon to the mug" -> "Mix-Mix the contents of the mug
        ";
    "Microwave-Microwave the contents of the mug for 1 minute" -> "Add-Add 1
        teaspoon of white sugar to the mug";
    START -> "Fill-Fill a microwave-safe mug with skimmed milk";
    "Heat-Heat the contents of the mug for 1 minute and serve" -> END;
    "Microwave-Microwave the contents of the mug for 1 minute" -> "Add-Add 2 pieces
         of chocolate to the mug";
    "Microwave-Microwave the contents of the mug for 1 minute" -> "Add-Add 1/5
        teaspoon cinnamon to the mug";
    "Fill-Fill a microwave-safe mug with skimmed milk" -> "Microwave-Microwave the
        contents of the mug for 1 minute";
}
```

Figure 13: Prompt example of a recipe in DOT format: “Spiced Hot Chocolate”

```
# Instruction
This is an evaluation task.
You will be given a question, gold answer(s), and predicted answer.
Your task is to evaluate if the predicted answer matches against the gold answer(s).

Give your ternary judge 0, 1, or 2:
* 0 means the predicted answer is wrong (unmatch)
* 1 means the predicted answer is partially correct/wrong (partial match)
* 2 means the predicted answer is correct (match)
When multiple gold answers are available (provided as a list), the predicted answer is correct/partially correct if it matches/partially matches with at least one of the gold answers.

Provide your feedback as follows:
# Feedback
[Rationale] (your rationale for the judge, as a text)
[Judge] (your judge, as a number, 0, 1, or 2)

# Note
The question is being asked by a user who is cooking Cucumber Raita.
Well-trained annotators constructed gold answer(s), while the predicted answer was by a machine, which answered based on the corresponding recipe and the frames of the cooking recording.

Here are the steps being performed already:
- Add-Add 1 teaspoon of cumin powder to the bowl
- add-add 1 tablespoon of chopped scallions to the bowl instead of cilantro
- Rinse-Rinse 1 medium sized zucchini
- Add-1/4 teaspoon of red chilli powder to the bowl
- whisk-In a mixing bowl, whisk 1 cup of chilled curd until smooth. Use fresh homemade or packaged curd
- chop or grate-chop or grate only 1/2 of zucchini instead of one medium cucumber

# Task
Now, here are the question, gold answer(s), and predicted answer:
[Question]
- Did I forget any other ingredients?
[Gold Answer(s)]
- No, you did not forget any ingredients at the moment.
[Predicted Answer]
- Based on the images, it seems you forgot to add 1/2 teaspoon of chaat masala powder.

# Feedback
[Rationale]
```

Figure 14: Prompt example for evaluation.